Qucs Equation-Defined Device modelling with a Verilog-A Prototyping Platform

Mike Brinson [1], mbrin72043@yahoo.co.uk.
Vadim Kuznetsov [2], ra3xdh@gmail.com
Wladek Grabinski [3], wladek@mos-ak.org

[1] **Centre for Communications Technology, London Metropolitan University, UK**

[2] **Bauman Moscow Technical University, Russia**

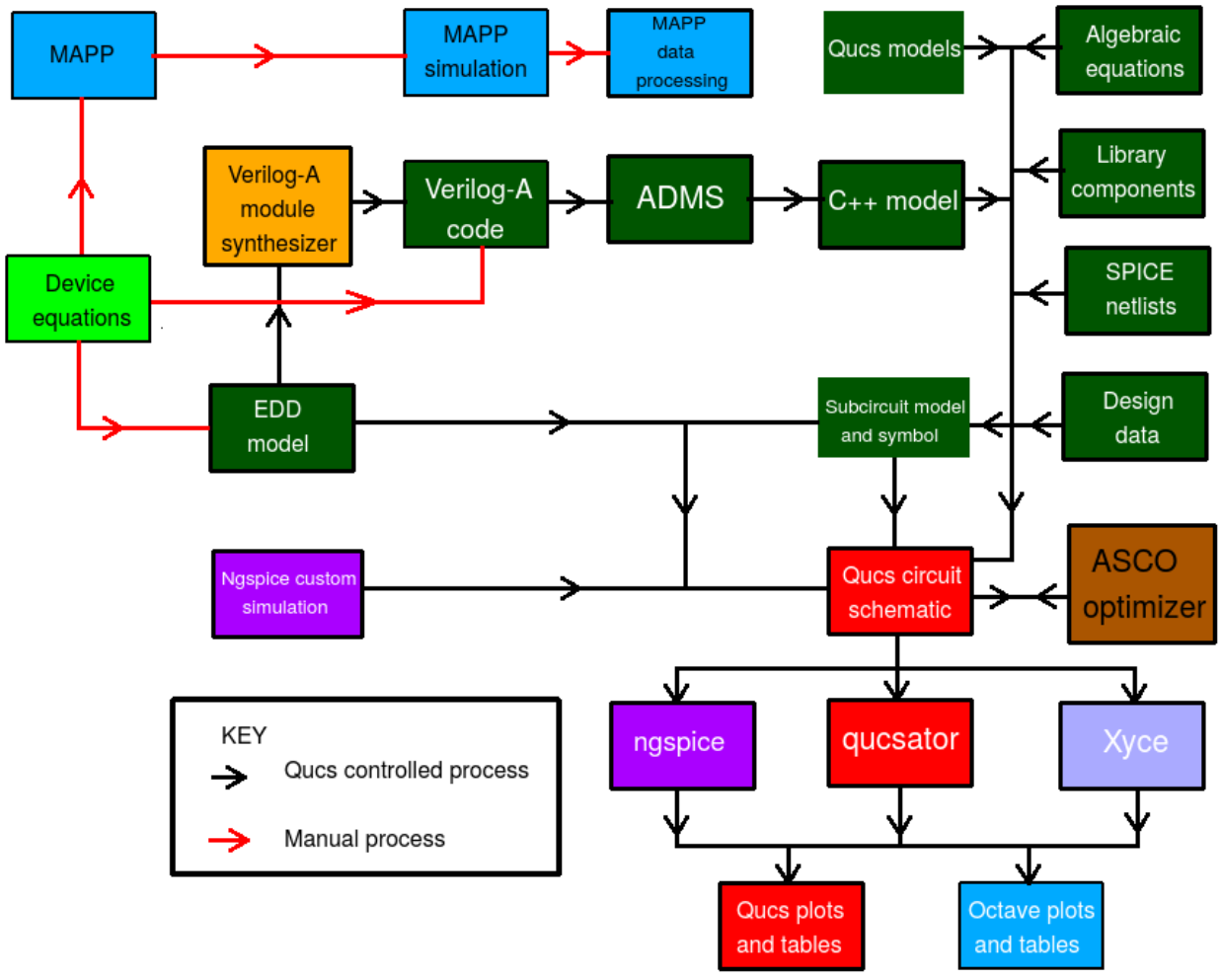
[3] **MOS-AK (EU)**

Presented at the 8th International MOS-AK Workshop, Washington DC, December 9 2015

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
- Compact modelling with Qucs, ngspice, and Xyce
  - EDD support: Current and charge equations
  - B-type SPICE sources
  - Harmonic balance simulation with Xyce and Qucs compact models
- Parametrization features and ngnutmeg scripting introduced with spice4qucs
- 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: – features and current Qucs-0.0.19S version

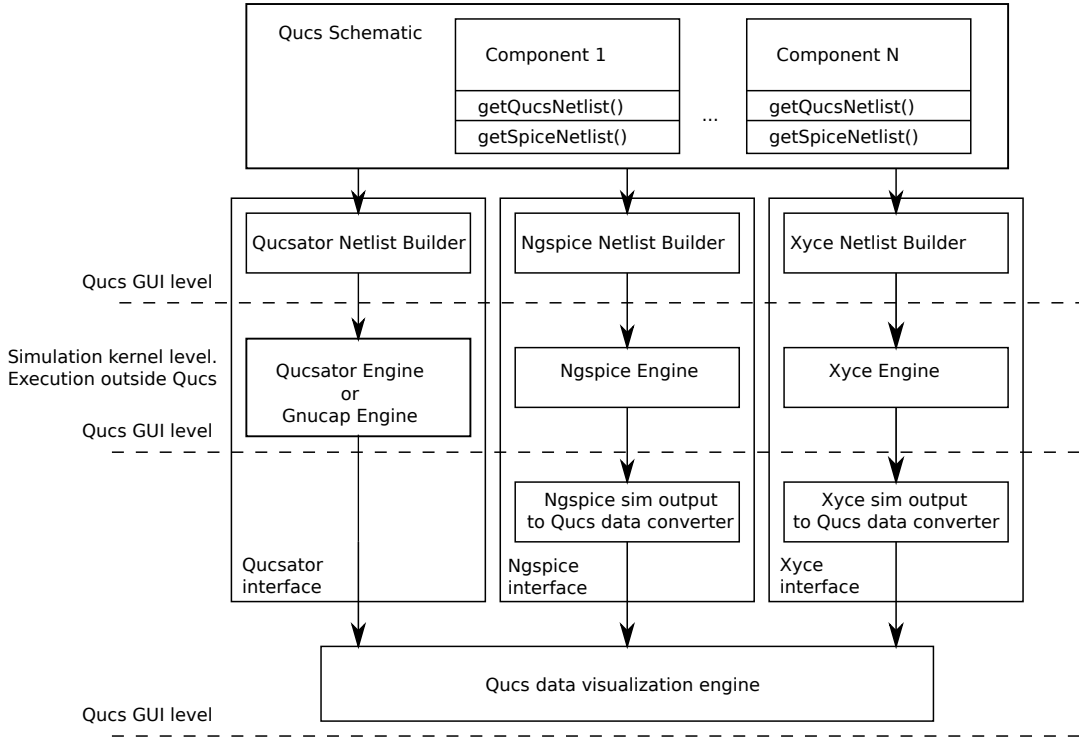
Spice4qucs features:

- 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

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

- Qucs<—>Ngspice/Xyce interfacing schematic



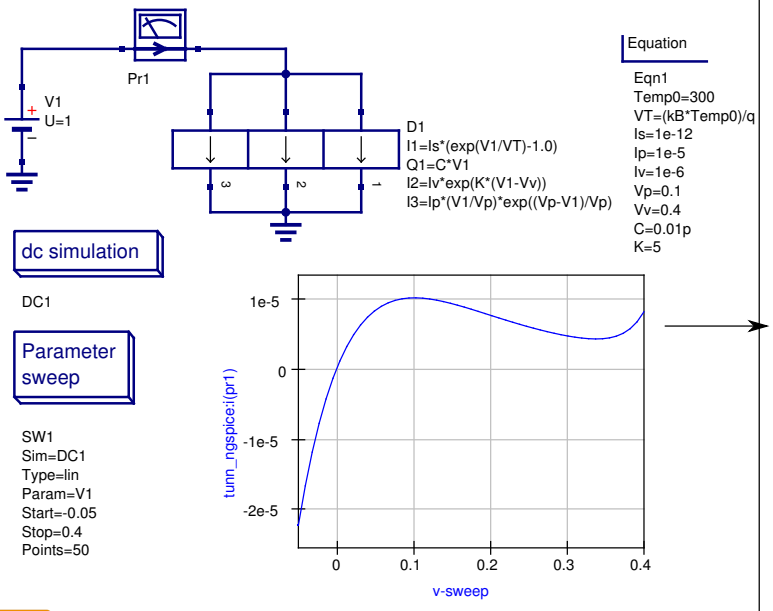
- Spice4qucs online documentation available here:
  `https://qucs-help.readthedocs.org/en/spice4qucs/index.html`

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

Consider tunnel diode model represented by

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

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



Spice Netlist

```
* Qucs 0.0.19  /home/vvk/.qucs/tunn.sch
.PARAM Temp0=300
.PARAM VT={(1.38065e-23*Temp0)/1.6021765e-19}
.PARAM Is=1e-12
.PARAM Ip=1e-5
.PARAM Iv=1e-6
.PARAM Vp=0.1
.PARAM Vv=0.4
.PARAM C={0.01p}
.PARAM K=5
VPr1 _net0 _net1 DC 0 AC 0
* EDD begin
BD1I0 _net1 0 I=Is*(exp((V(_net1)-V(0))/VT)-1.0)
GD1Q0 _net1 0 nD1Q0 0 1.0
LD1Q0 nD1Q0 0 1.0
BD1Q0 nD1Q0 0 I=-(C*(V(_net1)-V(0)))
BD1I1 _net1 0 I=Iv*exp(K*((V(_net1)-V(0))-Vv))
BD1I2 _net1 0 I=Ip*((V(_net1)-V(0))/Vp)*exp((Vp-(V(_net1)-V(0)))/Vp)
* EDD end
V1 _net0 0 DC 1
.control
set filetype=ascii
DC V1 -0.05 0.4 0.009
write tunn_dc.txt VPr1#branch
destroy all
reset

exit
.endc
.END
```

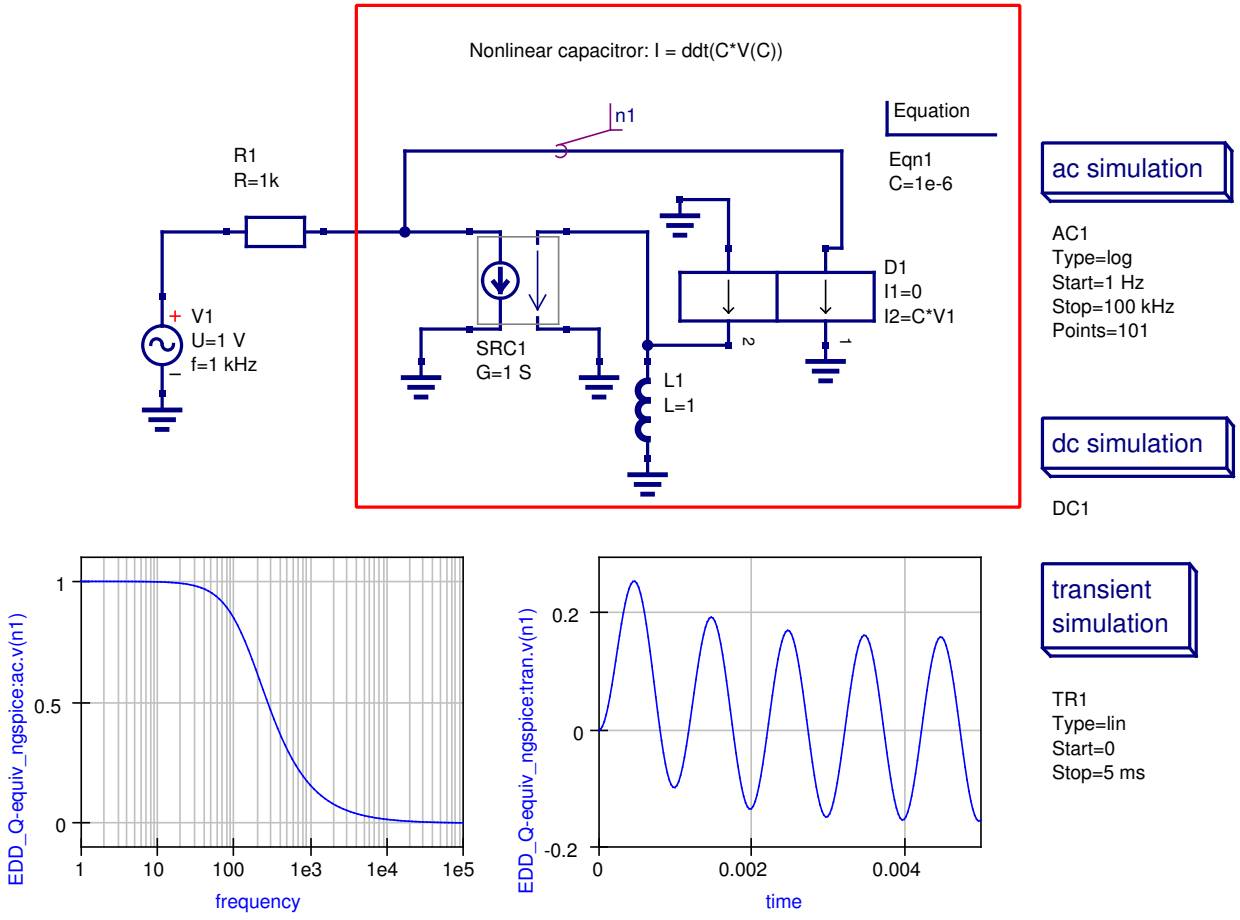
Compact modelling with Qucs and ngspice/Xyce: Part II – Charge equation approach

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:

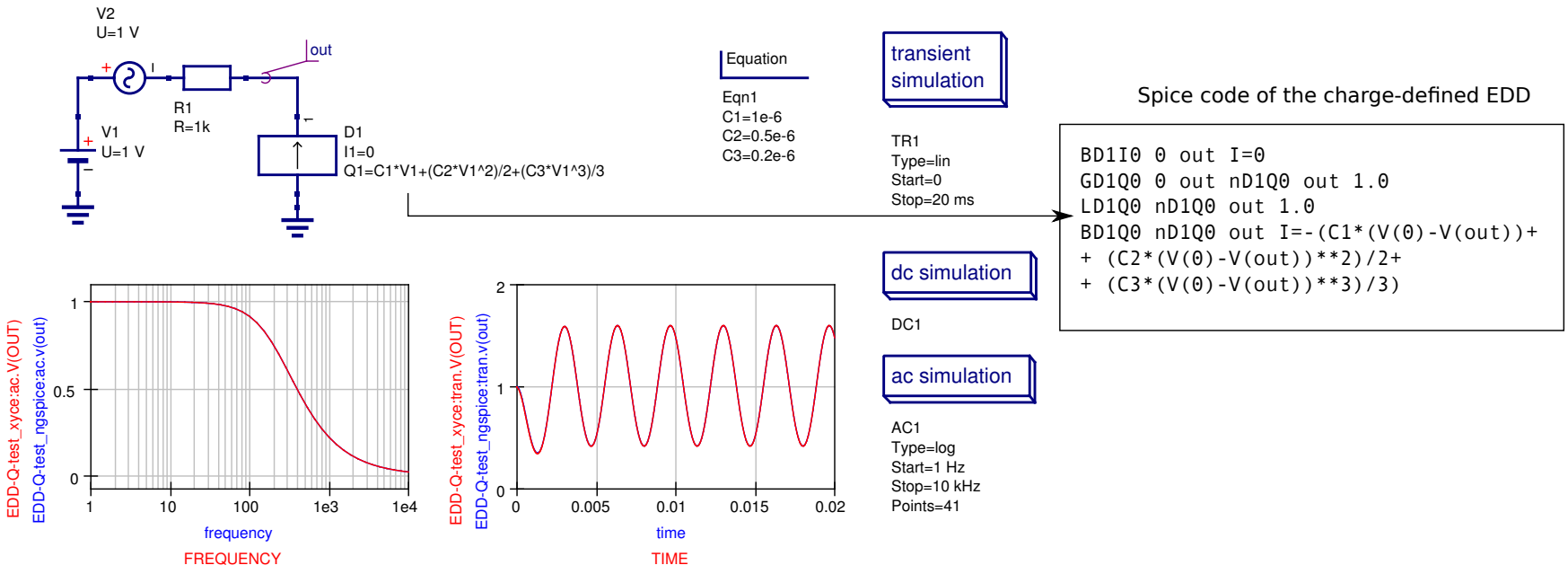
- Nonlinear capacitance equivalent circuit:



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

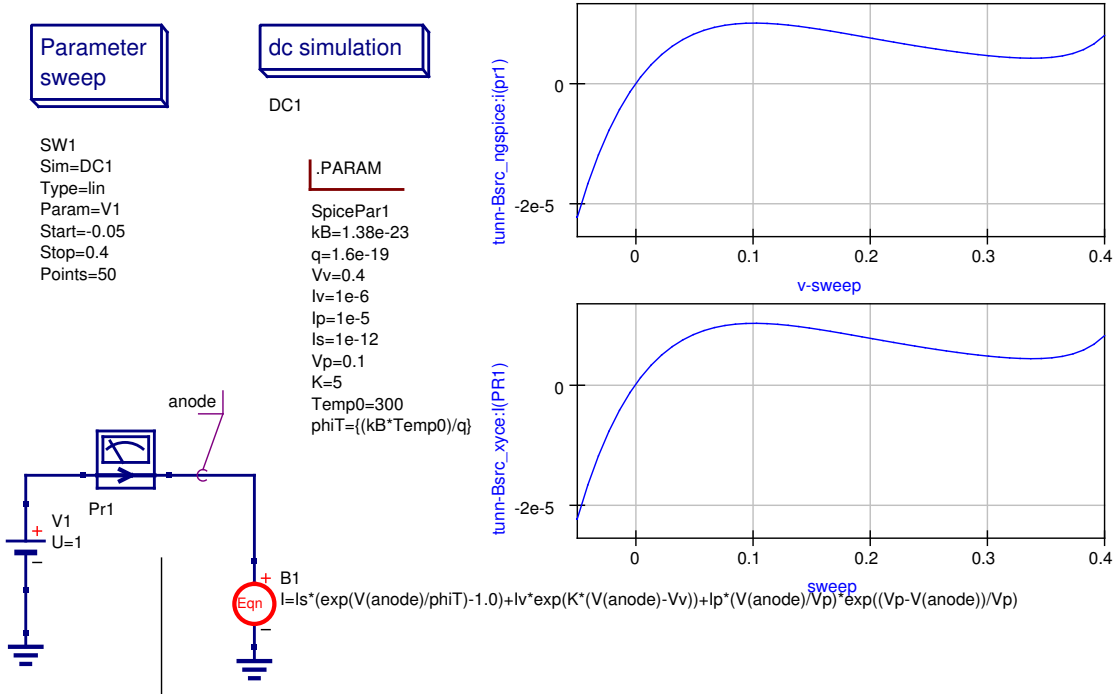
- In this example a nonlinear capacitance is simulated with ngspice and Xyce:

$$Q = C_1 V + \frac{C_2 V^2}{2} + \frac{C_3 V^3}{3} + \ldots + \frac{C_N V^N}{N} \tag{3}$$



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:

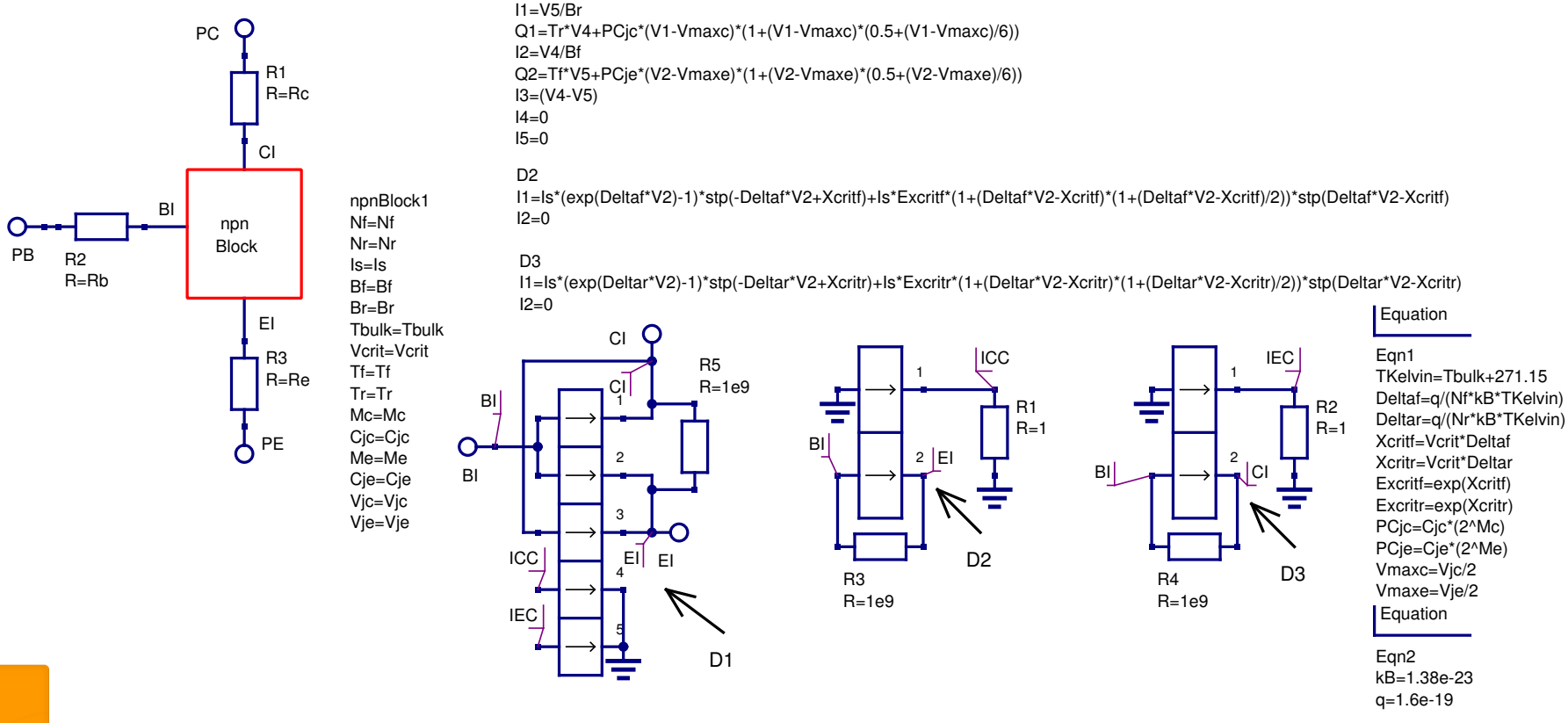


Auto-generated SPICE netlist

```
* Qucs 0.0.19  /home/vvk/.qucs/tunn-Bsrc.sch
.PARAM kB = 1.38e-23
.PARAM q = 1.6e-19
.PARAM Vv = 0.4
.PARAM Iv = 1e-6
.PARAM Ip = 1e-5
.PARAM Is = 1e-12
.PARAM Vp = 0.1
.PARAM K = 5
.PARAM Temp0 = 300
.PARAM phiT = {(kB*Temp0)/q}
VPr1 _net0 anode DC 0 AC 0
V1 _net0 0 DC 1
B1 anode 0 I = Is*(exp(V(anode)/phiT)-1.0)+
+ Iv*exp(K*(V(anode)-Vv))+
+ Ip*(V(anode)/Vp)*exp((Vp-V(anode))/Vp)
.control
set filetype=ascii
echo "" > spice4qucs.cir.noise
DC V1 -0.05 0.4 0.009
write tunn-Bsrc_dc.txt VPr1#branch v(anode)
destroy all
reset

exit
.endc
.END
```

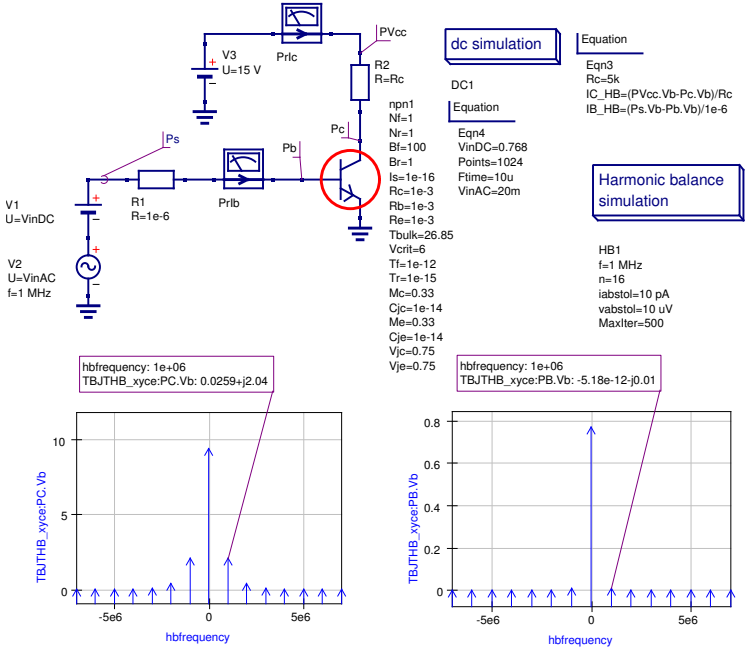
Compact modeling with Qucs and ngspice/Xyce: Part V – NPN BJT compact model used for Harmonic balance analysis of a one-stage BJT amplifier

- 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 VI – HB analysis SPICE netlist and output data for a BJT amplifier

- Xyce harmonic balance simulation data and auto generated netlist for one-stage BJT amplifier; see `http://www.mixdes.org/Mixdes3/`:

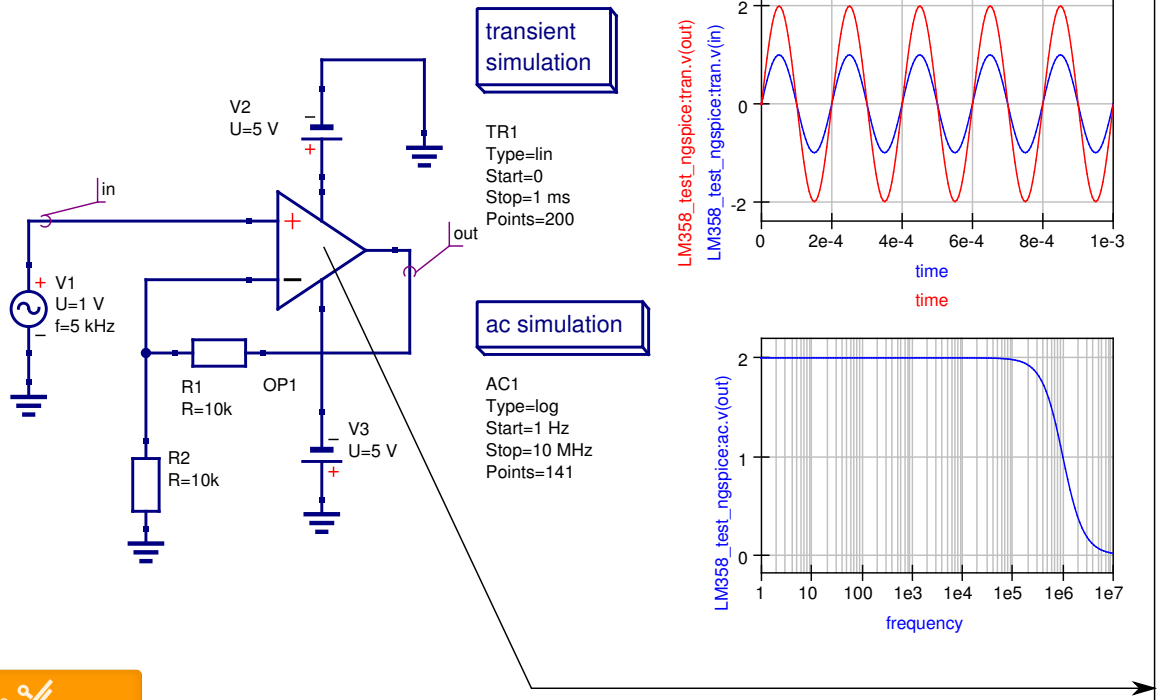


Xyce netlist

```
* Qucs 0.0.19  /home/vvk/.qucs/MIXDESBJTXyce_prj/TBJTHB.sch
* Qucs 0.0.19  npnBlock.sch
.SUBCKT npnBlock CI BI EI Nf=1 Nr=1 Is=1e-14 Bf=100
+ Br=1 Tbulk=26.85 Vcrit=6 Tf=1e-12 Tr=1e-15 Mc=0.33
+ Cjc=1e-14 Me=0.33 Cje=1e-14 Vjc=0.75 Vje=0.75
.PARAM kB=1.38e-23
.PARAM q=1.6e-19
.PARAM TKelvin={Tbulk+271.15}
.PARAM Deltaf={1.6021765e-19/(Nf*1.38065e-23*TKelvin)}
.PARAM Deltar={1.6021765e-19/(Nr*1.38065e-23*TKelvin)}
.PARAM Xcritf={Vcrit*Deltaf}
.PARAM Xcritr={Vcrit*Deltar}
.PARAM Excritf={exp(Xcritf)}
.PARAM Excritr={exp(Xcritr)}
.PARAM PCjc={Cjc*(2**Mc)}
.PARAM PCje={Cje*(2**Me)}
.PARAM Vmaxc={Vjc/2}
.PARAM Vmaxe={Vje/2}
R5 EI CI  1E9
R2 0 IEC  1
R4 BI CI  1E9
R1 0 ICC  1
R3 BI EI  1E9
BD3I0 0 IEC I=Is*(exp(Deltar*(V(BI)-V(CI)))-1)*stp(-Deltar*(V(BI)-V(CI))+Xcritr)+
+ Is*Excritr*(1+(Deltar*(V(BI)-V(CI))-Xcritr)*(1+(Deltar*(V(BI)-V(CI))-Xcritr)/2))*
+ stp(Deltar*(V(BI)-V(CI))-Xcritr)
BD3I1 BI CI I=0
BD2I0 0 ICC I=Is*(exp(Deltaf*(V(BI)-V(EI)))-1)*stp(-Deltaf*(V(BI)-V(EI))+Xcritf)+
+ Is*Excritf*(1+(Deltaf*(V(BI)-V(EI))-Xcritf)*(1+(Deltaf*(V(BI)-V(EI))-Xcritf)/2))*
+ stp(Deltaf*(V(BI)-V(EI))-Xcritf)
BD2I1 BI EI I=0
BD1I0 BI CI I=(V(IEC)-V(0))/Br
GD1Q0 BI CI nD1Q0 CI 1.0
LD1Q0 nD1Q0 CI 1.0
BD1Q0 nD1Q0 CI I=-(Tr*(V(ICC)-V(0))+PCjc*((V(BI)-V(CI))-Vmaxc)*
+ (1+((V(BI)-V(CI))-Vmaxc)*(0.5+((V(BI)-V(CI))-Vmaxc)/6)))
BD1I1 BI EI I=(V(ICC)-V(0))/Bf
GD1Q1 BI EI nD1Q1 EI 1.0
LD1Q1 nD1Q1 EI 1.0
BD1Q1 nD1Q1 EI I=-(Tf*(V(IEC)-V(0))+PCje*((V(BI)-V(EI))-Vmaxe)*
+ (1+((V(BI)-V(EI))-Vmaxe)*(0.5+((V(BI)-V(EI))-Vmaxe)/6)))
BD1I2 CI EI I=((V(ICC)-V(0))-(V(IEC)-V(0)))
BD1I3 ICC 0 I=0
BD1I4 IEC 0 I=0
.ENDS
* Qucs 0.0.19  npnBJT.sch
.SUBCKT npnBJT _net5 _net2 _net0 Nf=1 Nr=1 Bf=100 Br=1 Is=1e-14
+ Rc=0.1 Rb=1 Re=0.1 Tbulk=26.85 Vcrit=6 Tf=1e-12 Tr=1e-15
+ Mc=0.33 Cjc=1e-14 Me=0.33 Cje=1e-14 Vjc=0.75 Vje=0.75
R3 _net0 _net1  {RE}
R2 _net2 _net3  {RB}
R1 _net4 _net5  {RC}
XnpnBlock1  _net4 _net3 _net1 npnBlock Nf={NF} Nr={NR} Is={IS}
+ Bf={BF} Br={BR} Tbulk={TBULK} Vcrit={VCRIT} Tf={TF} Tr={TR}
+ Mc={MC} Cjc={CJC} Me={ME} Cje={CJE} Vjc={VJC} Vje={VJE}
.ENDS
.PARAM Rc={5k}
.PARAM VinDC=0.768
.PARAM Points=1024
.PARAM Ftime={10u}
.PARAM VinAC={20m}
R2 Pc PVcc  {RC}
Xnpn1  Pc Pb 0 npnBJT Nf=1 Nr=1 Bf=100 Br=1 Is=1E-16 Rc=1E-3 Rb=1E-3
+ Re=1E-3 Tbulk=26.85 Vcrit=6 Tf=1E-12 Tr=1E-15 Mc=0.33 Cjc=1E-14
+ Me=0.33 Cje=1E-14 Vjc=0.75 Vje=0.75
V1 Ps _net0 DC {VINDC}
V2 _net0 0 DC 0 SIN(0 {VINAC} 1MEG 0 0) AC {VINAC}
R1 Ps _net1  1E-6
V3 _net2 0 DC 15
VPrIc _net2 PVcc DC 0 AC 0
VPrIb _net1 Pb DC 0 AC 0
.options hbint numfreq=16 STARTUPPERIODS=2
.HB 1MEG
.PRINT  hb file=TBJTHB_hb.txt I(VPrIb) I(VPrIc) v(PVcc) v(Pb) v(Pc) v(Ps)
.END
```

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):



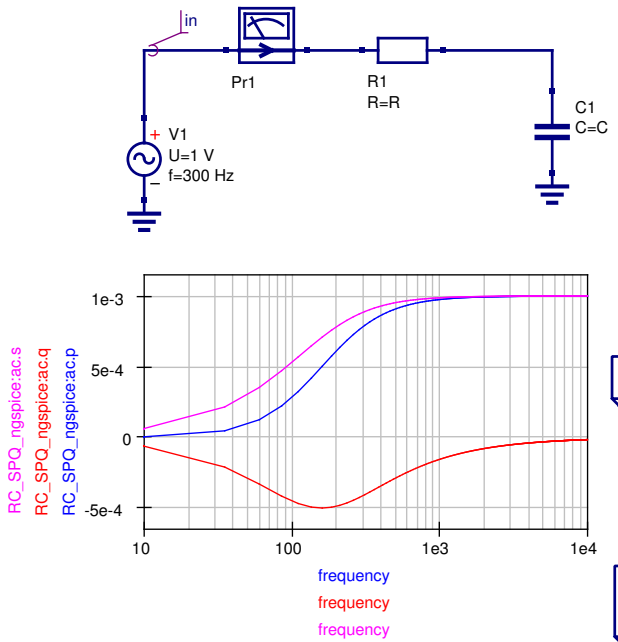
LM358 XSPICE macromodel

```
.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
HLIM 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
RO1   8  5 50
RO2   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(IS=800.0E-18)
.MODEL QX PNP(IS=800.0E-18 BF=250)
.ENDS
```

Qucs equation support in spice4qucs

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

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



Equation

Eqn2
R=1k
C=1u

Equation

Eqn1
S=in.v*(real(Pr1.i)-i*imag(Pr1.i))
P=real(S)
Q=imag(in.v*(real(Pr1.i)-i*imag(Pr1.i)))

ac simulation

AC1
Type=lin
Start=10 Hz
Stop=10 kHz
Points=400

transient simulation

TR1
Type=lin
Start=0
Stop=10 ms

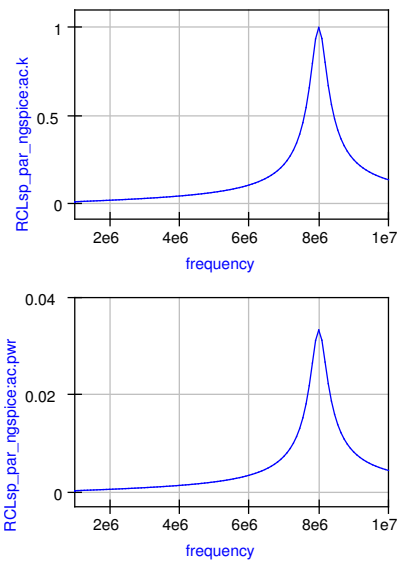
Auto-generated Ngspice netlist

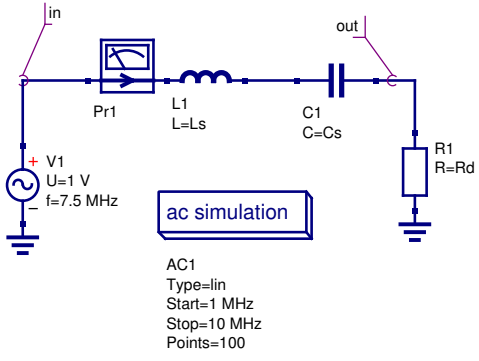
```
* Qucs 0.0.19  /home/vvk/.qucs/RC_SPQ.sch
.PARAM R={1k}
.PARAM C={1u}
VPr1 in _net1 DC 0 AC 0
V1 in 0 DC 0 SIN(0 1 300 0 0) AC 1
C1 0 _net0  {C}
R1 _net1 _net0  {R}
.control
set filetype=ascii
echo "" > spice4qucs.cir.noise
let R=1k
let C=1u
AC LIN 400 10 10K
let S=V(in)*(real(VPr1#branch)-i*imag(VPr1#branch))
let P=real(S)
let Q=imag(V(in)*(real(VPr1#branch)-i*imag(VPr1#branch)))
write RC_SPQ_ac.txt VPr1#branch v(in)  S P Q
destroy all
reset

TRAN 3.33333e-05 0.01 0
write RC_SPQ_tran.txt VPr1#branch v(in)
destroy all
reset

exit
.endc
.END
```

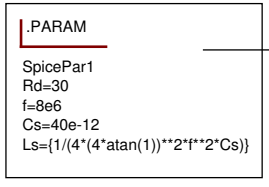
SPICE style parametrization and ngnutmeg postprocessor usage implemented by spice4qucs

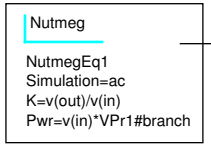
The following Qucs "equation" style icons introduce model parametrization and simulation data postprocessing:

- SPICE .PARAM section icon
- ngnutmeg equation icon





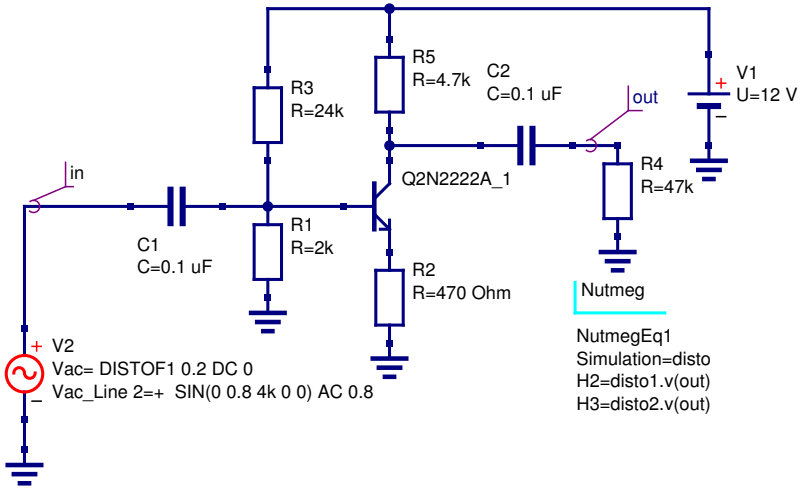




Auto-generated Ngspice netlist

```
* Qucs 0.0.19  /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 0 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



transient simulation

TR1
Type=lin
Start=0
Stop=1 ms

Fourier simulation

FOUR1
Sim=TR1
numfreq=20
F0=4kHz
Vars=V(out)

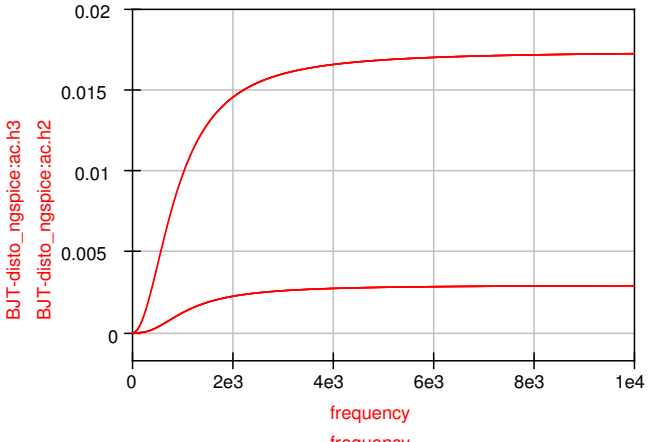
Noise simulation

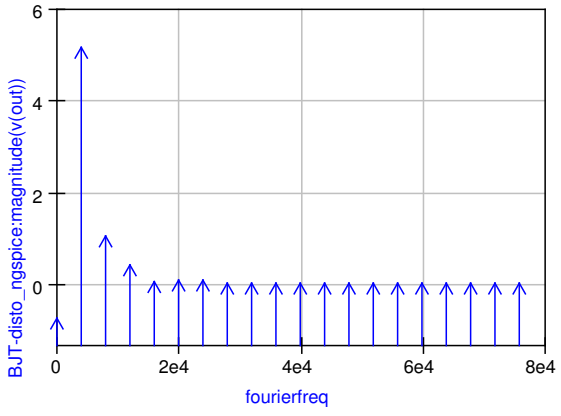
NOISE1
Type=lin
Start=1 Hz
Stop=10 MHz
Points=100
Output=v(out)
Source=V2

Distortion simulation

DISTO1
Type=lin
Start=1 Hz
Stop=10 kHz
Points=1000

number	BJT-disto_ngspice:inoise_total	BJT-disto_ngspice:onoise_total
1	1.48e-10	3.91e-09



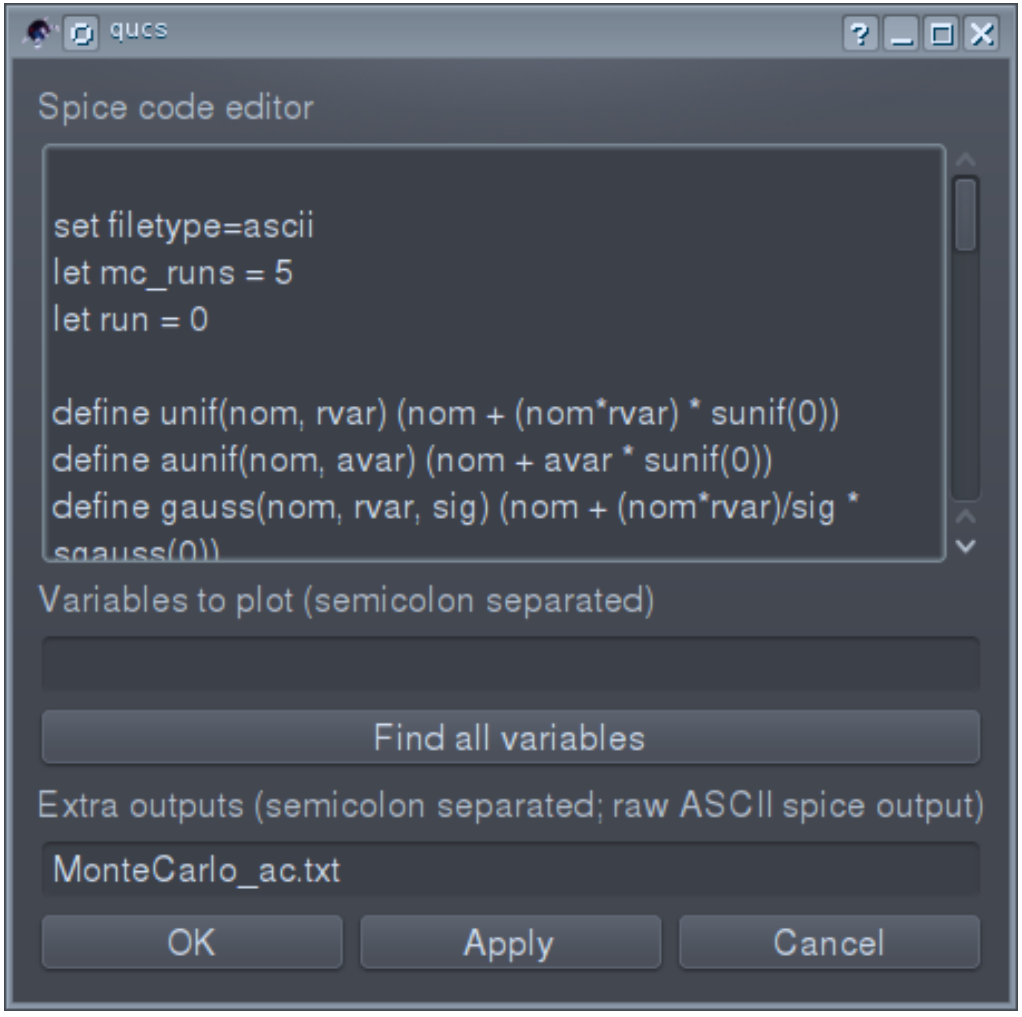


Ngspice custom simulation techniques: Part I – Main features

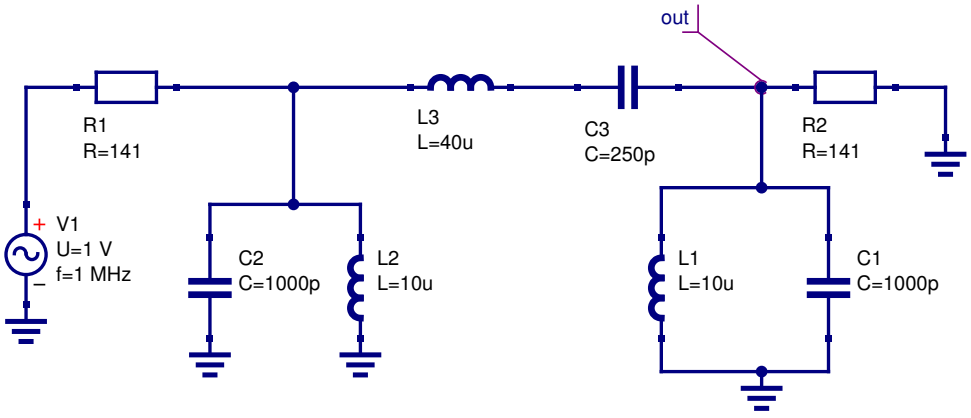
Main features:

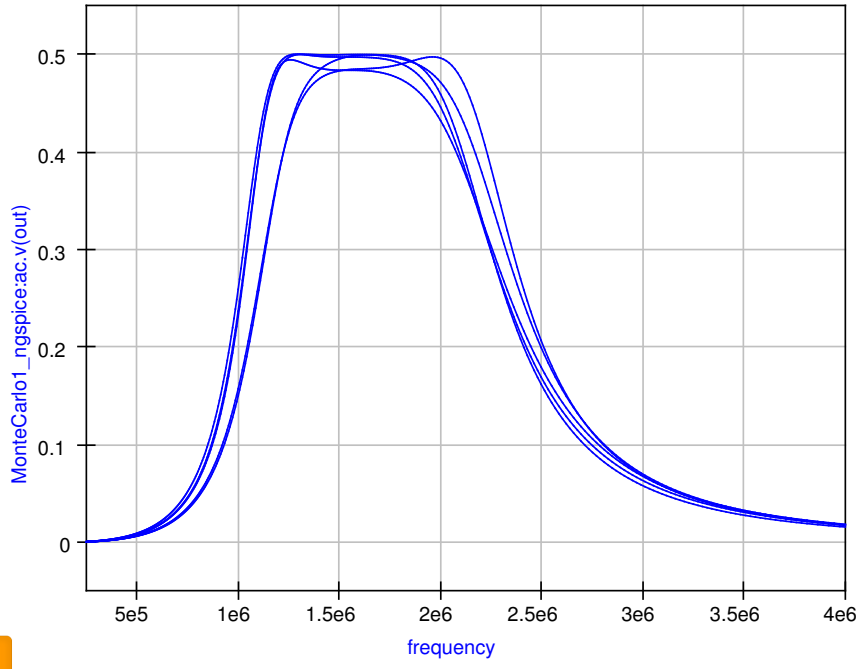
- Embedding user defined ngnutmeg scripts in a Qucs schematic
- Full ngnutmeg operator and function support
- User defined variables for plotting simulation data
- User defined raw ASCII SPICE3f5 style output

- Ngnutmeg script editing dialogue:



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





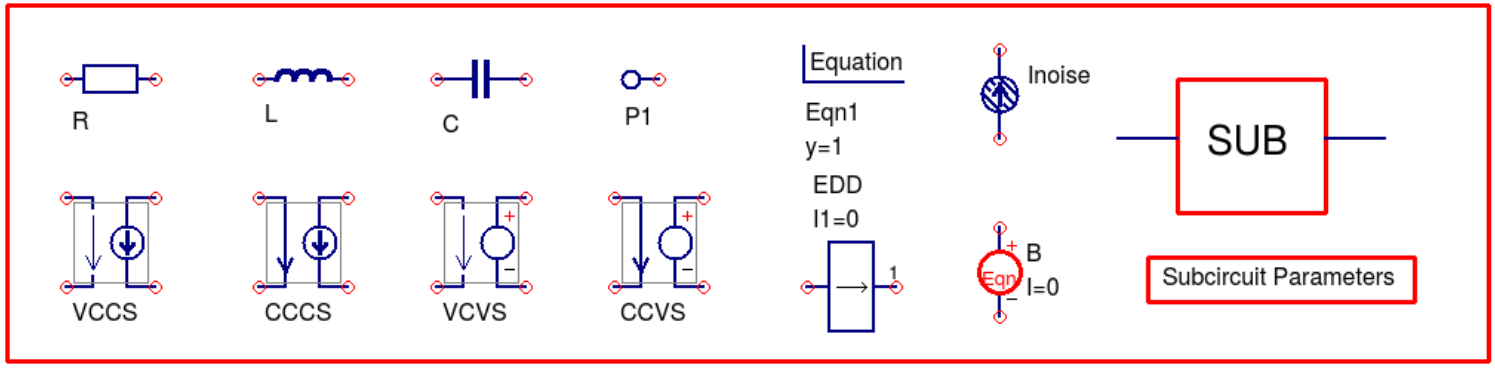
Ngspice
custom simulation

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

define unif(nom, rvar) (nom + (nom*rvar) * sunif(0))
define aunif(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 l1 = unif(10e-06, 0.1)
alter c2 = unif(1e-09, 0.1)
alter l2 = unif(10e-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
```

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

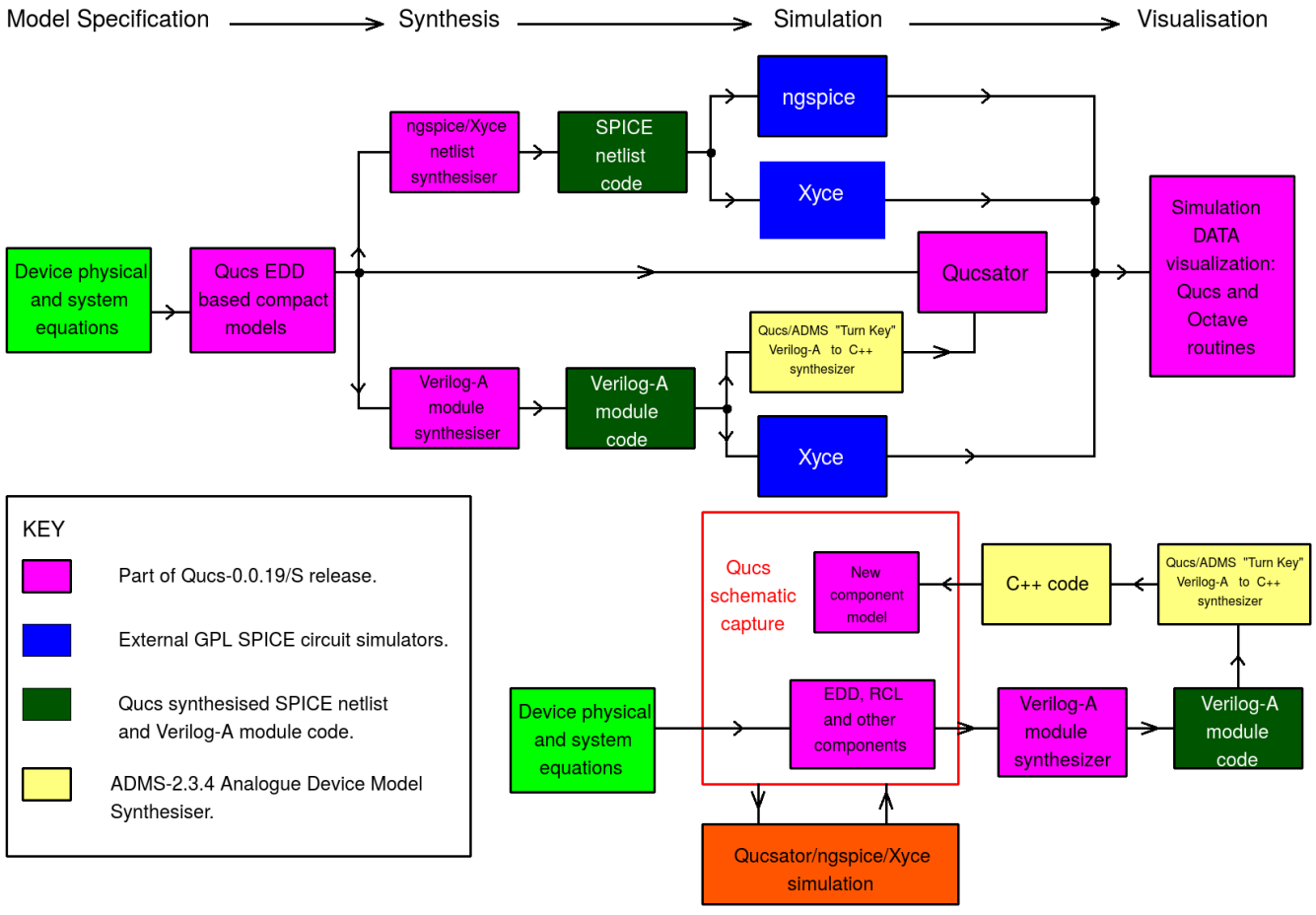
Qucs-0.0.19S includes the first release of a GPL Verilog-A synthesis tool for compact device modelling.

- 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:



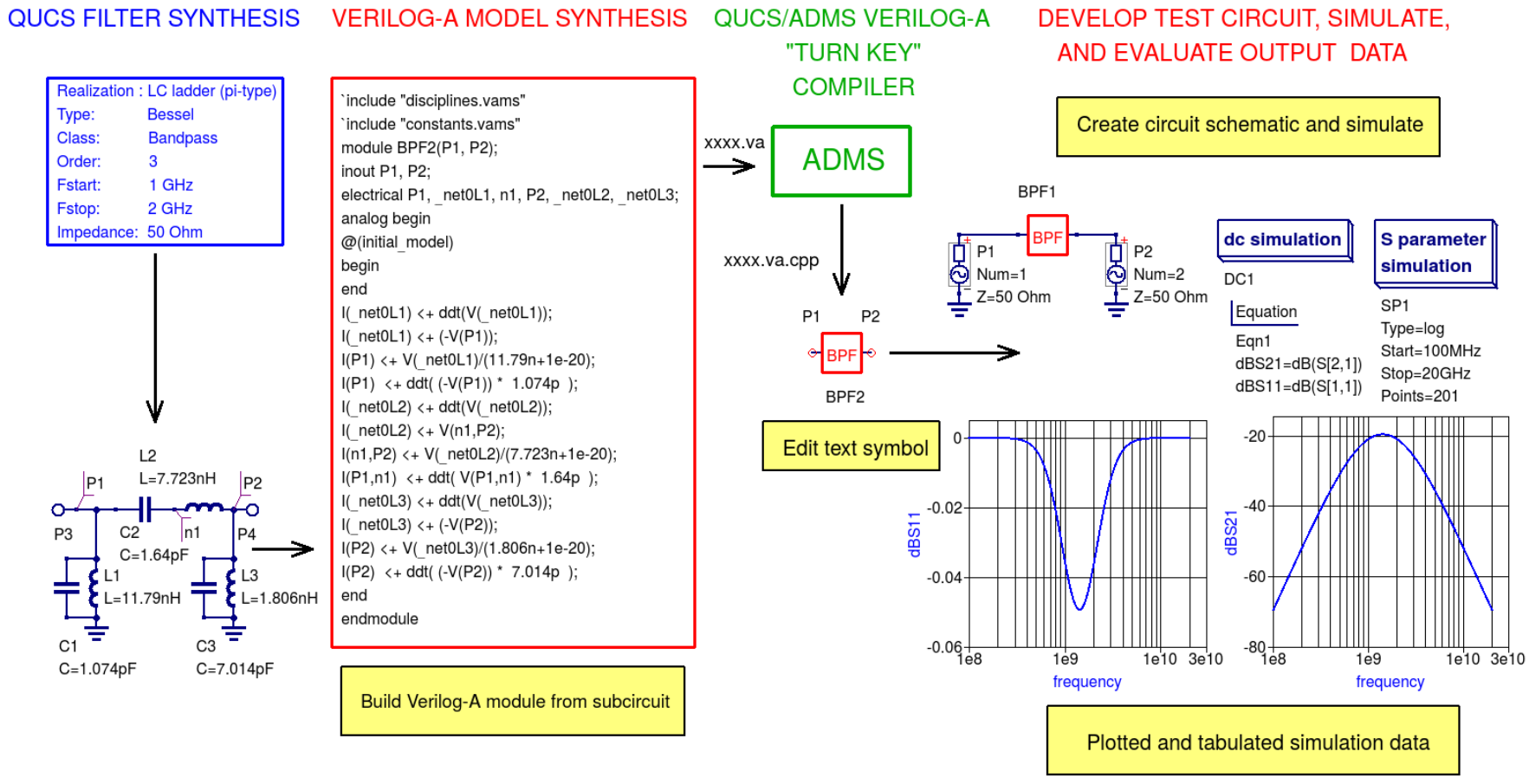
Introduction to the Qucs GPL Verilog-A module synthesizer: Part II

Structure:



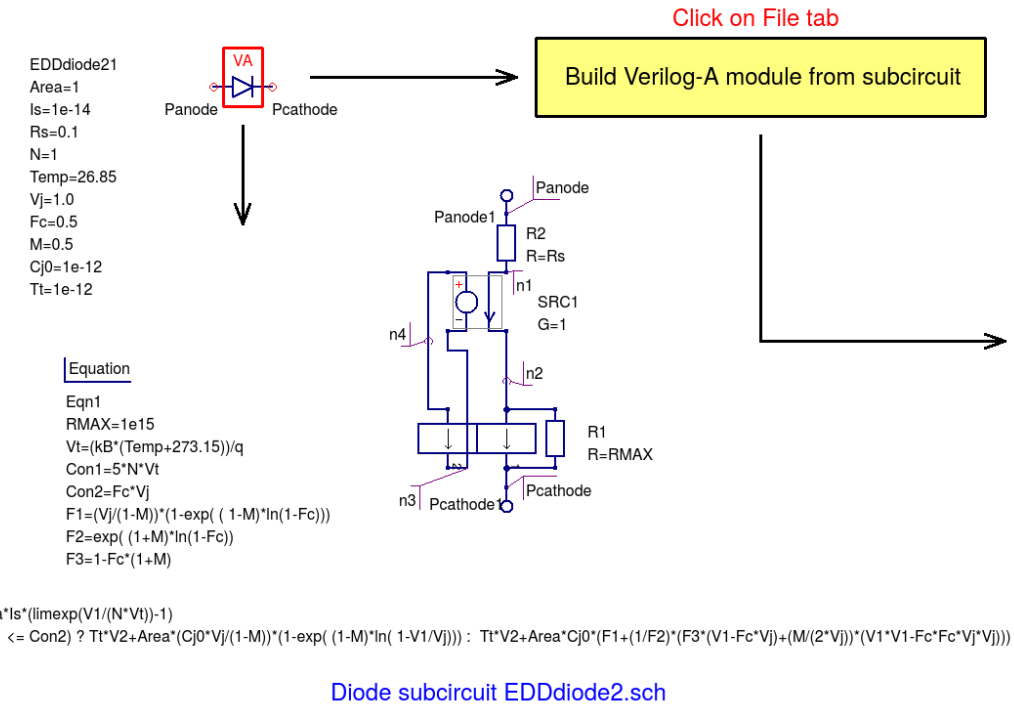
Introduction to the Qucs GPL Verilog-A module synthesizer: Part III

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



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

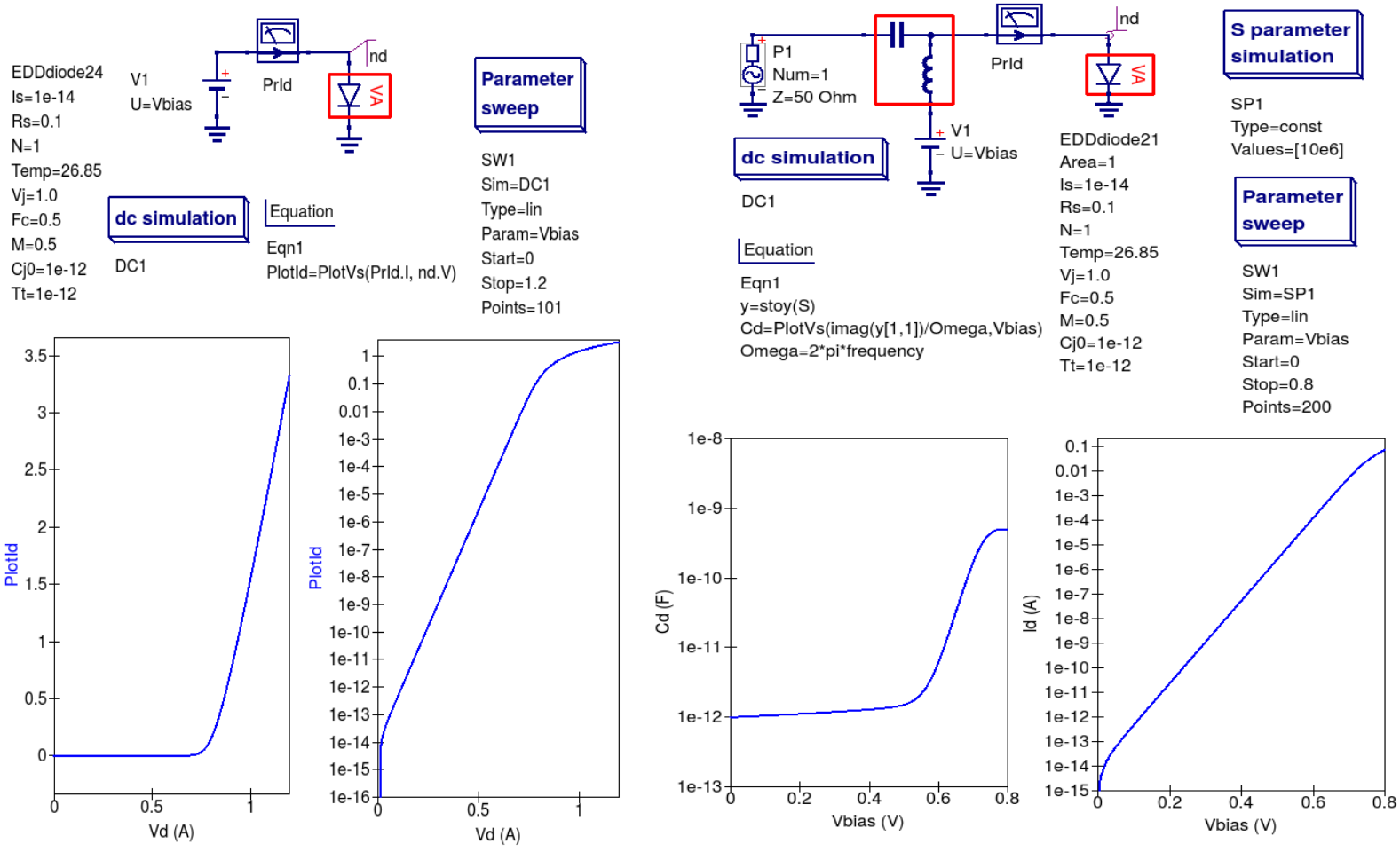
Synthesis of a SPICE like compact semiconductor diode model: static I_d and dynamic capacitance model plus synthesized Verilog-A module code.




```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode2(Panode, Pcathode);
inout Panode, Pcathode;
electrical Pcathode, n2, n1, Panode, n4, n3;
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 Vj=1.0;
parameter real Fc=0.5;          parameter real M=0.5;
parameter real Cj0=1e-12;       parameter real Tt=1e-12;
real RMAX, Vt, Con1, Con2, F1, F2, F3;
analog begin
@(initial_model)
begin
RMAX=1e15;     Vt=(`P_K*(Temp+273.15))/`P_Q;
Con1=5*N*Vt;   Con2=Fc*Vj;
F1=(Vj/(1-M))*(1-exp((1-M)*ln(1-Fc)));    F2=exp((1+M)*ln(1-Fc));
F3=1-Fc*(1+M);
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2) <+ V(n1,n2) * 1e3; I(n3,n4) <+ I(n3,n4) *1e-3;  I(n3,n4) <+ V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ Area*Is*(limexp(V(n2,Pcathode)/(N*Vt))-1);
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2) ? Tt*V(n4,n3)+Area*(Cj0*Vj/(1-M))*
                  (1-exp((1-M)*ln(1-V(n2,Pcathode)/Vj)))
                 : Tt*V(n4,n3)+Area*Cj0*(F1+(1/F2)*(F3*
                   V(n2,Pcathode)-Fc*Vj)+(M/(2*Vj))*
                   V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*Vj*Vj))) );
end
endmodule
```

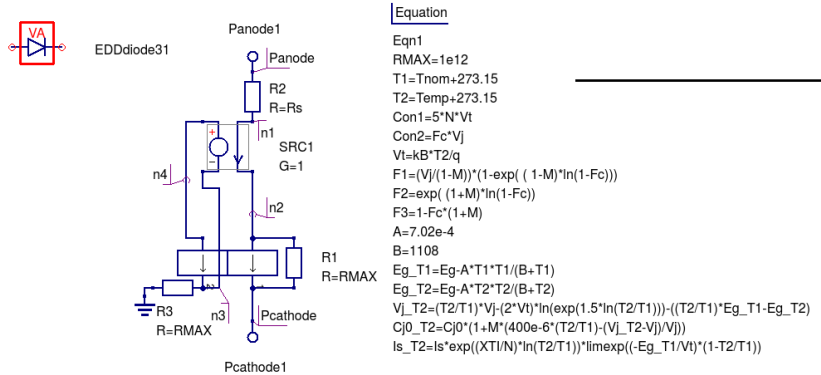
Introduction to the Qucs GPL Verilog-A module synthesizer: Part V

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



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

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



Equation

Eqn1
RMAX=1e12
T1=Tnom+273.15
T2=Temp+273.15
Con1=5*N*Vt
Con2=Fc*Vj
Vt=kB*T2/q
F1=(Vj/(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_T1=Eg-A*T1*T1/(B+T1)
Eg_T2=Eg-A*T2*T2/(B+T2)
Vj_T2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*Eg_T1-Eg_T2)
Cj0_T2=Cj0*(1+M*(400e-6*(T2/T1)-(Vj_T2-Vj)/Vj))
Is_T2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-Eg_T1/Vt)*(1-T2/T1))

D1
I1=(V1 > -Con1) ? Area*Is_T2*(limexp(V1/(N*Vt))-1) : -Is_T2
Q1= (V1 <= Con2) ? Tt*V2+Area*(Cj0_T2*Vj_T2/(1-M))*(1-exp((1-M)*ln(1-V1/Vj_T2))) : Tt*V2+Area*Cj0_T2*(F1+(1/F2)*(F3*(V1-Fc*Vj_T2)+(M/(2*Vj_T2))*(V1*V1-Fc*Fc*Vj_T2*Vj_T2)))
I2=0
Q2=0

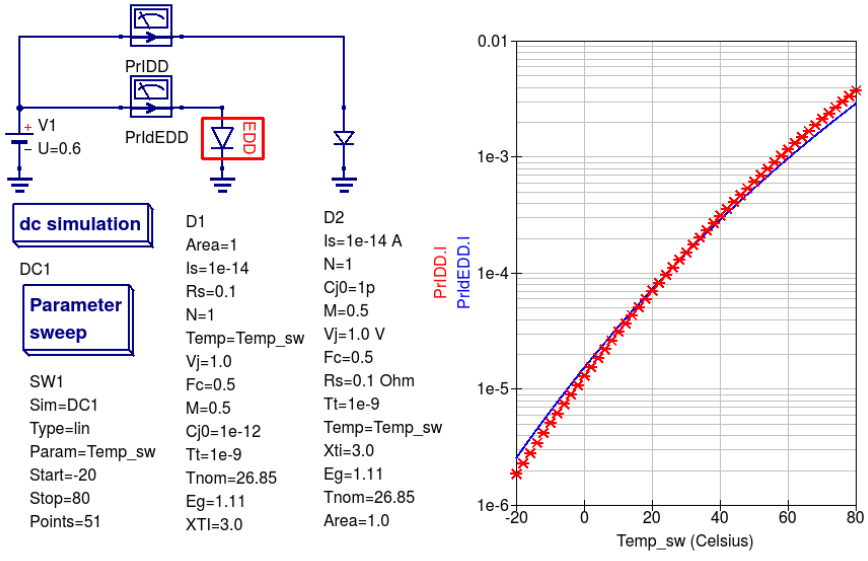
Qucs EDD diode model with temperature effects

Synthesized Verilog-A code

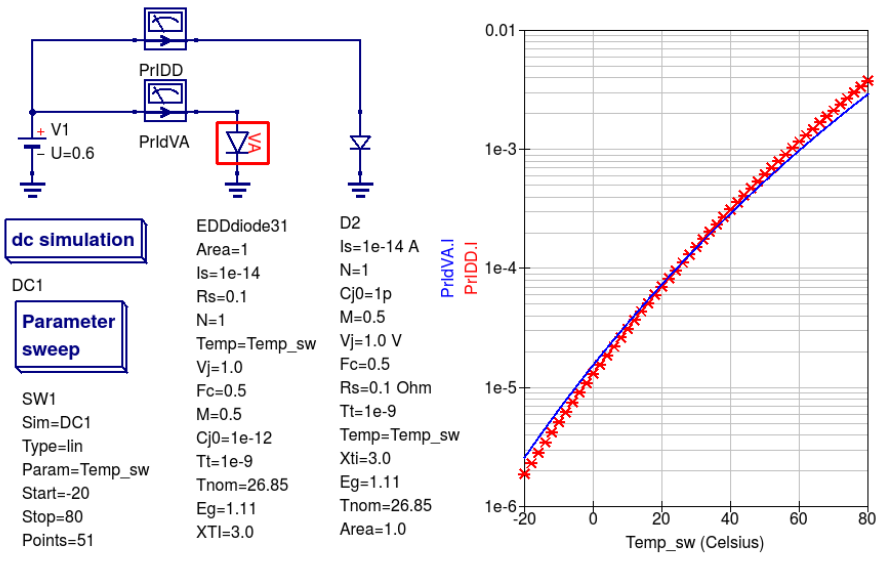
```
`include "disciplines.vams"
`include "constants.vams"
module EDDdiode3(Pcathode, Panode);
inout Pcathode, Panode;
electrical Pcathode, n2, n1, Panode, n4, n3;
parameter real Area=1;   parameter real Is=1e-14;  parameter real Rs=0.1;   parameter real N=1;
parameter real Temp=26.85; parameter real Vj=1.0;  parameter real Fc=0.5;  parameter real M=0.5;
parameter real Cj0=1e-12;   parameter real Tt=1e-9; parameter real Tnom=26.85; parameter real Eg=1.11;
parameter real XTI=3.0;
real RMAX, T1, T2, Con1, Con2, Vt, F1, F2, F3, A, B, Eg_T1, Eg_T2, Vj_T2, Cj0_T2, Is_T2;
analog begin
@(initial_model)
begin
RMAX=1e12;  T1=Tnom+273.15;  T2=Temp+273.15;  Con1=5*N*Vt;  Con2=Fc*Vj; Vt=`P_K*T2/`P_Q;
F1=(Vj/(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_T1=Eg-A*T1*T1/(B+T1); Eg_T2=Eg-A*T2*T2/(B+T2);
Vj_T2=(T2/T1)*Vj-(2*Vt)*ln(exp(1.5*ln(T2/T1)))-((T2/T1)*Eg_T1-Eg_T2);
Cj0_T2=Cj0*(1+M*(400e-6*(T2/T1)-(Vj_T2-Vj)/Vj));
Is_T2=Is*exp((XTI/N)*ln(T2/T1))*limexp((-Eg_T1/Vt)*(1-T2/T1));
end
I(Pcathode,n2) <+ V(Pcathode,n2)/( RMAX );
I(Pcathode,n2) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( RMAX ), "thermal" );
I(n1,Panode) <+ V(n1,Panode)/( Rs );
I(n1,Panode) <+ white_noise( 4.0*`P_K*( 26.85 + 273.15) / ( Rs ), "thermal" );
I(n1,n2)  <+  V(n1,n2) * 1e3; I(n3,n4)  <+  I(n3,n4) *1e-3; I(n3,n4)  <+  V(n1,n2) * 1e6 * 1 ;
I(n2,Pcathode) <+ (V(n2,Pcathode)>-Con1)?Area*Is_T2*(limexp(V(n2,Pcathode)/(N*Vt))-1):-Is_T2;
I(n2,Pcathode) <+ ddt( (V(n2,Pcathode)<=Con2)?Tt*V(n4,n3)+Area*(Cj0_T2*Vj_T2/(1-M))*(1-exp((1-M)*
                       ln(1-V(n2,Pcathode)/Vj_T2))):
                       Tt*V(n4,n3)+Area*Cj0_T2*(F1+(1/F2)*(F3*(V(n2,Pcathode)-Fc*Vj_T2)+(M/(2*Vj_T2))*
                       (V(n2,Pcathode)*V(n2,Pcathode)-Fc*Fc*Vj_T2*Vj_T2))) );
I(n3) <+ (-V(n3))/( RMAX );
I(n3) <+ white_noise( 4.0*`P_K*( 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.



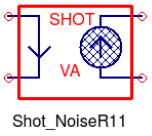
Simulation data for
Qucs EDD model and built-in diode model



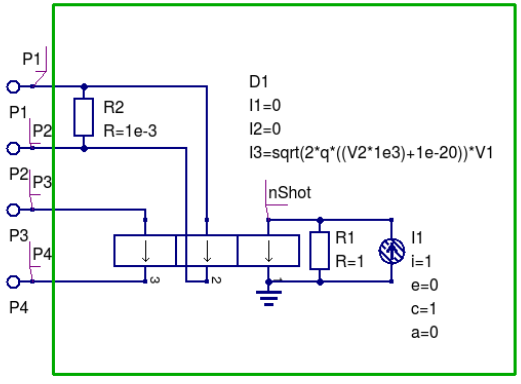
Simulation data for
Verilog-A model and built-in diode model

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.



Shot_NoiseR11

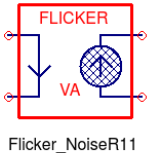


Noise model symbols

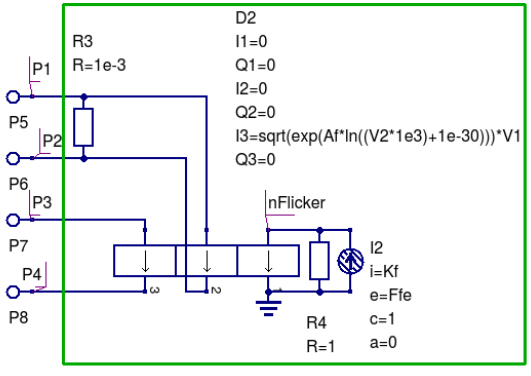
Compact modelling TEMPLATE

```
`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_Q*((V(P1,P2)*1e3)+1e-20))*V(nShot);
end
endmodule
```

Synthesized Verilog-A module code



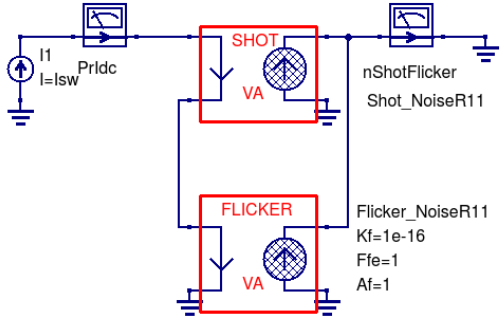
Flicker_NoiseR11
Kf=1e-16
Ffe=1
Af=1




```
`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 Ffe=1;
parameter real Af=1;
analog begin
@(initial_model)
begin
end
I(P2,P1) <+ V(P2,P1)/( 1e-3 );
I(nFlicker) <+ flicker_noise(Kf, Ffe, "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.



ac simulation

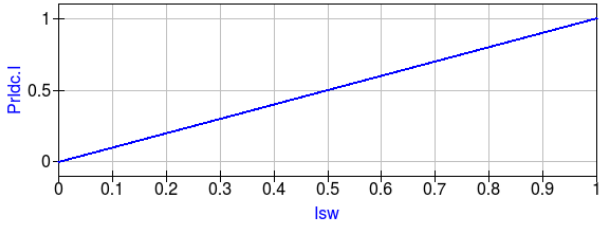
AC1
Type=log
Start=0.01
Stop=100k
Points=141
Noise=yes

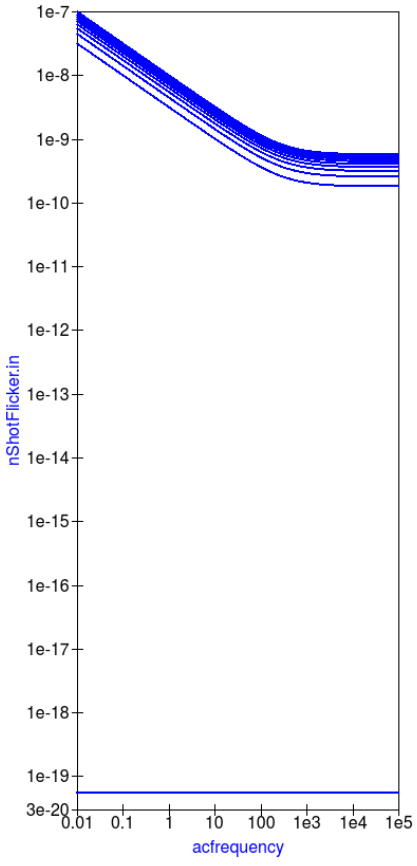
Parameter sweep

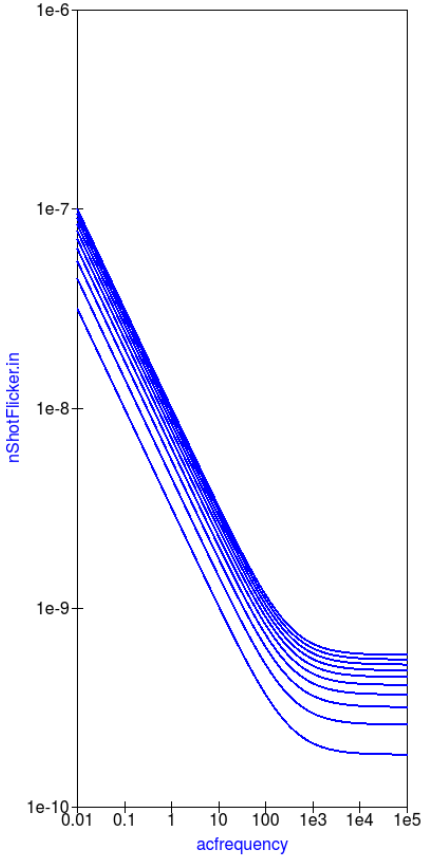
SW1
Sim=AC1
Type=lin
Param=Isw
Start=0
Stop=1
Points=11

dc simulation

DC1



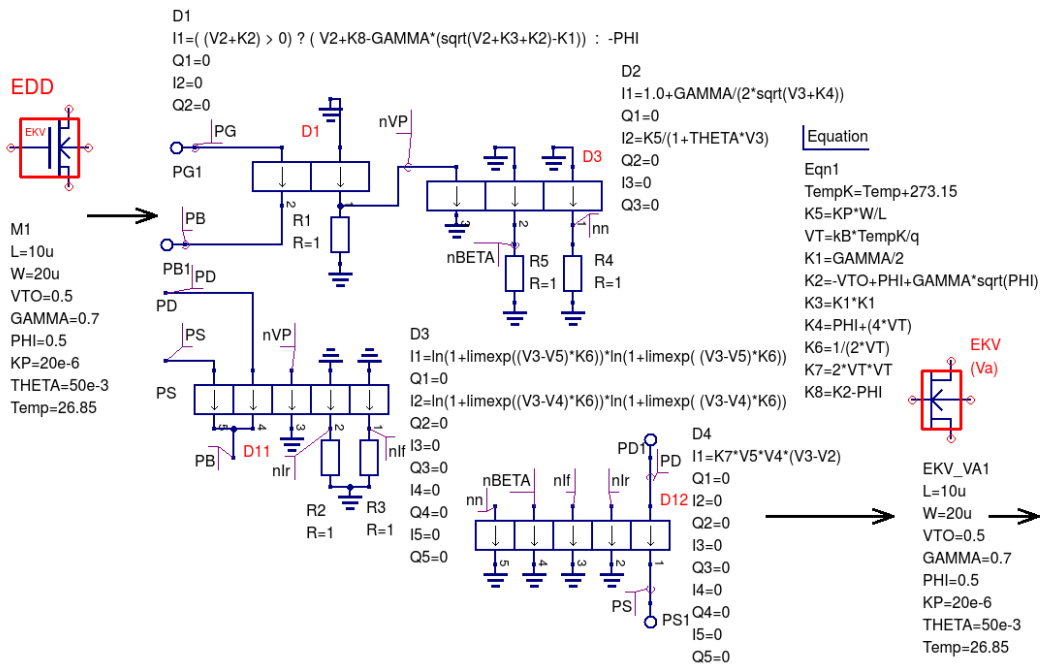




Introduction to the Qucs GPL Verilog-A module synthesizer: Part X

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.

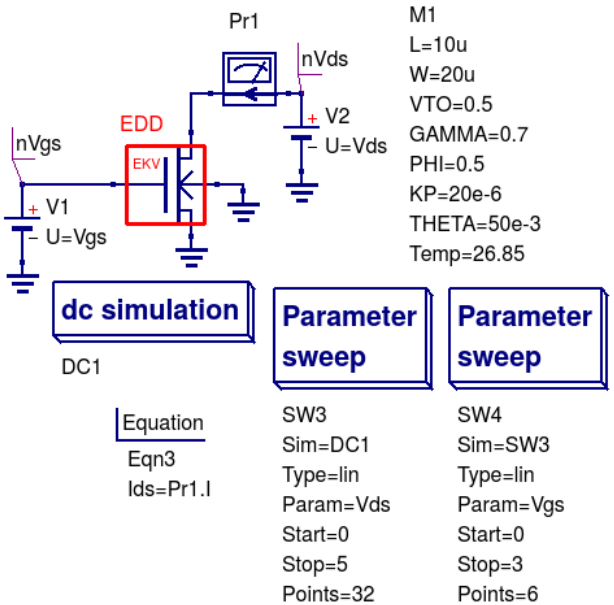


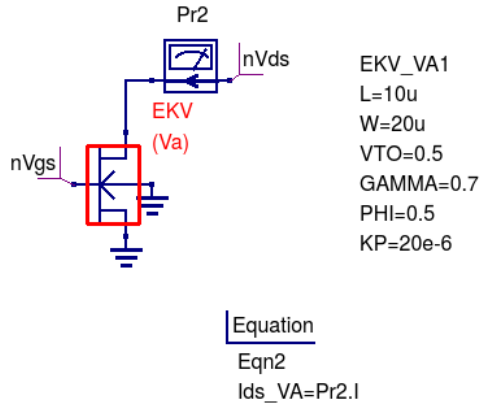
Qucs EDD EKV2p6 Ids=f(Vd, Vg, Vs, Vb) model

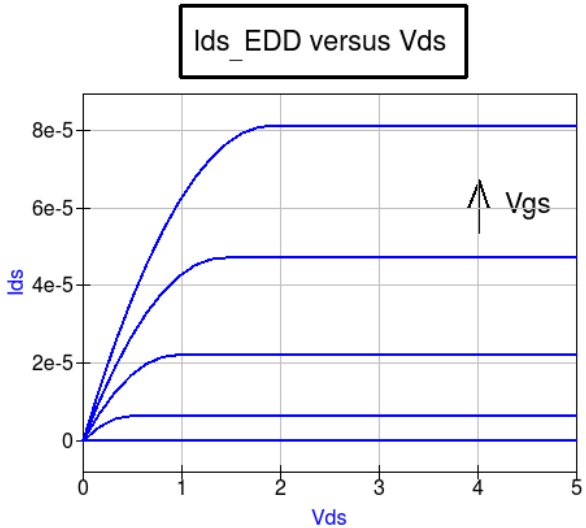
```
`include "disciplines.vams"
`include "constants.vams"
module EKV_VA(PB, PG, PD, PS);
inout PB, PG, PD, PS;
electrical PD, PS, nlr, nlf, nBETA, nn, nVP, PG, PB;
parameter real L=10u;    parameter real W=20u;    parameter real VTO=0.5;
parameter real GAMMA=0.7;    parameter real PHI=0.5;    parameter 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(PHI);
K3=K1*K1;    K4=PHI+(4*VT);
K6=1/(2*VT);    K7=2*VT*VT;    K8=K2-PHI;
end
I(PD,PS) <+ K7*V(nn)*V(nBETA)*(V(nlf)-V(nlr));
I(nVP) <+ -(((V(PG,PB)+K2)>0)?(V(PG,PB)+K8-GAMMA*(sqrt(V(PG,PB)+K3+K2)-K1)):-PHI);
I(nVP) <+ V(nVP)/( 1 );
I(nBETA) <+ -((-V(nBETA))/( 1 ));
I(nn) <+ -(1.0+GAMMA/(2*sqrt(V(nVP)+K4)));
I(nBETA) <+ -(K5/(1+THETA*(-V(nVP))));
I(nn) <+ -((-V(nn))/( 1 ));
I(nlr) <+ -((-V(nlr))/( 1 ));
I(nlf) <+ -((-V(nlf))/( 1 ));
I(nlf) <+ -(ln(1+limexp((V(nVP)-V(PS,PB))*K6))*ln(1+limexp((V(nVP)-V(PS,PB))*K6)));
I(nlr) <+ -(ln(1+limexp((V(nVP)-V(PD,PB))*K6))*ln(1+limexp((V(nVP)-V(PD,PB))*K6)));
end
endmodule
```

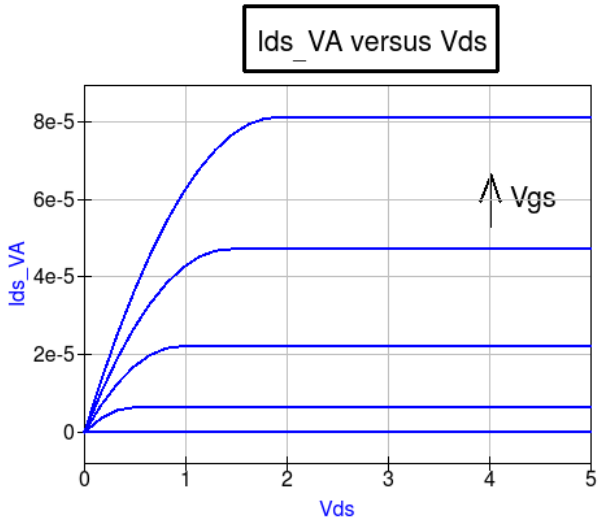
Synthesized EKV2p6 Ids=f(Vd, Vg, Vs, Vb) Verilog-A code

Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS
$I_{ds} = f(V_d, V_g, V_s, V_b)$ swept DC simulation data.









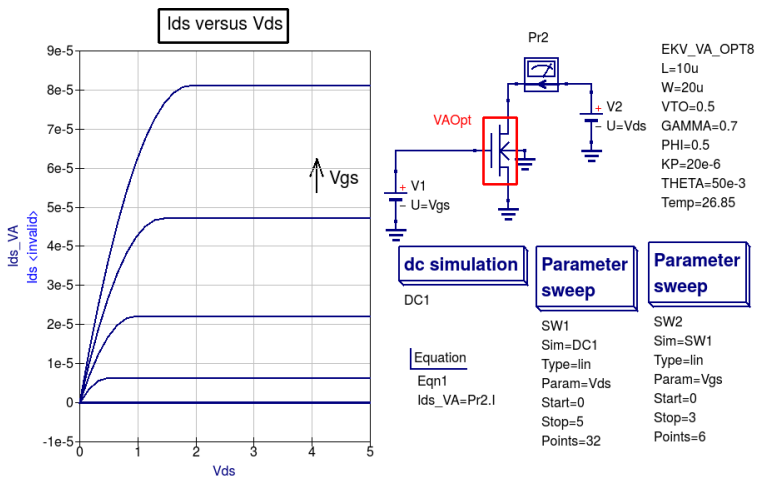
Introduction to the Qucs GPL Verilog-A module synthesizer: Part XII

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);
inout PB, PG, PD, PS;
electrical PD, PS, 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;
real Vg, Vs, Vd, nVP, nBETA, nn, nIf, nIr;
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(PHI);   K3=K1*K1;
K4=PHI+(4*VT);    K6=1/(2*VT);    K7=2*VT*VT;
end
Vg   = V(PG,PB);   Vs   = V(PS,PB);   Vd   = V(PD,PB);
nVP   = (Vg+K2>0)?(Vg+K2-PHI-GAMMA*(sqrt(Vg+K2+K3)-K1)):-PHI;
nBETA = K5/(1+THETA*nVP);
nn    = 1.0+GAMMA/(2*sqrt(nVP+K4));
nIf   = ln(1+limexp((nVP-Vs)*K6))*ln(1+limexp((nVP-Vs)*K6));
nIr   = ln(1+limexp((nVP-Vd)*K6))*ln(1+limexp((nVP-Vd)*K6));
I(PD,PS) <+ K7*nn*nBETA*(nIf-nIr);
end
endmodule
```

EKV2p6 EKV_VA_OPT optimized Verilog-A module code

TEST MODULE



NOTES

1. At this stage in the development of the Qucs synthesizer optimized Verilog-A module code is done manually.
2. General procedure:
   2.1 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 I(a) <+ ..... in the EKV nMOS example]
   2.2 Eliminate as many as possible internal model nodes and remove current to voltage one Ohm conversion resistors.
   [zero left in EKV nMOS example]

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.

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)