Qucs: Current and planned circuit simulation and device modelling developments; a review

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
Felix Salfelder [6], felix<notifications@github.com>
Frans Schreuder [7], fransschreuder@gmail.com
Guilherme Brondani Torri [4], guitorri@gmail.com

[1]Centre for Communications Technology, London Metropolitan University, UK
[2]The University of Edinburgh, UK
[3]Bauman Moscow Technical University, Russia
[4]Qucs Developer
[5]Laboratoire SATIE — CNRS UMR 8929, Université de Cergy-Pontoise, ENS Cachan, FR
[6]Gnucap and Qucs Developer
[7]Nikhef, Amsterdam, NL

Plus contributions from the Qucs "User Community"

Presented at the MOS-AK DATA Workshop, Dresden, 18 March 2016

Qucs: Current and planned circuit simulation and device modelling developments; a review

- Qucs-0.0.19 and Qucs-0.0.19-S-RC4:
    - 1. Background and release details
    - 2. Review of changes and improvements
- Qucs Verilog-A modelling: current position and the way forward with AMS and compact semiconductor device models?
    - 1. Background - free non-GPL model licences?
    - 2. New release of ADMS under GPL 3
    - 3. Possible solution to compact device modelling problems
- Qucs development after release 0.0.19: the way forward to Qucs-0.0.20; more improvements and merging of Qucs and Qucs-S branches
    - 1. Qucs-0.0.19: work in progress
    - 2. Qucs-0.0.19-S: work in progress
    - 3. Structure after merging Qucs-0.0.19 and Qucs-0.0.19-S ? - (Wish list)
    - 4. Integrated compact modelling capabilities
- Summary

Qucs-0.0.19: background, features and release details

Background

Features
- GUI/IDE
- Schematic capture

Simulation tools
- Qucsator
- Optimizer (ASCO)
- Icarus-Verilog
- FreeHDL (VHDL)
- Data visualization
- Equation system
- Component library
- Design/synthesis tools

Extensible
- SPICE netlist import
- Verilog-A model builder
- Octave/MATLAB support

Dependencies
- C++ compiler
- Qt4 (with Qt3 support)
- Autotools / CMake
- gperf / flex / bison
- ADMS
- LaTex

Operation

Qucs schematic
- Schematic to netlist
- Schematic to print
- Dump components data

Custom file formats
- Schematic
- Library
- Netlist
- Data file

Qucsator simulator
- DC
- Transient
- AC
- AC Noise
- S-Parameter
- S-Parameter Noise
- (Harmonic Balance)

Qucsconv converter
- SPICE to Qucs
- SPICE to Qucslib
- vcd to Qucsdata
- Qucsdata to csv
- Qucsdata to Touchstone
- citi to Qucsdata
- Touchstone to Qucsdata
- csv to Qucsdata
- zvr to Qucsdata
- mdl to Qucsdata
- Qucsdata to MATLAB/Octave

Release date and improvements

Release 0.0.19 (February 05, 2016)

Bug fixing, usability improvements, build system cleanup

Ongoing port Qt3Support to Qt4

New active-filter synthesis tool

Integration of regression tests, qucs-test repository

Removal of non-GPL models

Release locations

1) Source tarball:

http://sourceforge.net/projects/qucs/files/qucs/0.0.19-snapshots/qucs-0.0.19-160204-git-83cc216.tar.gz

The latest build instructions can be found at: https://github.com/Qucs/qucs/blob/master/README.md

On Windows, perhaps the easiest way is to use MSYS2. The AppVeyor script gives a good idea on how to proceed: https://github.com/Qucs/qucs/blob/master/.appveyor.yml

2) Binaries for Windows 64bit:

http://sourceforge.net/projects/qucs/files/qucs-binary/0.0.19-snapshots/qucs-0.0.19-160204-git-83cc216-win64.zip

http://sourceforge.net/projects/qucs/files/qucs-binary/0.0.19-snapshots/qucs-0.0.19-160204-git-83cc216-win64.exe

Qucs-0.0.19-S: background, features and release details

Features
- GUI/IDE
- Schematic capture

Simulation tools
- Qucsator
- Ngspice
- Xyce (serial)
- Xyce (parallel)
- SpiceOpus
- Optimizer (ASCO)
- Icarus-Verilog
- FreeHDL (VHDL)
- Data visualization
- Equation system
- Component library
- Design/synthesis tools

Extensible
- SPICE netlist import
- Verilog-A model builder
- Verilog-A model synthesizer
- Octave/MATLAB support

Dependencies
- C++ compiler
- Qt4 (with Qt3 support)
- Autotools / CMake
- gperf / flex / bison
- ADMS
- LaTex

Background

Qucs schematic
- Schematic to netlist
- Schematic to print
- Dump components data

Custom file formats
- Schematic
- Library
- Netlist
- Data file

SPICE specific commands
- .PARAM .GLOBAL
- .OPTIONS .IC
- .NODESET .MODEL
- .INCLUDE

nutmeg (ngspice)

Qucsator
- DC
- Transient
- AC
- AC Noise
- S-Parameter
- S-Parameter Noise
- (Harmonic Balance)

SPICE
- DC
- Transient
- AC
- Noise
- Harmonic Balance (Xyce)
- Fourier
- Distortion
- Pole-zero
- Custom (Ngspice)

Qucsconv converter
- SPICE to Qucs
- SPICE to Qucslib
- vcd to Qucsdata
- Qucsdata to csv
- Qucsdata to Touchstone
- citi to Qucsdata
- Touchstone to Qucsdata
- csv to Qucsdata
- zvr to Qucsdata
- mdl to Qucsdata
- Qucsdata to MATLAB/Octave

Operation

Release 0.0.19-S-rc4 (January, 2016)
- Added Pole-Zero analysis with Ngspice
- Added XSPICE analogue devices
- Added magnetic core models
- Added .MODEL and .INCLUDE directives support
- Added new Transformers library for Qucs+Ngspice
- Unified SPICE components icons
- Added PlotVs() emulation for SPICE
- Fixed different bugs including bugfixes from Qucs-0.0.19

Release date and improvements

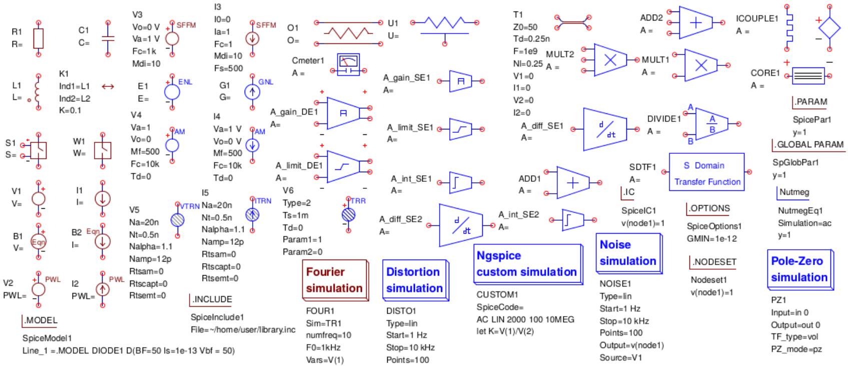
1) Source tarball: https://github.com/ra3xdh/qucs/releases/tag/0.0.19S-rc4qucs-0.0.19S-rc4.tar.gz
   Source code: https://github.com/ra3xdh/qucs/releases/tag/Source code (zip)
   https://github.com/ra3xdh/qucs/releases/tag/Source code (tar.gz)
   Follow official build instructions at https;//github.com/Qucs/qucs to build executables from the source tarball.

2) Binary for Windows 64bit: https://github.com/ra3xdh/qucs/releases/tag/qucs-0.0.19S-rc4-setup.exe

3) spice4qucs-help documentation: https://qucs-help.readthedocs.org/en/spice4qucs/index.html

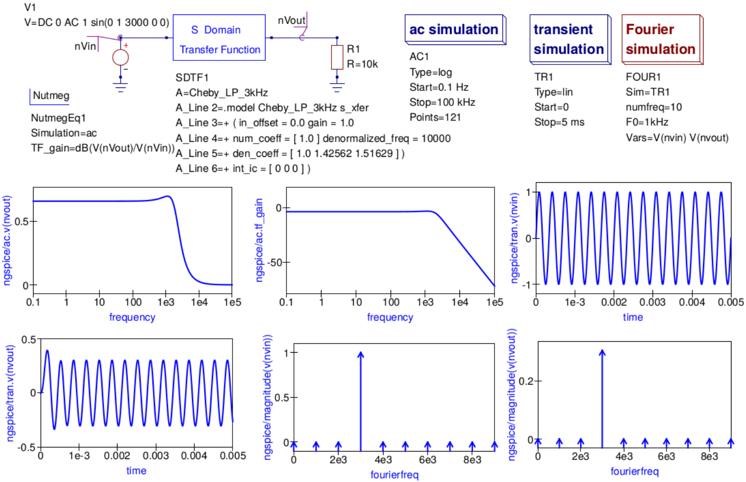
Release locations

Qucs-0.0.19-S additional SPICE and XSPICE component, control and simulation icons

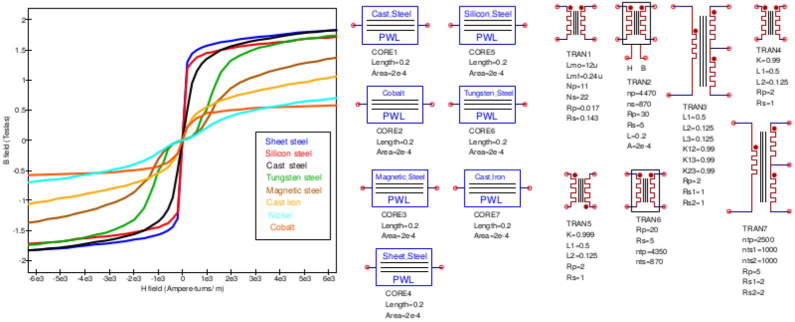


→ Ngspice and Xyce → Ngspice and SPICEOPUS Symbol Key

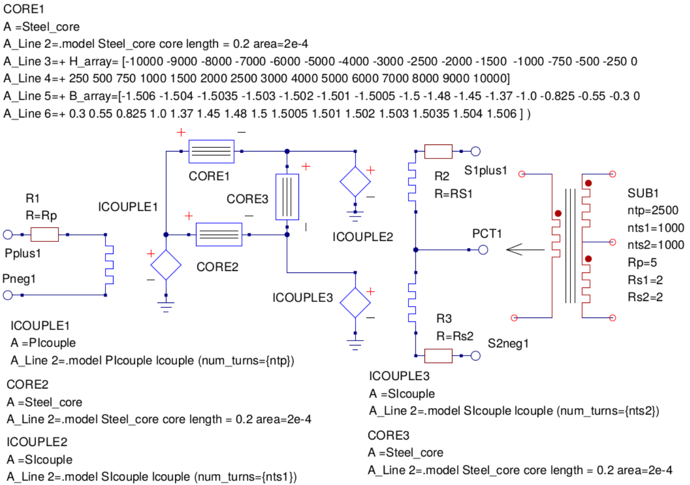
Qucs-0.0.19-S XSPICE standard models: (1) S domain transfer function block



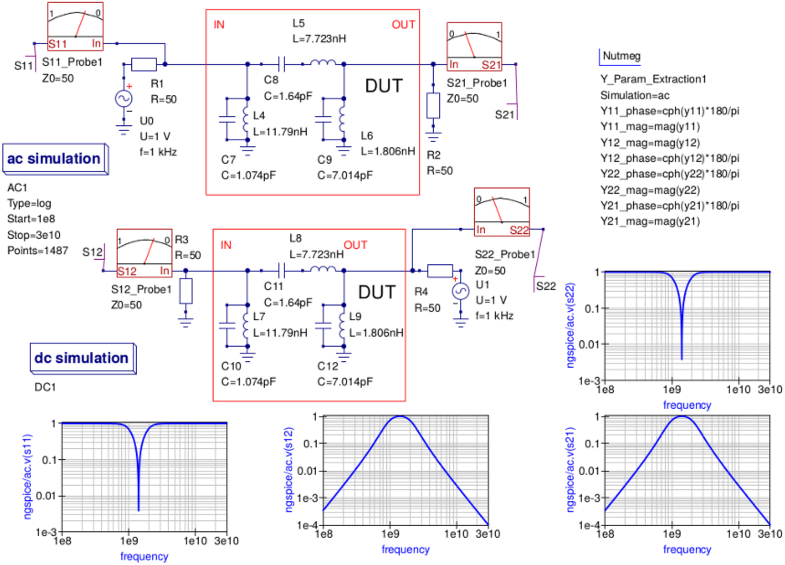
Qucs-0.0.19-S XSPICE standard models: (2) non-linear transformer and magnetic core blocks



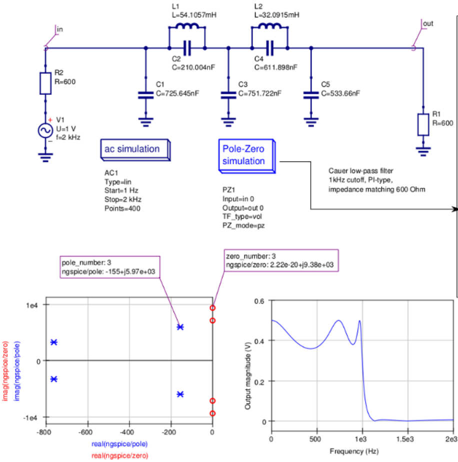
Qucs-0.0.19-S XSPICE standard models: (3) non-linear transformer and magnetic core block macromodel example



Qucs-0.0.19-S two port network analysis: new probes, parameter conversion subcircuits and nutmeg parameter conversion blocks



Qucs-0.0.19-S Ngspice and Xyce new simulation features: (1) An example Pole-Zero analysis with Qucs and Ngspice



Auto-generated netlist

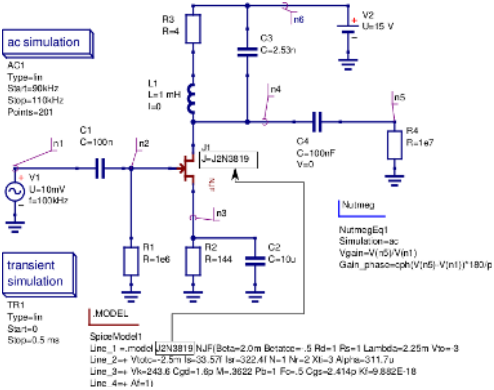
```
* Qucs 0.0.19  /home/vvk/.qucs/RCL_PZ.sch
V1 _net0 0 DC 0 SIN(0 1 2K 0 0) AC 1
R1 0 out  600
R2 _net0 in  600
C1 0 in  725.645N
C2 in _net1  210.004N
L1 in _net1  54.1057M
C3 0 _net1  751.722N
C4 _net1 out  611.898N
L2 _net1 out  32.0915M
C5 0 out  533.66N
.control
set filetype=ascii
echo "" > spice4qucs.cir.noise
echo "" > spice4qucs.cir.pz
ac lin 400 1 2k
write RCL_PZ_ac.txt v(in) v(out)
destroy all
reset

pz in 0 out 0 vol pz
echo "PZ analysis" >> spice4qucs.cir.pz
let dummy_var = 0.0
print all >> spice4qucs.cir.pz
destroy all
reset

exit
.endc
.END
```

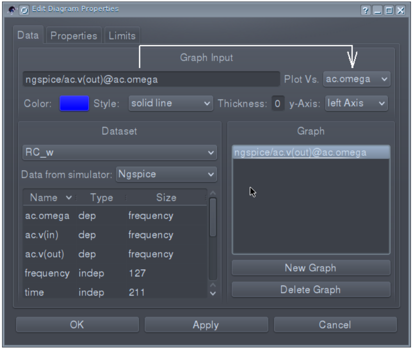
Qucs-0.0.19-S Ngspice and Xyce new simulation features: (2) SPICE .MODEL directive support

- The SPICE .MODEL directive allows use of unmodified SPICE modelcards provided by electronic devices manufacturers
- Place a .MODEL directive on a schematic then copy a SPICE model from a component datasheet, finally pasting it on the .model directive
- An example of .MODEL attachment for a JFET



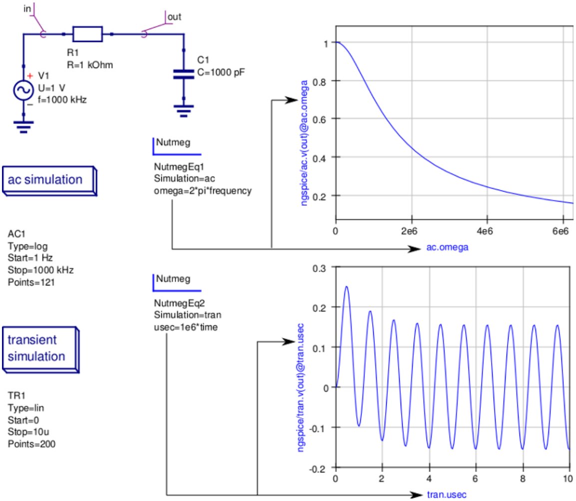
Qucs-0.0.19-S Ngspice and Xyce new simulation features: (3) Qucs PlotVs() support

- Ngspice has no PlotVs() equivalent for the generation of user defined data plots
- With Qucs-0.0.19-S the PlotVs() processing function has been moved to the GUI level in order to provide this feature. Qucs-0.0.19-S makes use of the @ symbol to specify X-variable



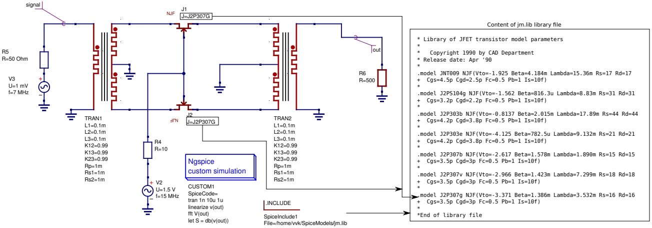
Qucs-0.0.19-S Ngspice and Xyce new simulation features: (3) Qucs PlotVs() support continued

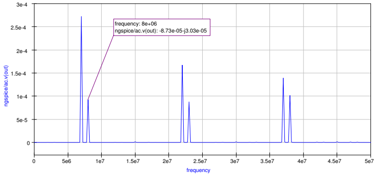
- An example of PlotVs() usage for frequency and time-domain simulation



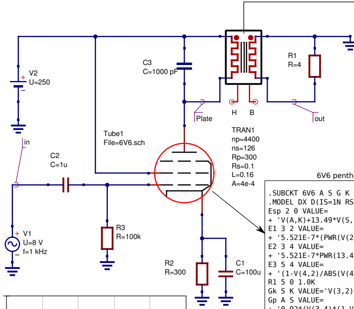
Qucs-0.0.19-S Ngspice and Xyce new simulation features: (4) FFT spectrum analysis with Nutmeg scripting

- Use nutmeg scripts with "Ngspice Custom simulation" to obtain the output spectrum
- .INCLUDE directive allows attachment of unchanged SPICE libraries to a schematic with the new SPICE-compatible devices symbols





Qucs-0.0.19-S Ngspice and Xyce new simulation features: (5) Single ended thermionic valve (tube) amplifier; demonstration of SPICE model usage and the new XSPICE transformer library

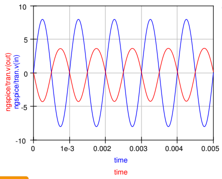


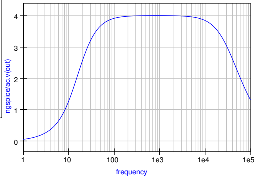
Output transformer library model

```
.SUBCKT Transformers_LossyTransformer2 gnd nPlus
+ nPneg nSplus nSneg nH nB np=4470 ns=870
+ Rp=30 Rs=5 L=0.2 A=2e-4
L3 ndB  0 1
B3 0  n1  I = pwl ( V(nH), -10000, -1.506, -9000, -1.504, -8000,
+ -1.5035, -7000, -1.053, -6000, -1.502, -5000, -1.501,
+ -4000, -1.5005, -3000, -1.5, -2500, -1.48, -2000, -1.45, -1500, -1.37,
 -1000, -1.0, -750, -0.825, -500, -0.55, -250, -0.3,
+ 0, 0, 250, 0.3, 500, 0.55, 750, 0.85, 1000, 1.0,
+ 1500, 1.37, 2000, 1.45, 2500, 1.48,
+ 3000, 1.50, 4000, 1.5005, 5000, 1.501, 6000, 1.502, 7000, 1.503,
+ 8000, 1.5035, 9000, 1.504, 10000, 1.506)
R4 0  nH 1
ESRC1 nP2 nPneg ndB 0 {np*A}
ESRC2 nS2 nSneg ndB 0 {ns*A}
R1 nP1  nPlus Rp
R2 nSplus  nS1 Rs
HSRC3 nB 0  VSRC3 1
VSRC3 n1 ndB DC 0
HSRC4 nIp 0  VSRC4 1
VSRC4 nP1 nP2 DC 0
HSRC5 nIs 0  VSRC5 1
VSRC5 nS1 nS2 DC 0
BD2I0 0 nH I=({np}*V(nIp)+{ns}*V(nIs))/{L}
.ENDS
```

6V6 penthode model

```
.SUBCKT 6V6 A S G K
.MODEL DX D(IS=1N RS=1 CJO=1PF TT=1N)
Esp 2 0 VALUE=
+ 'V(A,K)+13.49*V(S,K)+130.4*V(G,K)'
E1 3 2 VALUE=
+ '5.521E-7*(PWR(V(2),1.5)+PWR(V(2),1.5))/2'
E2 3 4 VALUE=
+ '5.521E-7*PWR(13.49*V(S,K),1.5)*V(A,K)/25'
E3 5 4 VALUE=
+ '(1-V(4,2)/ABS(V(4,2)+0.001))/2'
R1 5 0 1.0K
Gk S K VALUE='V(3,2)'
Gp A S VALUE=
+ '0.92*(V(3,4)*(1-V(5,4))+V(3,2)*V(5,4))'
R3 G 10 3k
* FOR GRID CURRENT
D3 10 K DX
* FOR GRID CURRENT
Cgk G K 4.5P
Cgs G S 4.5P
Cgp G A 0.7P
Cpk A K 7.5P
.ENDS
```





ac simulation

AC1
Type=log
Start=1 Hz
Stop=100 kHz
Points=101

transient simulation

TR1
Type=lin
Start=0
Stop=5 ms

Qucs Verilog-A modelling: current position and the way forward with ADMS and compact semiconductor device models?

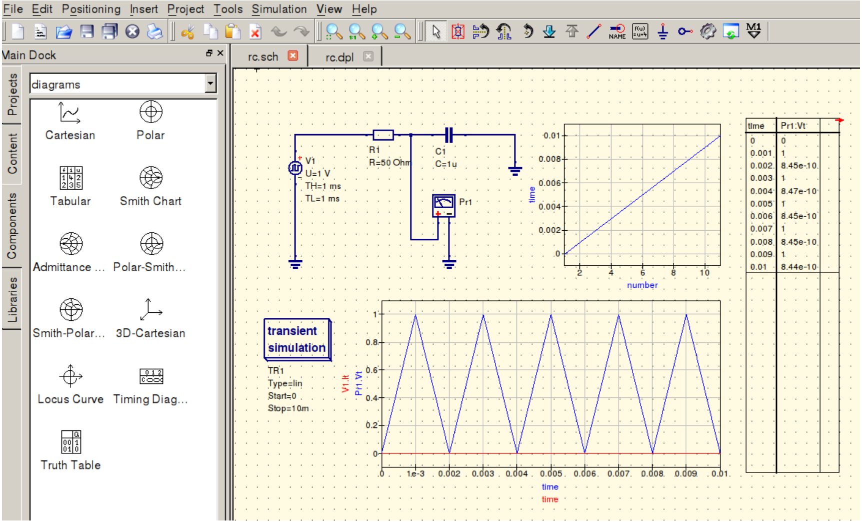
- 1. Background - free non-GPL model licences?
  Problems: Free model code but not GPL because, for example:
  agree not to charge, under GPL the code is free, but people can charge for a nicely packaged source, as long there are other ways to get the source for free;
  agree to acknowledge ... in the documentation, this originates in the original BSD license, the advertising clause, which is incompatible with GPL and has been removed in new BSD licenses.
  agree to obey all government restrictions, *which government*? this adds on top of what is allowed by GPL, which makes it incompatible;

  Non-GPL Verilog-A device models removed from Qucs-0.0.19 as a temporary measure until a solution can be found. The same action will take place for Qucs-0.0.19-S when it is finally formally released or merged with Qucs-0.0.19.
- 2. New release of ADMS under GPL 3 - version adms-2.3.5
  Add new simplified constants.vams and discplines.vams. Tested to work with models currently in use by Qucs, Ngspice, Xyce and Gnucap. Whenever these headers are used, adms informs the user about the availability of the standard headers at: http://accellera.org/downloads/standards/v-ams.
- 3. Possible solution to compact device modelling problems
  In the future the Qucs team will try to work around the model license issues and provide Qucs users with a way to load these models dynamically. No solution ready yet.

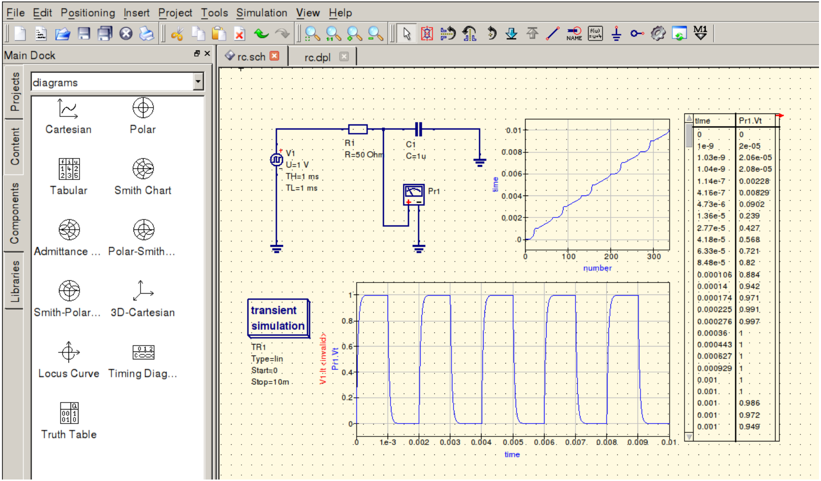
QUCS + GNUCAP by Felix Salfelder

- Proof of concept based on old code fragments by Fabian Vallon,
- **gnucsator**, a gnucap based qucsator supplemented by means of:
- Input deck parser (qucsator input),
- A few compatible components,
- Command and semantics emulation (noninteractive, one-shot, probe-placement)
- GNUCAP simulation data translated into Qucs dataset format ('.dat')
- Independent implementation, work in progress, see https://www.github.com/QUCS/gnucsator
- For proper integration: more work on both the front and back ends of the package needs to be done.

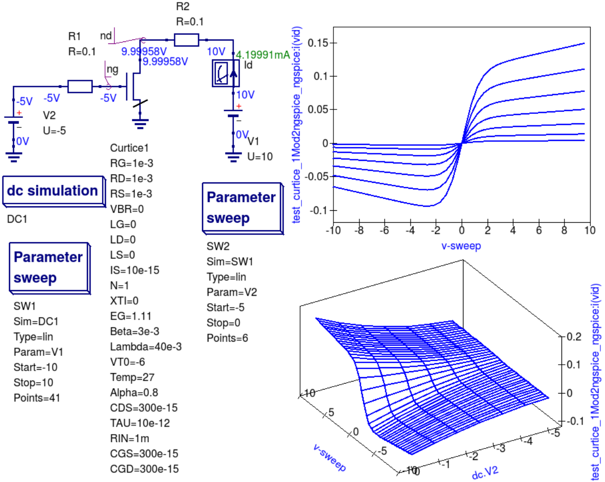
Work in progress :Qucs+GNUCAP;$ QUCSATOR=qucsator qucs -i rc.sch



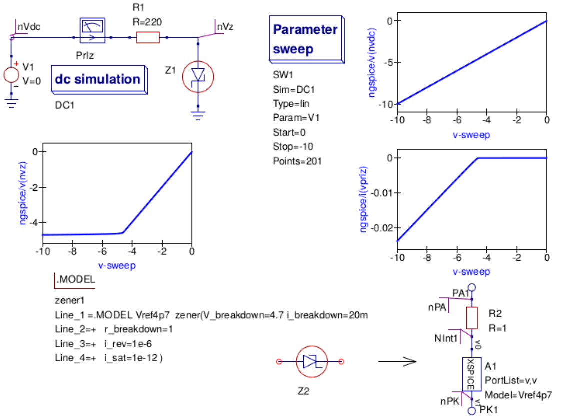
Work in progress :Qucs+GNUCAP;$ QUCSATOR=gnucsator.sh qucs -i rc.sch



Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 1 DC bias display on a schematic - press key F8



Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 2 XSPICE code model subcircuits



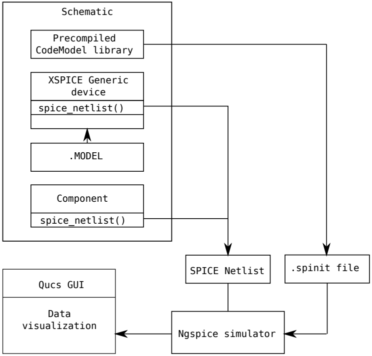

```
* Qucs 0.0.19  Zener4p7.sch
.SUBCKT Zener4p7 nPA nPK
A NInt1 nPK Vref4p7
R1 NInt1  nPA 1

.MODEL Vref4p7  zener(V_breakdown=4.7
+  i_breakdown=20m
+  r_breakdown=1
+  i_rev=1e-6
+  i_sat=1e-12 )
.ENDS
VPrIz nVdc _net0 DC 0 AC 0
R1 nVz  _net0 220
V1 nVdc  0 0
XZ1  nVz 0 Zener4p7
.control
set filetype=ascii
echo "" > spice4qucs.cir.noise
echo "" > spice4qucs.cir.pz
dc v1 0 -10 -0.0497512
write Test_zener4p7_dc.txt VPrIz#branch v(nVdc) v(nVz)
destroy all
reset
exit
.endc
.END
```

Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 3 XSPICE CodeModel support subsystem

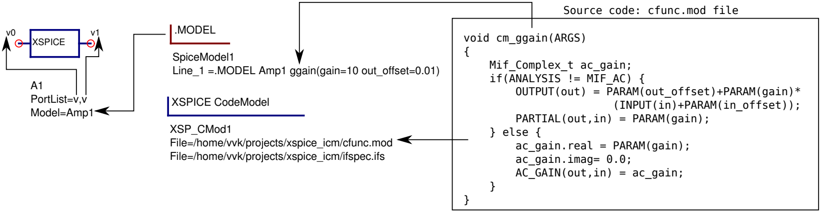
- The "XSPICE generic device" component is the foundation for
  - Precompiled XSPICE device (*.cm) library support, and
  - Dynamic XSPICE CodeModels compilation system which allows **CodeModel sources to be attached to a schematic and compiled automatically at simulation time**.

- Precompiled CodeModel *.cm library attachment dataflow diagram

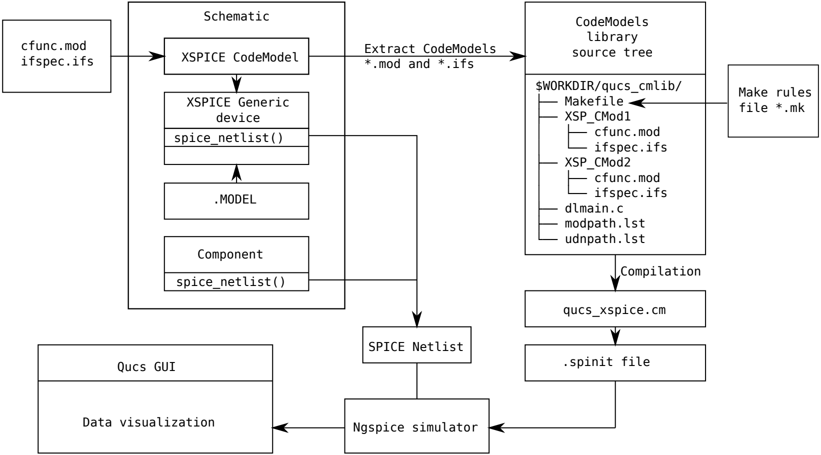


Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 4 "XSPICE generic device" component

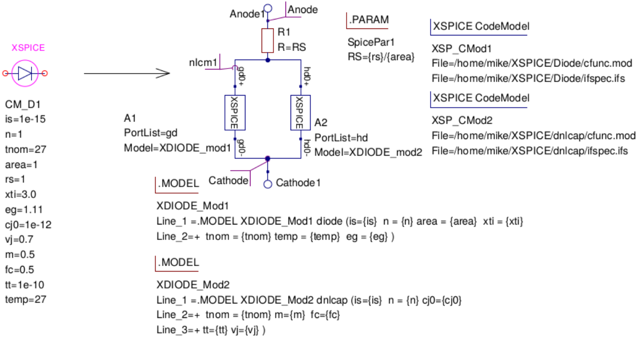
- The "XSPICE generic device" component is a building block for the construction of user-defined A-devices. It is defined by a comma separated port list, with allowed XSPICE port designators, then attached to a SPICE .MODEL statement



Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 5 XSPICE "turn-key" model generation; compiler system dataflow diagram



Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (a) The Qucs-S subcircuit symbol and model circuit



XSPICE diode model based on:

1. P. Antognetti and G. Massobrio (Editors), "Semiconductor device modeling with SPICE, 1988, McGraw-Hill Book Componey, New York, pp1-32.
2. S. Jahn amd M.E. Brinson, "Interactive device modelling using Qucs equation-defined devices., 2008, International Journal of Numerical Modelling: Electrical Networks, Devices and Fields, 21:335-249, DOI: 10.1002/jnm.676.

Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (b) The XSPICE Diode/func.mod code

```
/*
   diode cm model.     4 March 2016    Mike Brinson

  This file contains the mode code for an experimental semiconductor diode model.
  This is  used as a test bench for constructing compact device models
  using the Qucs-0.0.19-S automatic XSPICE CodeModel compiler system.

  This is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation; either version 2, or (at your option)
  any later version.
*/
#define DERIVE 0
#include <math.h>
void cm_diode(ARGS)
{
  double Vt_temp, Vd, P1, P3, P4, PTNOM, PTEMP;
  double PIS, PAREA, PXTI,  PEG,  PN;
  double Tr, Is_temp, Id;
  double *derive;
  double exp80 = 5.5406334e34;
  double GMIN = 1e-12;

   PTNOM = PARAM(tnom)+273.15;
   PTEMP = TEMPERATURE+273.15;
   Vt_temp = 8.65387195e-5*PTEMP;
   PEG = PARAM(eg);
   PIS  = PARAM(is);
   PN   = PARAM(n);
   PAREA = PARAM(area);
   PXTI  = PARAM(xti);

   P1 = 1/(PN*Vt_temp);
   Tr = PTEMP/PTNOM;
   Is_temp = PAREA*PIS*exp( (PXTI/PN)*log(Tr))*exp( (-PEG/Vt_temp)*(1.0-Tr));
   P3 = -5*PN;
   P4 = Is_temp*exp80;

  if(INIT) {
     cm_analog_alloc(DERIVE, sizeof(double));
     derive = (double *)cm_analog_get_ptr(DERIVE, 0);
     *derive = 0.0;
   }
  else {
       derive = (double *)cm_analog_get_ptr(DERIVE, 0);
     }

  if(ANALYSIS != AC) {
     Vd = INPUT(diode);
     if  ( Vd > P3*Vt_temp ) {
        if (P1*Vd <= 80)  {
          Id = Is_temp*(exp(P1*Vd)-1.0) + GMIN * Vd;
          OUTPUT(diode) = Id;
          *derive = P1*Is_temp*exp(P1*Vd)+GMIN;
          PARTIAL(diode, diode) = *derive;
        }
        else {
          Id = Is_temp*exp80*(1+(P1*Vd-80))+GMIN*Vd;
          OUTPUT(diode) = Id;
          *derive = P1*P4+GMIN;
          PARTIAL(diode, diode) = *derive;
        }
     }
     if (  Vd <= -5*PN*Vt_temp){
       Id = -Is_temp+GMIN*Vd;
      OUTPUT(diode) = Id;
       *derive = GMIN;
       PARTIAL(diode, diode) = *derive;
     }
  }
}
```

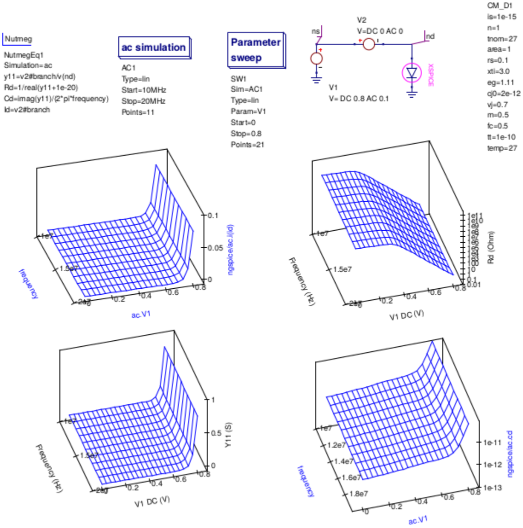
Semiconductor diode non-linear I_d / V_d characteristics, including Verilog-A limexp function and temperature effects.

Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (c) The XSPICE non-linear diode capacitance dnlcap/func.mod code

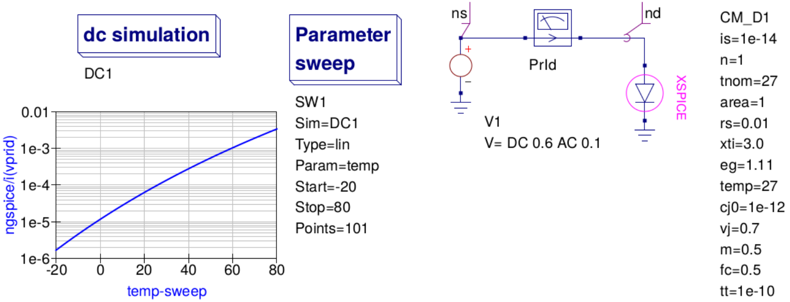
```c
/*  dnlcap cm model.     4 March 2016    Mike Brinson
   This file contains the model code for an experimental
   semiconductor diode capacitance: both Cdep and Cdiff are modelled.
   This is  used as a test bench for constructing compact device models
   with the Qucs-0.0.19-S automatic XSPICE CodeModel compiler system.
   This is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.
*/
#define CVC 0
#include <math.h>
void cm_dnlcap(ARGS)
{
  Complex_t ac_gain;
  static double PCJ0, PVJ, PM, PFC, PTT, PIS, PN;
  double P1, Vd, partial, Vt_temp;
  double PTEMP, W1, Wr, Wi, Rd;
  double *cvc;
  static double cap, F2, F3, cdep, derive, Id, P3, P4;
  double Rp = 1.0e12;
  double exp80 = 5.5406334e34;
  double GMIN = 1e-12;
  PTEMP = TEMPERATURE+273.15;
   Vt_temp = 8.65387195e-5*PTEMP;
   P1   = 1/(Vt_temp);
   if(INIT==1) {
     cm_analog_alloc(CVC, sizeof(double));
     cvc = (double *) cm_analog_get_ptr(CVC, 0);
     *cvc = 0.0;
     cap = 1e-16;
     derive = 1e-20;
     PCJ0 = PARAM(cj0);
     PVJ  = PARAM(vj);
     PM   = PARAM(m);
     PFC  = PARAM(fc);
     PTT  = PARAM(tt);
     PIS  = PARAM(is);
     PN   = PARAM(n);
     F2 = exp( (1+PM)*log(1-PFC) );
     F3 = 1-PFC*(1+PM);
     P3 = -5*PN;
     P4 = PIS*exp80;
    }
  else {
    cvc = (double *) cm_analog_get_ptr(CVC,0);
    }
  if (ANALYSIS != AC) {
   Vd =  *cvc;
   if  ( Vd > P3*Vt_temp ) {
           if (P1*Vd <= 80)  {
             Id = PIS*(exp(P1*Vd)-1.0) + GMIN * Vd;
             derive = P1*PIS*exp(P1*Vd)+GMIN;
            }
            else {
             Id = P4*(1+(P1*Vd-80))+GMIN*Vd;
             derive = P1*P4+GMIN;
            }
    }
    else {
      Id = -PIS+GMIN*Vd;
      derive = GMIN;
     }
   if ( Vd < PFC*PVJ ) {
       cdep = PCJ0/exp(PM*log(1.0 - (Vd/PVJ)) );
    }
    else {
        cdep = (PCJ0/F2)*(F3+(PM*Vd/PVJ));
    }
    cap = PTT*Id/Vt_temp + cdep;
  }
  if (ANALYSIS == DC) {
    *cvc = INPUT(dnlcap)*Rp;
    OUTPUT(dnlcap) =  *cvc;
    PARTIAL(dnlcap, dnlcap) =  Rp;
  }
 if (ANALYSIS == TRANSIENT) {
    cm_analog_integrate(INPUT(dnlcap) / (cap + 1e-17), cvc, &partial);
    partial /= cap;
    OUTPUT(dnlcap) = *cvc;
    PARTIAL(dnlcap, dnlcap) = partial;
 }
 if (ANALYSIS == AC) {
    Rd = 1/derive;
    W1  = 1+RAD_FREQ*RAD_FREQ*Rd*Rd*cap*cap;
    Wr  =  Rd/W1;
    Wi  =  RAD_FREQ*cap*Rd*Rd/W1;
    ac_gain.real = Wr;
    ac_gain.imag = -1.0*Wi;
    AC_GAIN(dnlcap, dnlcap) = ac_gain;
  }
}
```

Semiconductor diode non-linear capacitance characteristics, including depletion and diffusion components

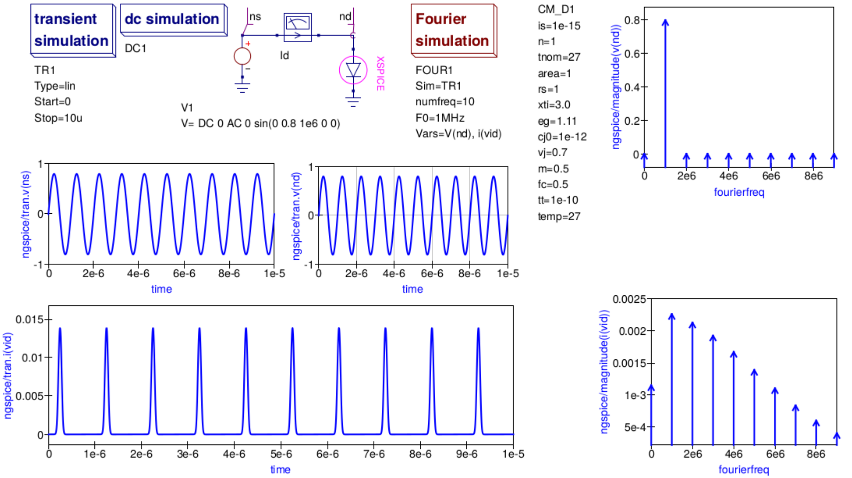
Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (d) The diode small signal AC performance; Y parameter, Rd and Cd extraction



Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (e) The diode Id/Vd temperature variation



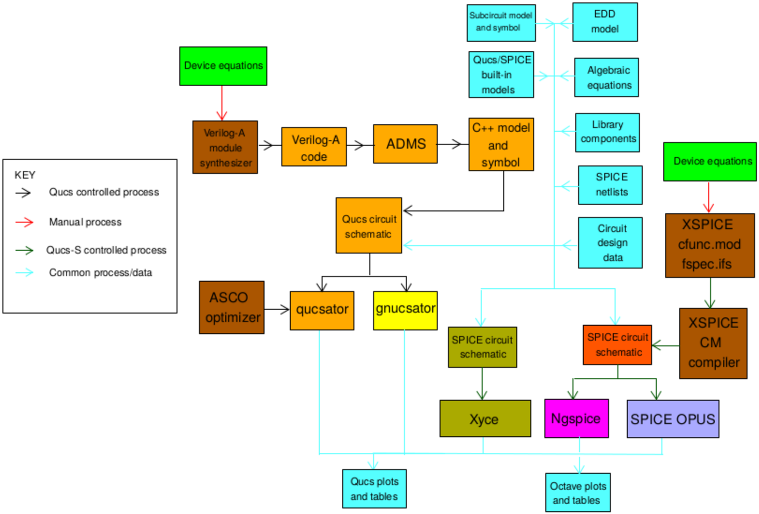
Qucs-S development after release 0.0.19-S: the way forward to Qucs-0.0.20; Part 6 XSPICE diode model - (f) Diode transient response



Qucs Structure after merging Qucs-0.0.19 and Qucs-0.0.19-S ? - Wish list

- **How and when Qucs-0.0.19 and Qucs-0.0.19-S are merged is not decided yet! Indeed they may never merge but continue to function as two separate packages side-by-side**!
- GUI and simulator:
  - Refactor/rewrite, (Qt4) Qt5, plug-ins, API...; Standard file formats, exchangeable
- Powerful circuit analysis tools:
  - Robust algorithms (Eigen, KLU); API, high level interface (SWIG);improved Harmonic-Balance
  - EM field simulation / extraction (openEMS, NEC2++); improved SPICE flavors compatibility/converter
  - Co-simulation (analogue + Verilog/VHDL), interface (icarus, GHDL); Monte-Carlo simulation
  - Solvers: Ngspice, Xyce, Gnucap, SpiceOpus
- Design and synthesis tools:
  - Data import / export
- Industry standard device models:
  - MEXTRAM, VBIC, HiSIM, IGBJT, UTSOI, ...
- Hardware implementation:
  - Output layout data for input to PCB and IC packages, for example KiCad and Klayout

Integrated Qucs and Qucs-S compact modelling capabilities



Summary

Qucs 0.0.19 is a major release of the circuit simulator package with the extended features introduced in this presentation included. Qucs 0.0.19 has benefited from the significant amount of work done by the Qucs 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. As the Qucs Development Team moves on to release 0.0.20 it is difficult to say what the structure of the next release will be. However, whatever the final decision is concerning merging Qucs-0.0.19 and Qucs-0.0.19-S the Qucs project will continue to provide a freely available modern full featured circuit simulator under GPL.

Stable and development versions of Qucs-0.0.19 and Qucs-0.0.19-S can be downloaded from:

1. **Qucs-0.0.19**: http://qucs.sourceforge.net/ and https://github.com/Qucs/qucs/
2. **Qucs-0.0.19-S**: https://github.com/ra3xdh/qucs/releases/tag/0.0.19S-rc4