Building device models and circuit macromodels with the Qucs GPL circuit simulator: A demonstration

M.E. Brinson: Faculty of Computing, London Metropolitan University, UK;
Qucs development team: mbrin72043@yahoo.co.uk

S. Jahn: Qucs project manager, Munich, Germany; stefan@lkcc.org

- **Background**
- **Introduction to circuit simulation with Qucs**
- **Building a compact semiconductor model of an npn bipolar phototransistor**
- **Compiling and linking Verilog-A code to Qucs**
- **Summary**

Presented on April 2, 2009, at the COMON meeting, IHP, Frankfurt (Oder), Germany

Background

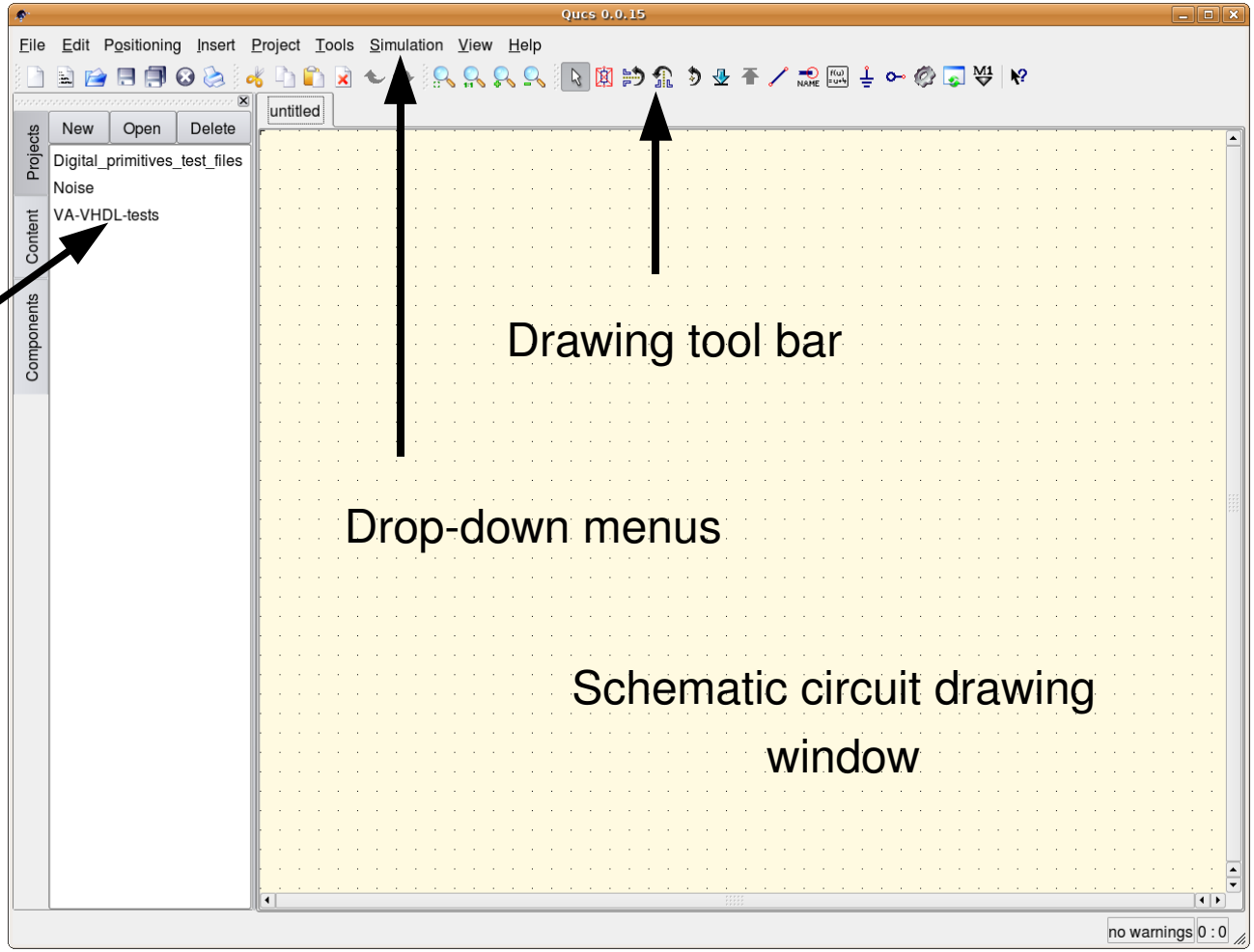
- Qucs is an open source circuit simulator with a graphical user interface (GUI) based on QT® by Trolltech®
- Qucs is available from http://qucs.sourceforge.net/
- Qucs aims to support a wide range of circuit simulation, including DC, AC, transient, S-parameter, noise analysis, Harmonic Balance analysis and digital system simulation
- Developed using GNU/Linux under the General Public License (GPL)
- Available for most of the popular computer operating systems, including GNU/Linux, Windows®, Solaris®, NetBSD, FreeBSD and MacOS®
- Qucs is one of the GPL circuit simulators taking part in the MOS-AK Verilog-A standardization initiative
- Developers: Loose group of international engineers, scientists and programmers
- Qucs is multilingual and has been translated into: Romanian, German Italian, Polish, French, Portuguese, Spanish, Japanese, Hungarian, Hebrew, Swedish, Turkish, Russian, Czech and Catalan

Introduction to circuit simulation with Qucs

Qucs current versions: production 0.0.14; development 0.0.15

Start Qucs by issuing the command **# qucs**

OR by clicking the appropriate icon on your start menu or desktop



File -> Application settings

Open projects folders

Drawing tool bar

Drop-down menus

Schematic circuit drawing window

Introduction to circuit simulation with Qucs

Tool suite: Qucs consists of a group of standalone programs which interact with each other through a graphical user interface (GUI)

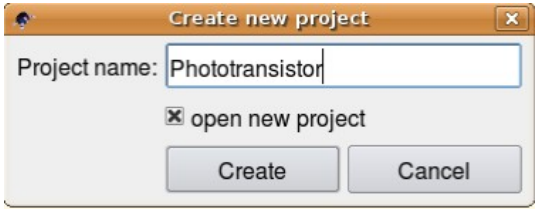
- the GUI: used to create schematics, setup simulations, display simulation results and writing VHDL/Verilog code amongst other tasks
- the analogue simulator: a command line program which is run by the GUI to simulate user generated schematics
- a simple text editor: which is used to display netlists, simulation log files and to edit SPICE, Touchstone and other text files
- a filter synthesis program: used for the design of passive filters
- a transmission line calculator: used for the design and analysis of transmission lines
- a component library
- an attenuator synthesis program: used for the design of passive attenuators
- a command line conversion program: this program acts as a tool for the import and export of data sets, netlists and schematics from and to other CAD/EDA software

The GUI is also used by Qucs to launch other EDA tools: FreeHDL (http://www.freehdl.seul.org) for digital VHDL, Icurus Verilog (http://icarus.com/eda/verilog/) for digital Verilog and ASCO (http://asco.sourceforge.net/) for circuit optimization

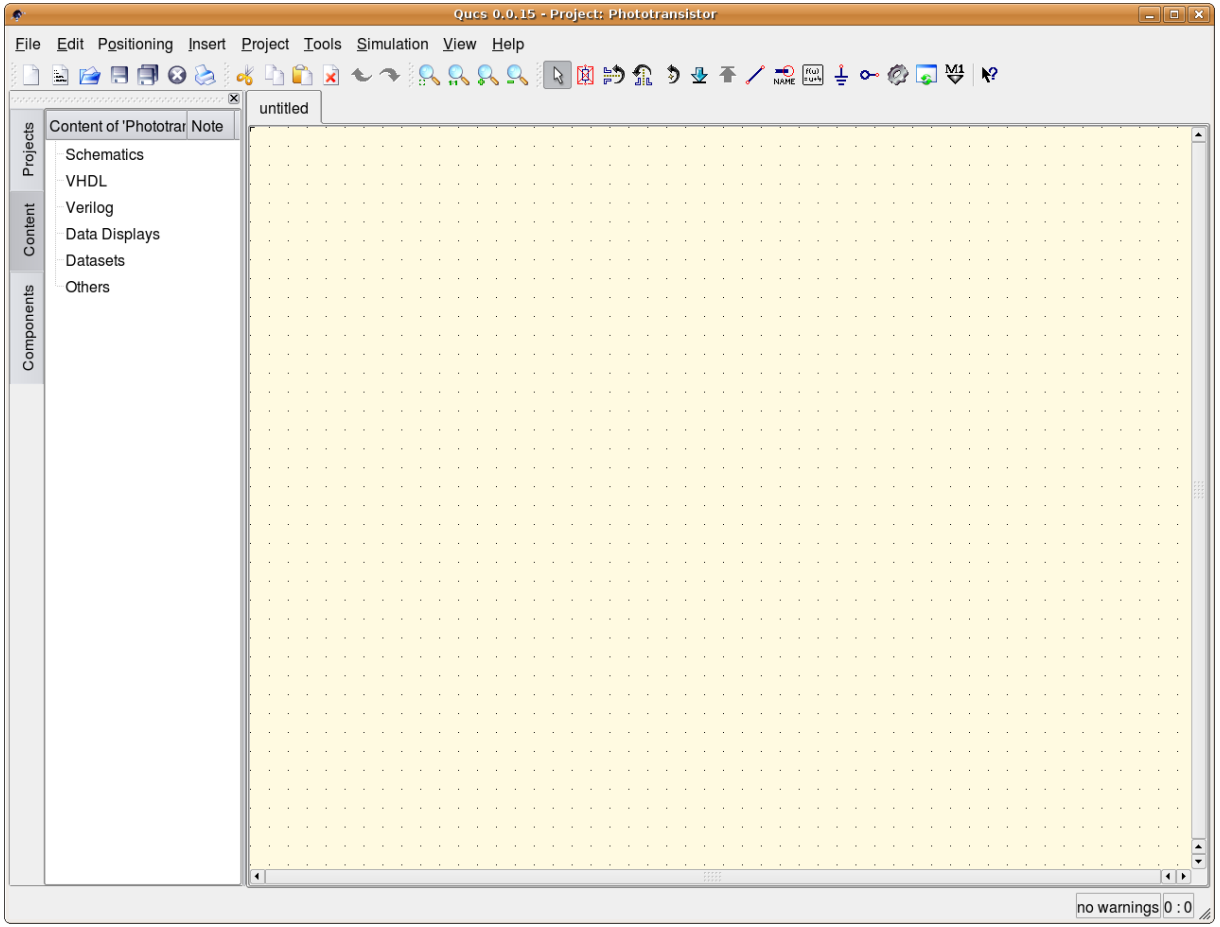
Introduction to circuit simulation with Qucs

Setting up schematics

Either press the **New** button above the projects folder or use the menu entry **Project → New Project** and enter the project name

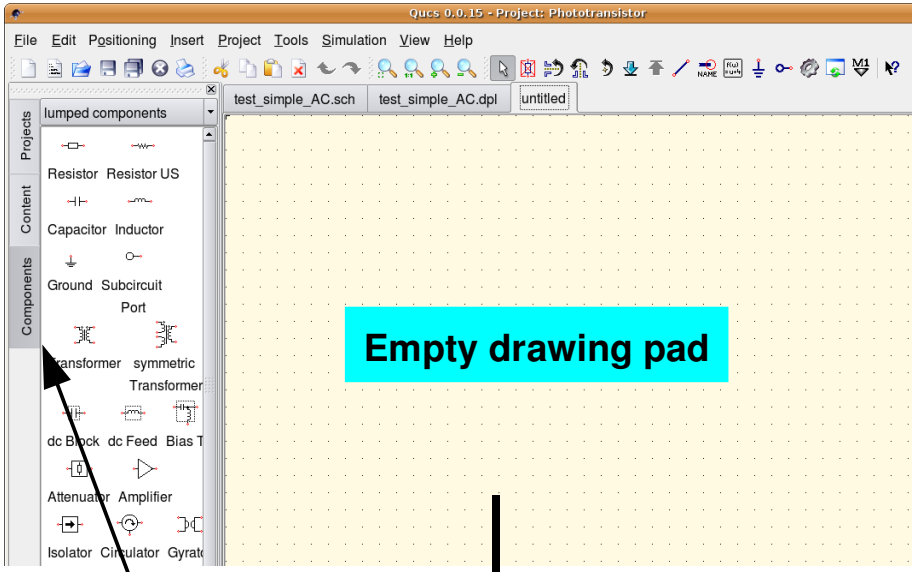


Confirm the dialog by pressing "**Create** " button. Qucs then opens the Phototransistor project and displays a blank project window with the **Content** tab open.

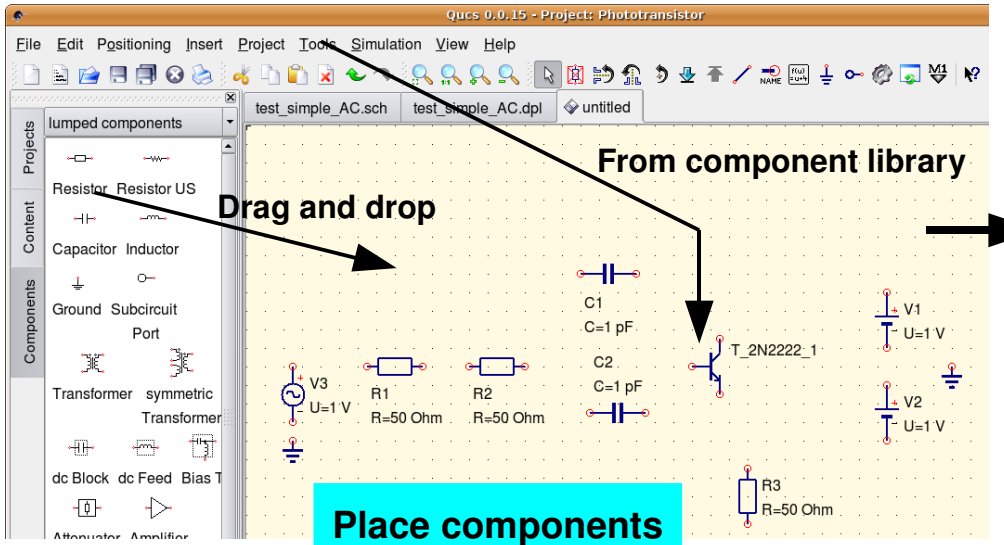


Introduction to circuit simulation with Qucs

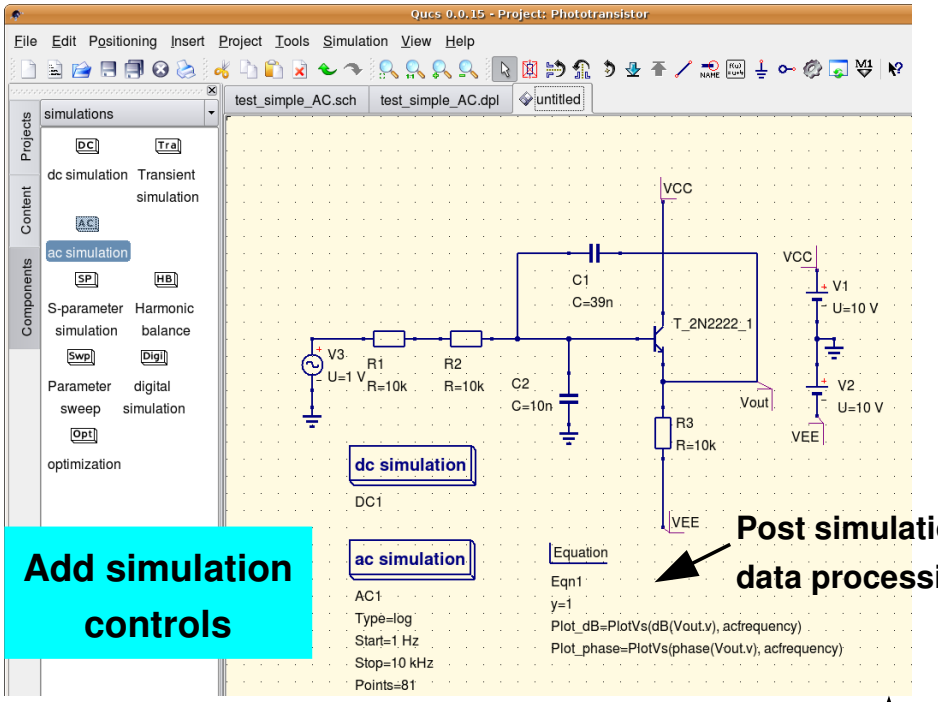
Building a simple schematic

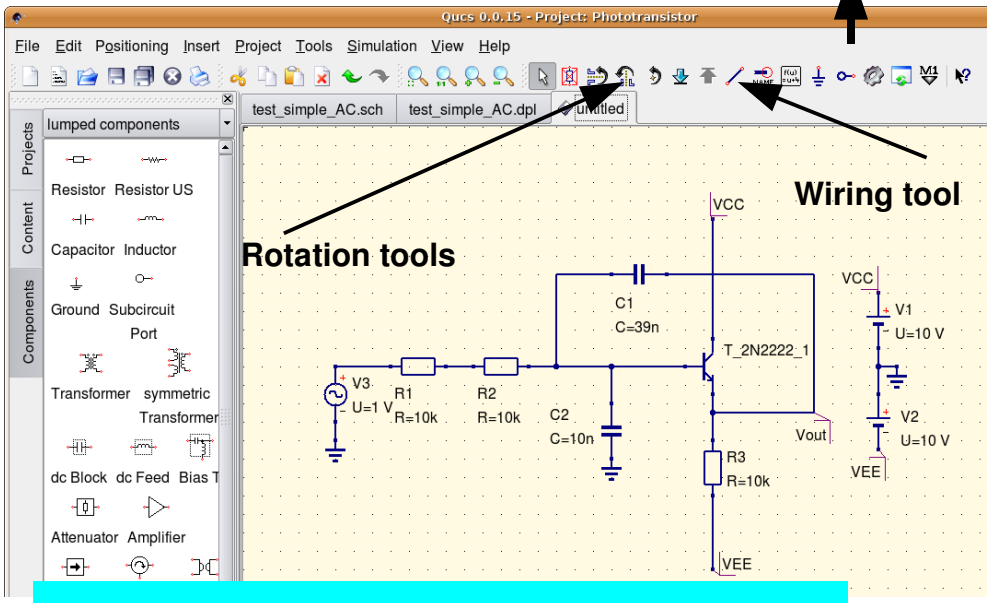


Component tab ON



Built-in components

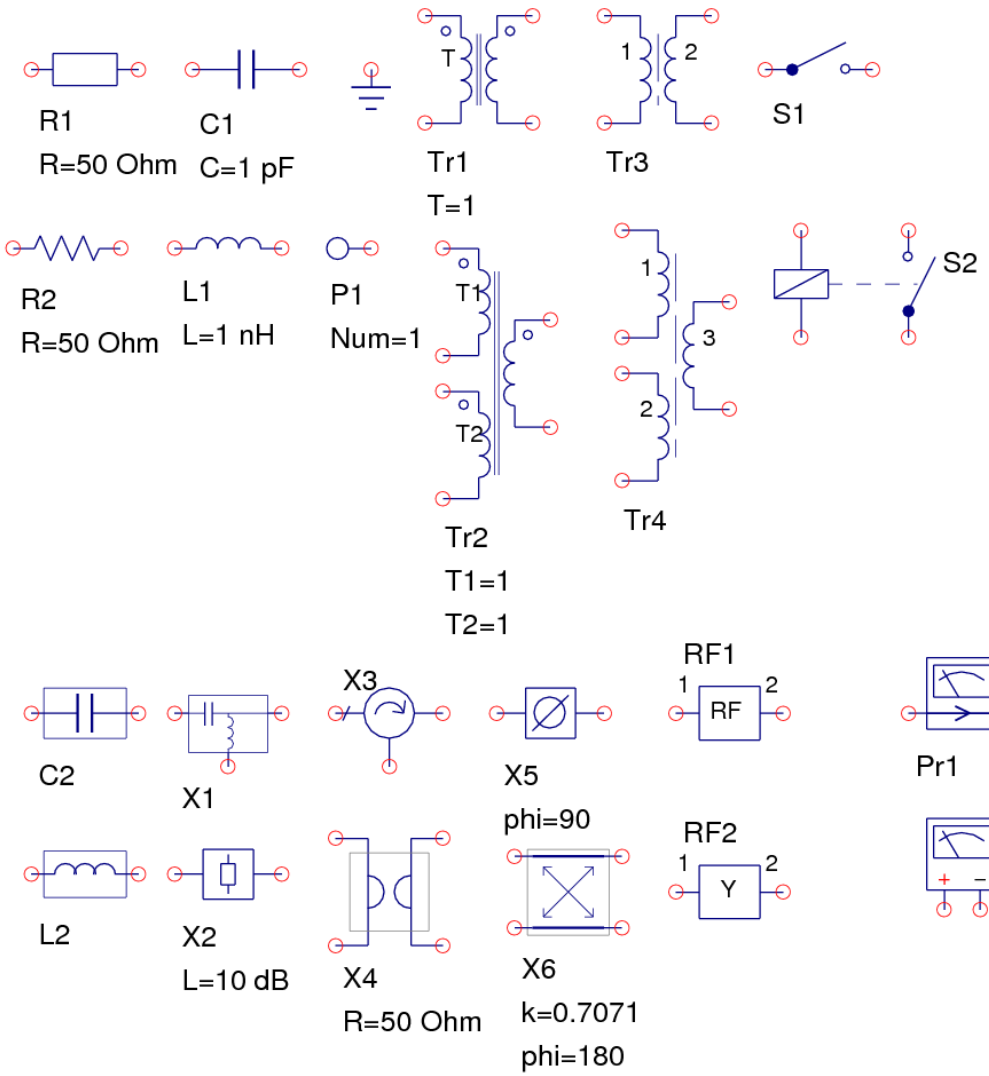




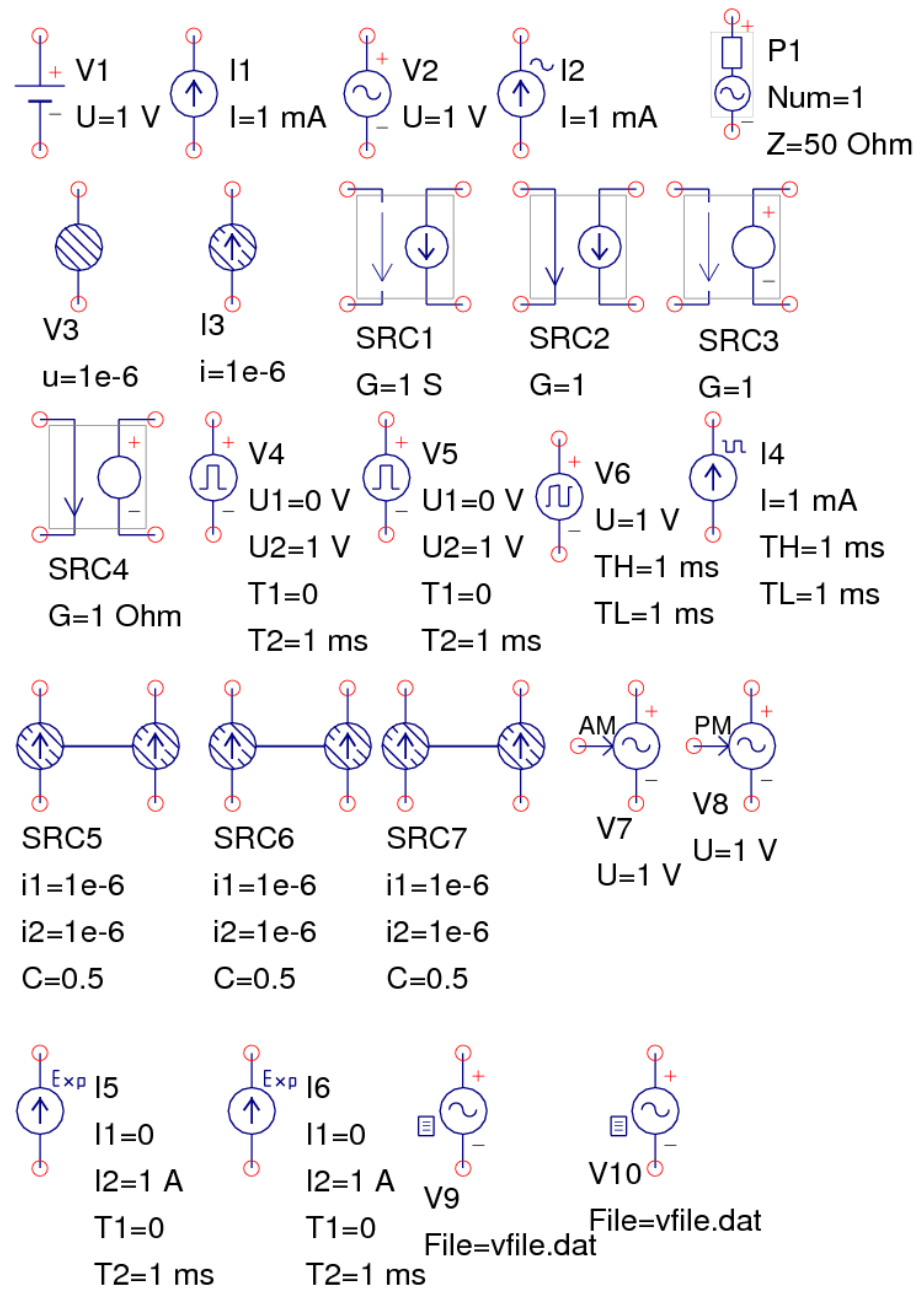
Introduction to circuit simulation with Qucs

Qucs built-in analogue components: 1

Lumped components



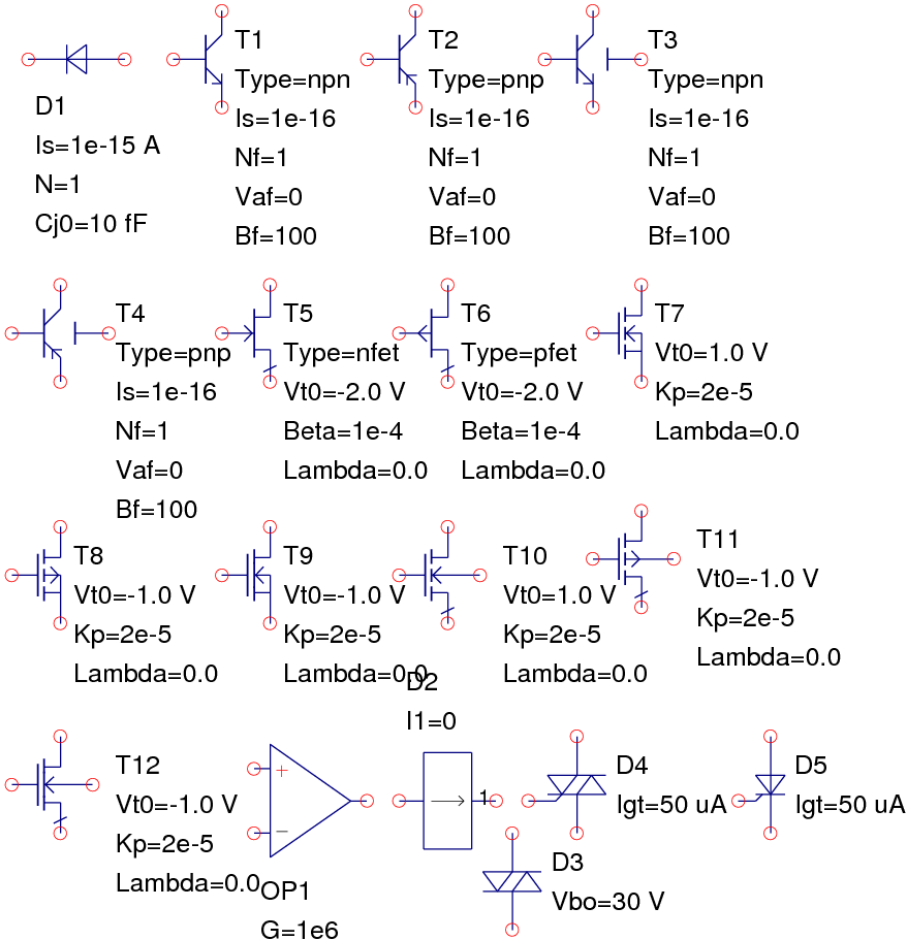
Sources



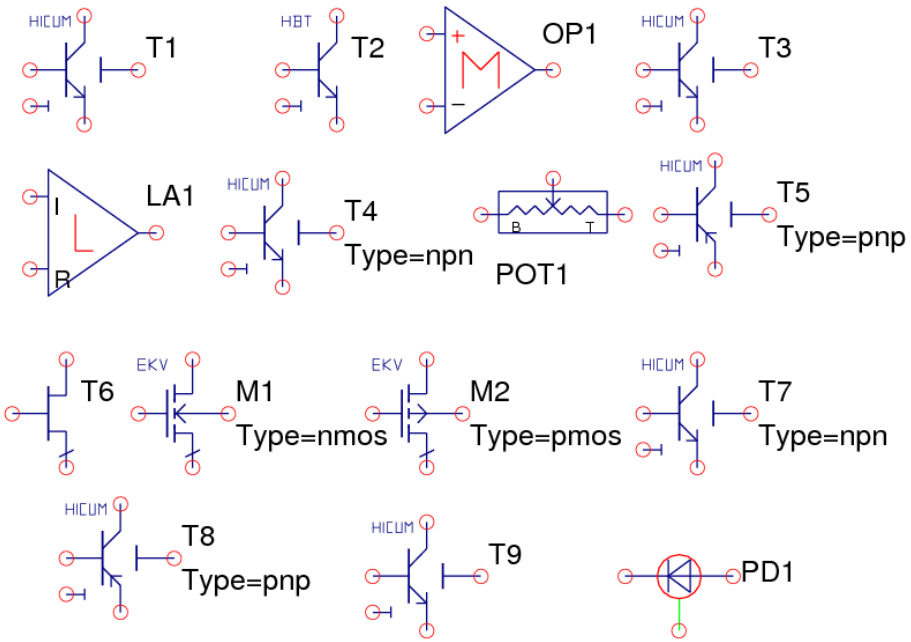
Introduction to circuit simulation with Qucs

Qucs built-in analogue components: 2

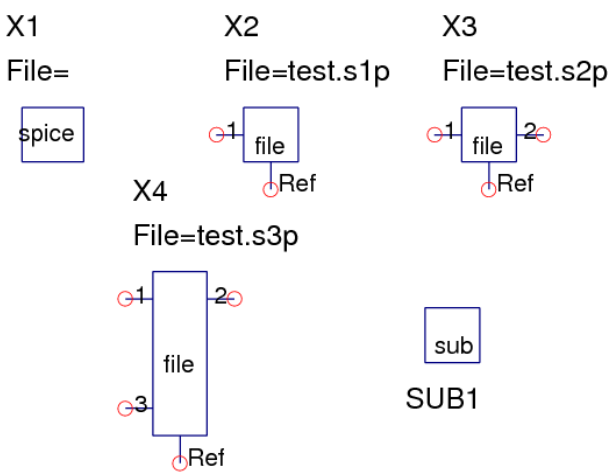
Non-linear components



Verilog-A components

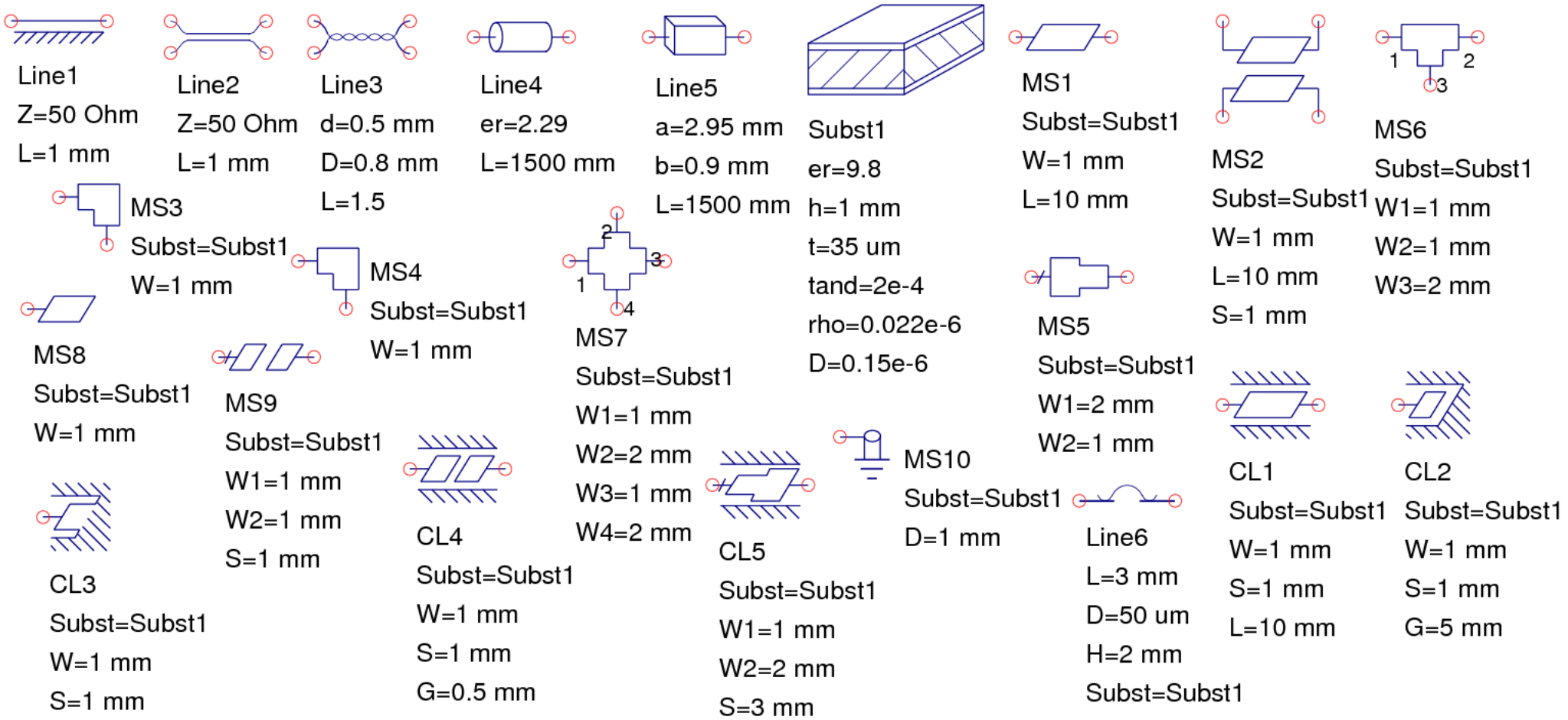


File components



Introduction to circuit simulation with Qucs

Qucs built-in analogue components: 3



Introduction to circuit simulation with Qucs

Qucs built-in analogue components: 4

Simulations

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

S parameter simulation

SP1
Type=lin
Start=1 GHz
Stop=10 GHz
Points=19

Harmonic balance simulation

HB1
n=4

Parameter sweep

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

digital simulation

Digi1
Type=TruthTable

Optimization

Opt1

Diagrams

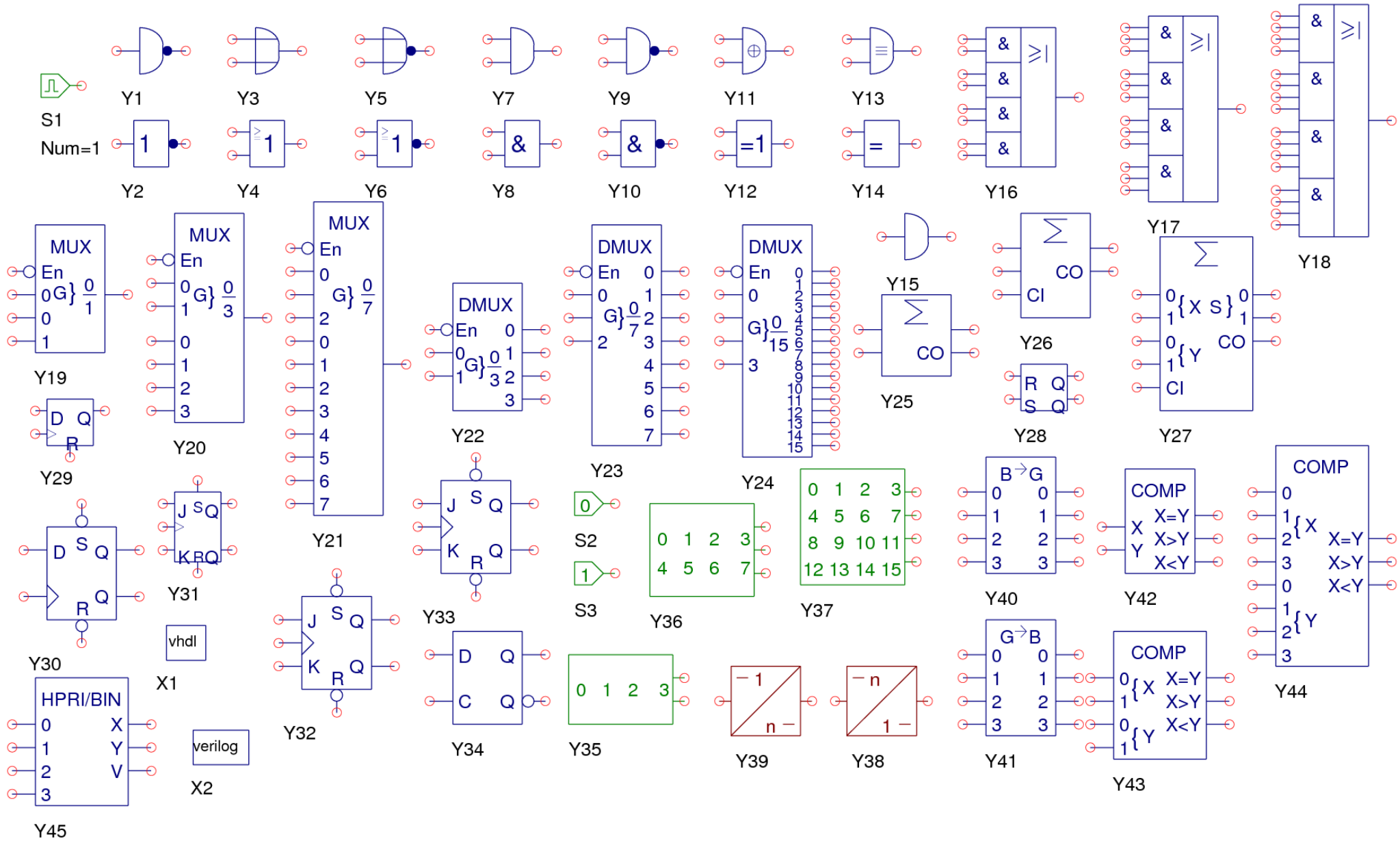
Cartesian Polar Tabular
Smith Chart Admittance Smith
Polar-Smith Combi
Smith-Polar Combi
3D-Cartesian
Locus curve
Timing Diagram
Truth Table

Paintings

Line Arrow Text
Ellipse Rectangle
filled Ellipse
filled Rectangle
Elliptic Arc

Introduction to circuit simulation with Qucs

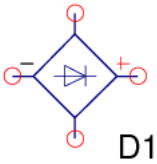
Qucs built-in digital components



Introduction to circuit simulation with Qucs

Qucs built-in libraries of components

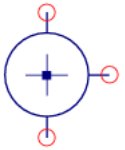
Diode Bridges



Semiconductor Diodes

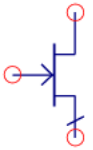
D_1N4148_1
Is=222p
N=1.65

Ideal functional components



VADD1

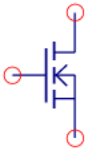
JFETs



J201_1

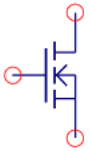
LEDs

MOSFETs



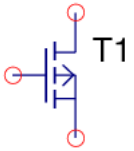
T_BSS123_1
Vt0=1

NMOSFETs

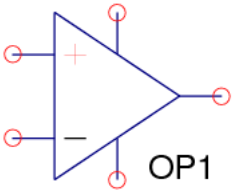


M2N3796_1

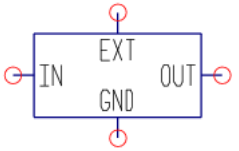
PMOSFETs



OPAMPs

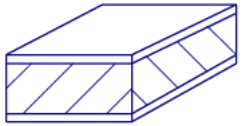


Regulators



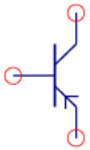
PVR100AZ_B12V_1

Substrates



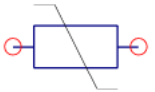
RO3003_1
er=3.00
h=0.8 mm
t=35 um
tand=0.0013
rho=0.022e-6
D=1.4e-6

Transistors



T_2DA1774R_1

Varistors



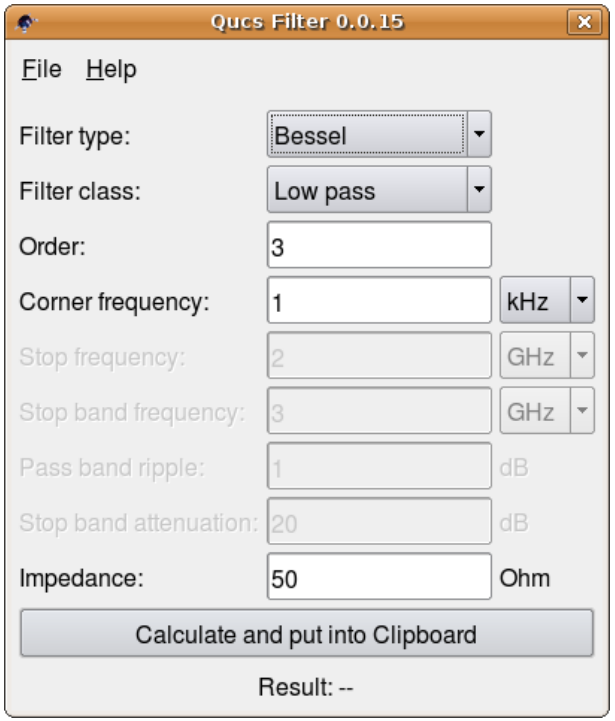
VDR1
Tol=0

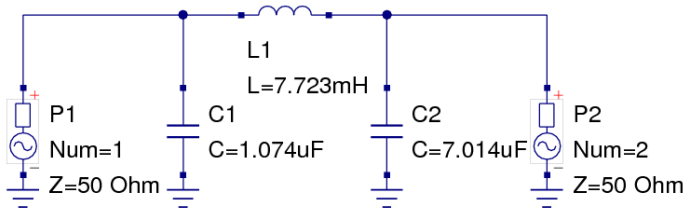
Zener Diodes

Introduction to circuit simulation with Qucs

Generation of circuits using Qucs design tool suite:
passive filter synthesis example

Low pass filter synthesis



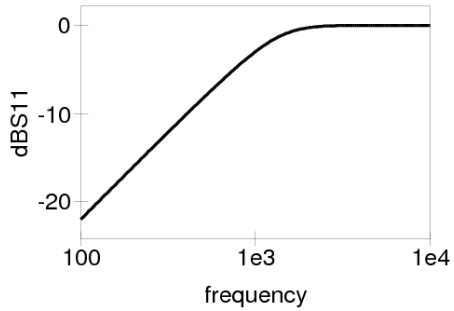


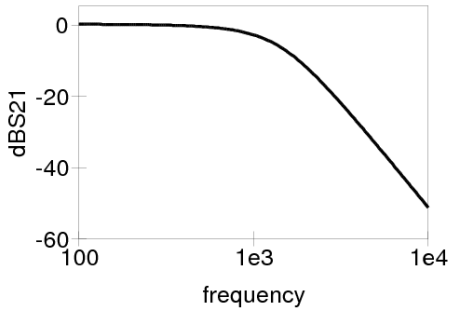
S parameter simulation

SP1
Type=log
Start=100Hz
Stop=10kHz
Points=200

Equation
Eqn1
dBS21=dB(S[2,1])
dBS11=dB(S[1,1])

Bessel low-pass filter
1kHz cutoff, PI-type,
impedance matching 50 Ohm

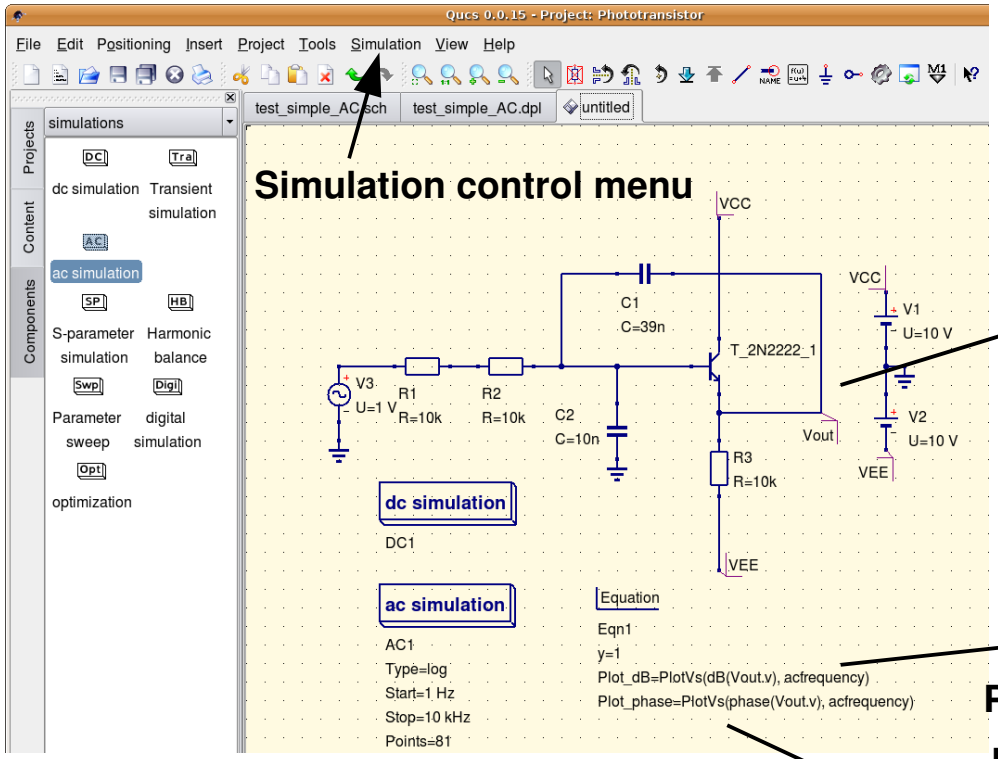




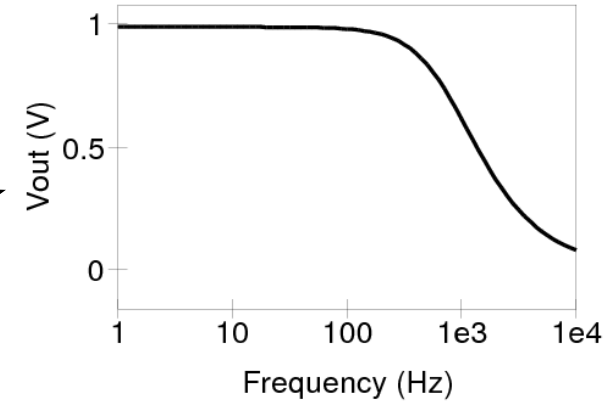
Other synthesis/design tools:
Text Editor, Line Calculator, Matching Circuits and Attenuator Synthesis

Introduction to circuit simulation with Qucs

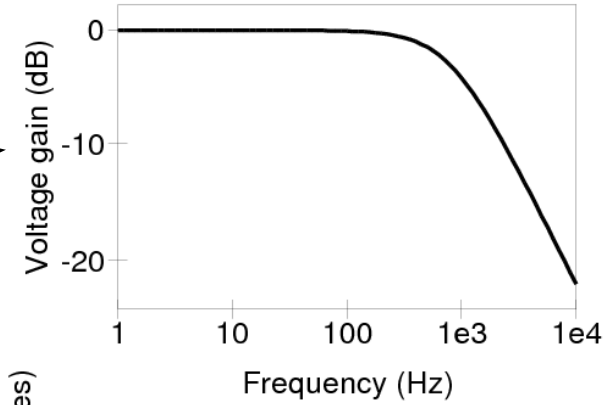
Launching a simulation

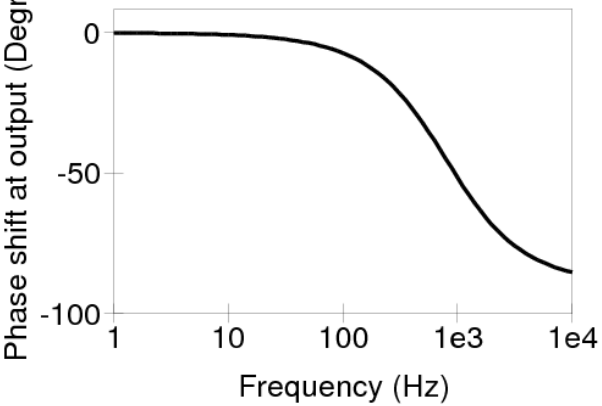


Plotted data



Post-simulation processed and plotted data





Data sets

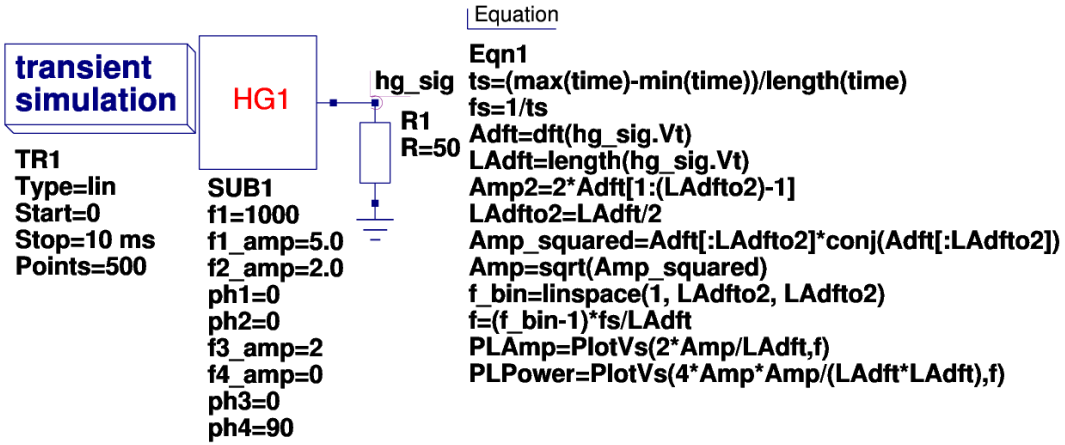
S[1,1]	S-parameter value
nodename.V	DC voltage at node nodename
name.I	DC current through component name
nodename.v	AC voltage at node nodename
name.i	AC current through component name
nodename.vn	AC noise voltage at node nodename
name.in	AC noise current through component name
nodename.Vt	Transient voltage at node nodename
name.It	Transient current through component name

Introduction to circuit simulation with Qucs

Qucs: Post simulation data processing – MATLAB®*/Octave style**

Equation blocks > data processing >Plots/Tables

Simulation data sets plus any of the following



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

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

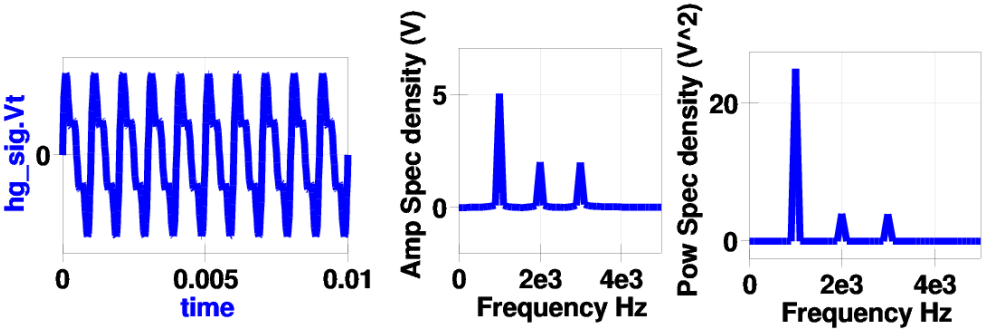
Immediate: 2.5, 1.4+j5.1, [1, 3, 4, 5, 7], [11, 12; 21, 22]

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

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

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

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



Functions:

abs adjoint angle arccos arccosec arccot arcosech arcosh arcoth 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 rtoy rtoz runavg sec sech sign sin sinc sinh sqr sqrt srandom StabCircleL StabCircleS StabFactor StabMeasure stddev step stos stoy stoz sum tan tanh Time2Freq transpose twoport unwrap variance vt w2dbm xvalue ytor ytos ytoz yvalue ztor ztos ztoy

* MATLAB, Mathworks, http://www.mathworks.com/

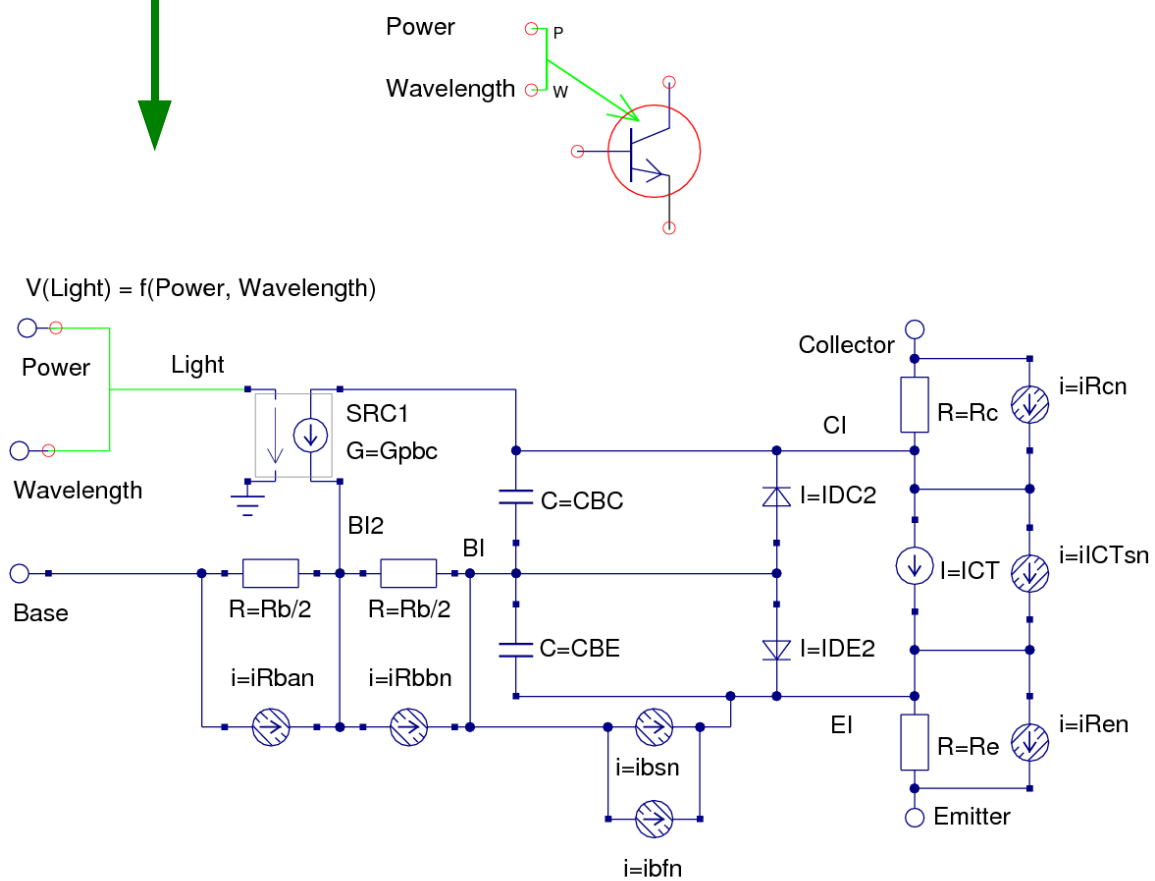
** Octave, http://www.gnu.org/software/octave/

Building a compact semiconductor model of an npn bipolar phototransistor

Schematic symbol, default parameter list and equivalent circuit of an npn bipolar phototransistor

Light bus

The phototransistor consists of an Ebers-Moll bipolar junction transistor model which has been extended to include depletion and diffusion capacitance, forward and reverse Early effects, high current forward and reverse beta degradation, thermal and shot noise, plus a light bus which connects external light signals to the phototransistor



Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9

Building a compact semiconductor model of an npn bipolar phototransistor

Phototransistor model parameters

Name	Symbol	Description	Unit	Default
BF	βf	Forward beta		100
BR	βr	Reverse beta		0.1
Is	Is	Saturation current	A	1e-10
Nf	*Nf*	Forward emission coefficient		1
Nr	*Nr*	Reverse emission coefficient		1
Var	*Var*	Reverse Early voltage	V	100
Vaf	*Vaf*	Forward Early voltage	V	100
Mje	*Mje*	Base-emitter exponential factor		0.33
Vje	*Vje*	Base-emitter built-in potential	V	0.75
Cje	Cje	Base-emitter zero-bias depletion capacitance	F	1p
Mjc	*Mjc*	Base-collector exponential factor		0.33
Vjc	*Vjc*	Base-collector built-in potential	V	0.75
Cjc	Cjc	Base-colector zero-bias depletion capacitance	F	1p
Tr	*Tr*	Ideal reverse transit time	s	100n
Tf	*Tf*	Ideal forward transit time	s	0.1n
Ikf	*Ikf*	High current corner for forward beta	A	0.5
Ikr	*Ikr*	High current corner for reverse beta	A	0.5
Rc	*Rc*	Collector series resistance	Ω	2
Re	*Re*	Emitter series resistance	Ω	1
Rb	*Rb*	Base series resistance	Ω	100
Kf	*Kf*	Flicker noise coefficient		1e-12
Ffe	*Ffe*	Flicker noise frequency exponent		1
Af	*Af*	Flicker noise exponent		1
Responsivity	*Responsivity*	Responsivity at peak wavelength	A/W	1.5
P0	*P0*	Relative selectivity polynomial coefficient	%	2.6122×10^{3}
P1	*P1*	Relative selectivity polynomial coefficient	%/nm	-1.4893×10^{1}
P2	*P2*	Relative selectivity polynomial coefficient	%/nm^2	3.0332×10^{-2}
P3	*P3*	Relative selectivity polynomial coefficient	%/nm^3	-2.5708×10^{-5}
P4	*P4*	Relative selectivity polynomial coefficient	%/nm^4	7.6923×10^{-9}

Building a compact semiconductor model of an npn bipolar phototransistor

Bipolar phototransistor model equations

DC I/V characteristics

$$IEC = Is \cdot \left[\exp\left(\frac{V(BI,CI)}{Nr \cdot vt(300)} \right) - 1 \right] \quad IDC2 = \frac{IEC}{\beta r} + GMIN \cdot V(BI,CI) \quad IDE2 = \frac{ICC}{\beta f} + GMIN \cdot V(BI,EI)$$

$$ICC = Is \cdot \left[\exp\left(\frac{V(BI,EI}{Nf \cdot vt(300)} \right) - 1 \right] \quad ICT = \frac{[ICC - IEC]}{\left(\frac{q1}{2}\right) \cdot \left(1 + \sqrt{1 + 4 \cdot q2}\right)} \quad q1 = 1 + \frac{V(BI,CI)}{Vaf} + \frac{V(BI,EI)}{Var}$$

$$q2 = \frac{ICC}{Ikf} + \frac{IEC}{Ikr} \quad vt(T) = \frac{K \cdot T}{q}$$

Capacitance

$$CBC = \frac{dQ(BI,CI)}{dV(BI,CI)} = \frac{Cjc}{\left[1 - \frac{V(BI,CI)}{Vjc}\right]^{Mjc}} + Tr \cdot \frac{dIEC}{dV(BI,CI)} \quad \forall\ V(BI,CI) < \frac{Vjc}{2}$$

$$= 2^{Mjc} \cdot Cjc \cdot \left[\frac{2 \cdot Mjc \cdot V(BI,CI)}{Vjc} + (1 - Mjc) \right] + Tr \cdot \frac{dIEC}{dV(BI,CI)} \quad \forall\ V(BI,CI) >= \frac{Vjc}{2}$$

$$CBE = \frac{dQ(BI,EI)}{dV(BI,EI)} = \frac{Cje}{\left[1 - \frac{V(BI,EI)}{Vje}\right]^{Mje}} + Tr \cdot \frac{dICC}{dV(BI,EI)} \quad \forall\ V(BI,EI) < \frac{Vje}{2}$$

$$= 2^{Mje} \cdot Cje \cdot \left[\frac{2 \cdot Mje \cdot V(BI,EI)}{Vje} + (1 - Mje) \right] + Tr \cdot \frac{dICC}{dV(BI,EI)} \quad \forall\ V(BI,EI) >= \frac{Vje}{2}$$

Photo current

$$Iopt = Gpbc \cdot Popt \quad Gpbc = \frac{\mathrm{Rel}Sensitivity \cdot \mathrm{Re}sponsivity}{\beta f \cdot 100}$$

$$\mathrm{Rel}Sensitivity = P0 + P1 \cdot \lambda + P2 \cdot \lambda^2 + P3 \cdot \lambda^3 + P4 \cdot \lambda^4$$

Noise

$$iRcn^2 = \frac{4 \cdot K \cdot T}{Rc} \cdot \Delta f \quad i\mathrm{Re}n^2 = \frac{4 \cdot K \cdot T}{\mathrm{Re}} \cdot \Delta f \quad iICTsn^2 = 2 \cdot q \cdot IC \cdot \Delta f$$

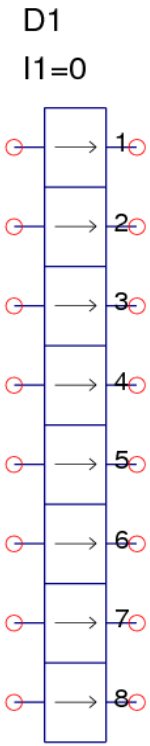
$$iRban^2 = \frac{8 \cdot K \cdot T}{Rb} \cdot \Delta f \quad iRbbn^2 = \frac{8 \cdot K \cdot T}{Rb} \cdot \Delta f \quad ibsn^2 = 2 \cdot q \cdot IB \cdot \Delta f \quad ibfn^2 = Kf \cdot \frac{IB^{Af}}{f^{Ffe}} \cdot \Delta f$$

Where K is the Boltzmann constant, T is the temperature in Kelvin, q is the electron charge, GMIN is a small admittance in parallel with the device junctions, △f is the noise frequency bandwidth in Hz and λ is the light wavelength in nm. Other symbols and node names are defined in the previous slides.

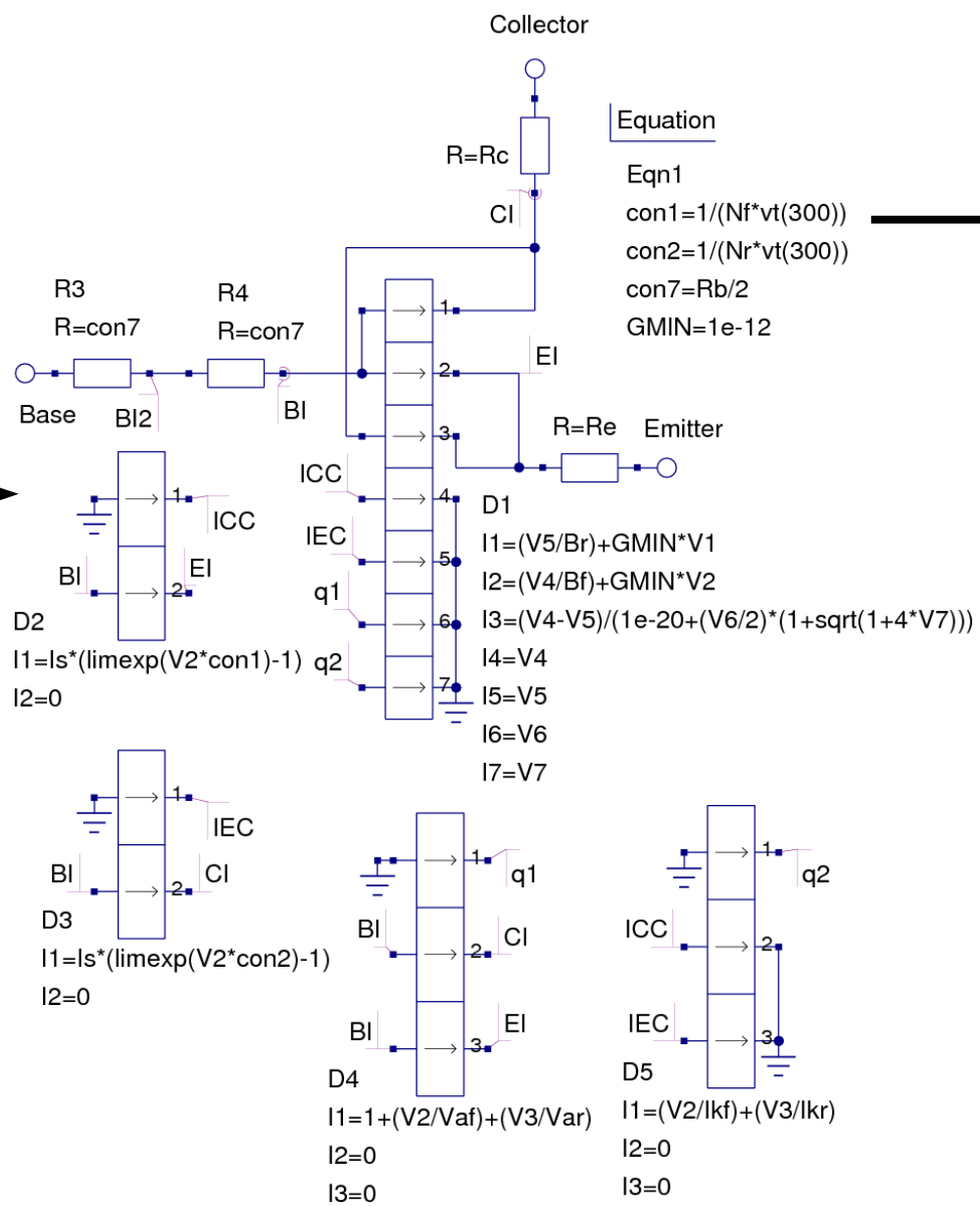
Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 1 – the basic npn transistor model

Equation defined device

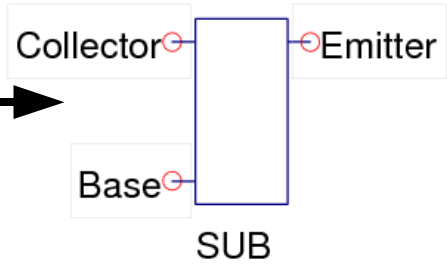


Large signal DC model

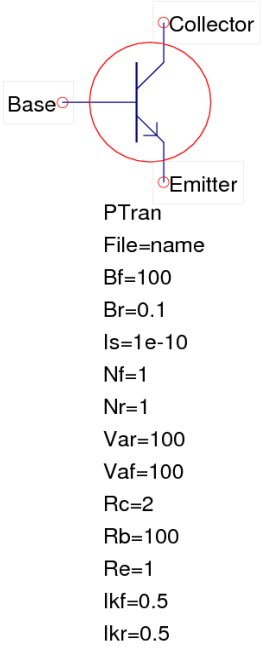


Generate symbol

Click right-hand mouse button on drawing pad



Redrawn symbol



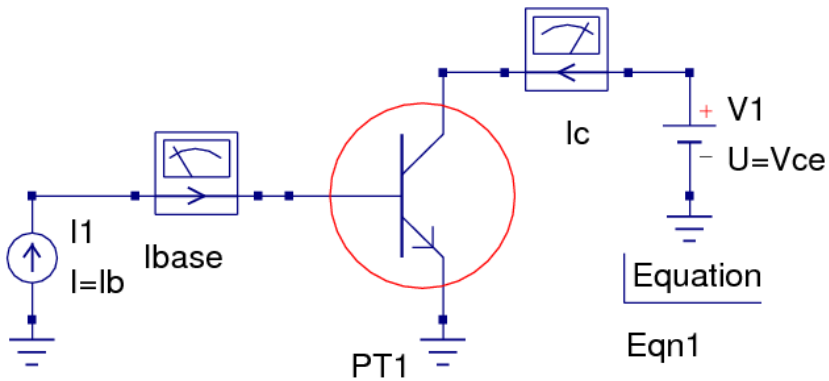
$$I=I(V),\ g=\frac{dI}{dV}$$

$$Q=Q(I,V)$$

$$C=\frac{dQ}{dV}=\frac{dQ(V)}{dV}+\frac{dQ(I)}{dI}\cdot g$$

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 2 – DC simulation tests



Equation

Eqn1

Beta=Ic.I/Ibase.I

PLBeta=PlotVs(Beta, Ic.I)

PT1

Bf=100

Br=0.1

Is=1e-10

Nf=1

Nr=1

Var=100

Vaf=100

Rc=2

Rb=100

Re=1

Ikf=0.5

Ikr=0.5

dc simulation

DC1

Parameter sweep

SW1

Sim=SW2

Type=log

Param=Ib

Start=1e-6

Stop=1m

Points=16

Parameter sweep

SW2

Sim=DC1

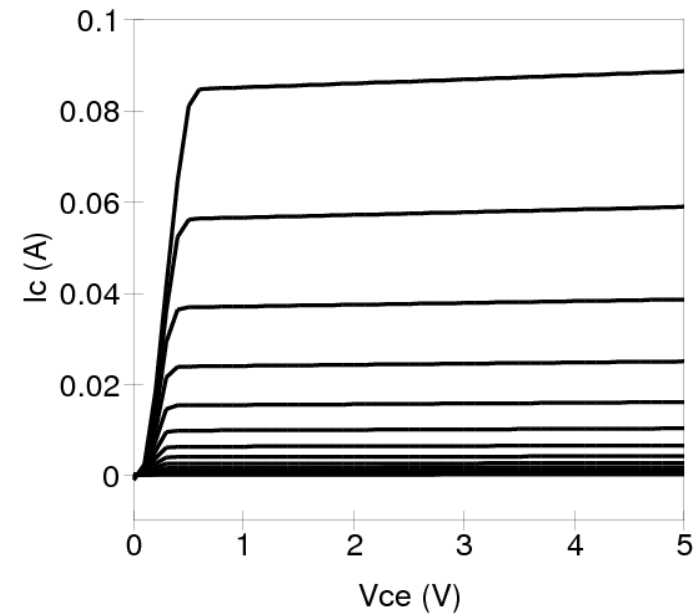
Type=lin

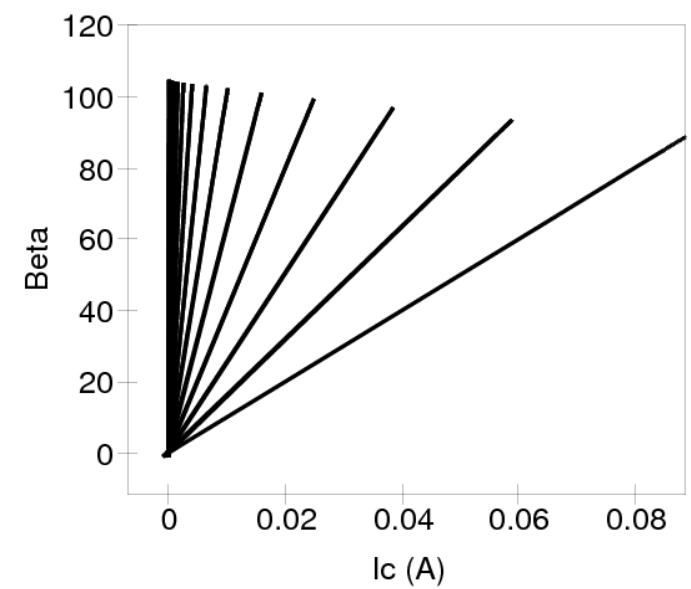
Param=Vce

Start=0

Stop=5

Points=51





Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 3 – Adding capacitance to the phototransistor model

Charge equations

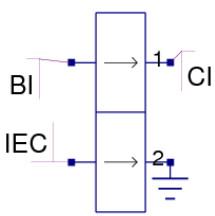
$$Q(BI,CI) = \int_0^{V(BI,CI)} CBC \cdot dV = Tr \cdot IEC + 2^{Mjc} \cdot Cjc \cdot \left[Mjc \cdot \frac{V(BI,CI)^2}{Vjc} + (1-Mjc) \cdot V(BI,CI) \right] \quad \forall\ V(BI,CI) > \frac{Vjc}{2}$$

$$= Tr \cdot IEC + \frac{Cjc \cdot Vjc}{1-Mjc} \left[1 - \left\{ 1 - \frac{V(BI,CI)}{Vjc} \right\}^{1-Mjc} \right] \quad \forall\ V(BI,CI) <= \frac{Vjc}{2}$$

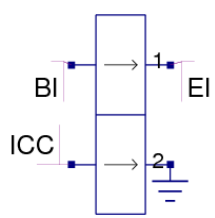
$$Q(BI,EI) = \int_0^{V(BI,EI)} CBE \cdot dV = Tf \cdot ICC + 2^{MjE} \cdot Cje \cdot \left[Mje \cdot \frac{V(BI,EI)^2}{Vje} + (1-Mje) \cdot V(BI,EI) \right] \quad \forall\ V(BI,EI) > \frac{Vje}{2}$$

$$= Tf \cdot ICC + \frac{Cje \cdot Vje}{1-Mje} \left[1 - \left\{ 1 - \frac{V(BI,EI)}{Vje} \right\}^{1-Mje} \right] \quad \forall\ V(BI,EI) <= \frac{Vje}{2}$$

EDD blocks



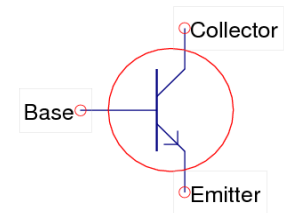
D6
I1=0
Q1=(V1 >= Vjc/2) ? Tr*V2+Cjc*con6*(Mjc*V1*V1/Vjc+con4*V1) : Tr*V2+Cjc*((Vjc/con4)*(1-(1-V1/Vjc)^con4))
I2=0



D7
I1=0
Q1=(V1>= Vje/2) ? Tf*V2+Cje*con5*(Mje*V1*V1/Vje+con3*V1) : Tf*V2+Cje*((Vje/con3)*(1-(1-V1/Vje)^con3))
I2=0

Equation

Eqn2
con3=1-Mje
con4=1-Mjc
con5=2^Mje
con6=2^Mjc

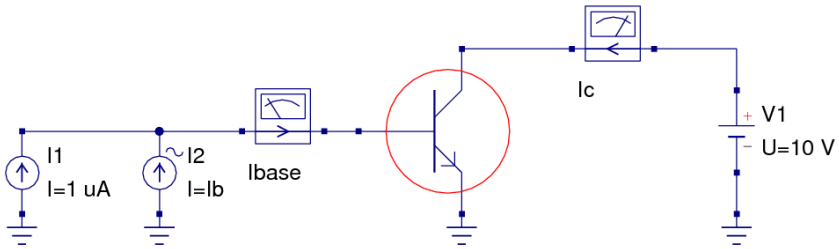


PTran
File=name
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Rc=2
Rb=100
Re=1
Ikf=0.5
Ikr=0.5
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 4 – simulating capacitive effects

AC gain



Parameter sweep

SW1
Sim=AC1
Type=list
Param=Ib
Values=[50u; 200u; 500u]

dc simulation

DC1

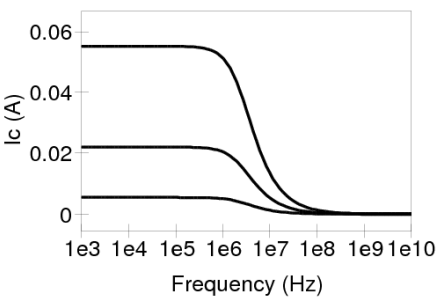
Equation

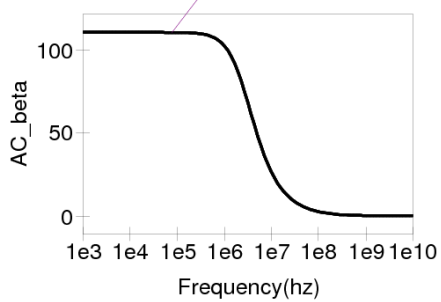
Eqn1
AC_beta=Ic.i/Ibase.i

ac simulation

AC1
Type=log
Start=1 kHz
Stop=10 GHz
Points=101

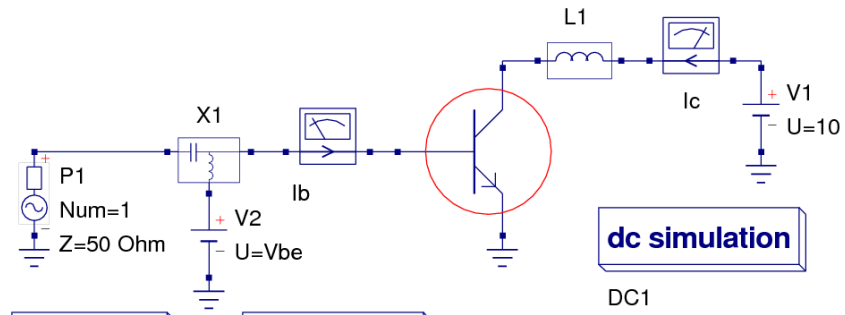
acfrequency: 7.76e+04
Ib: 5e-05
AC_beta: 110-j3.44





PTran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Rc=2
Rb=100
Re=1
Ikf=0.5
Ikr=0.5
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n

Cbe and Rbe extraction



Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Vbe
Start=-1
Stop=0.5
Points=71

S parameter simulation

SP1
Type=const
Values=[100kHz]

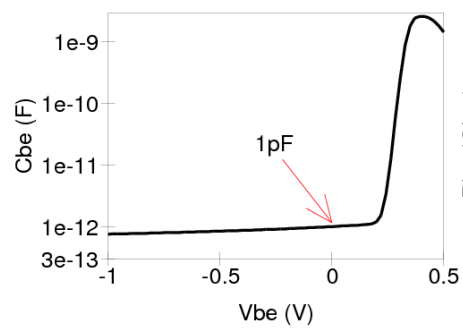
dc simulation

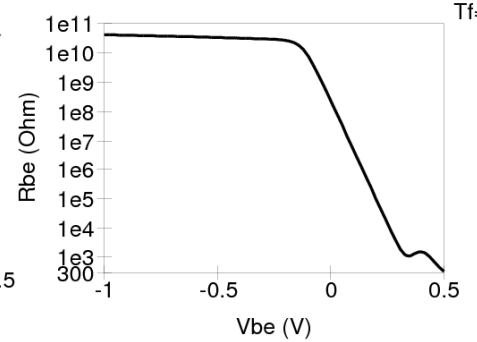
DC1

Equation

Eqn1
y=stoy(S)
PL_y11=PlotVs(y[1,1], Vbe)
Cbe=imag(y[1,1])/Omega
Rbe=1/real(y[1,1])
PL_Cbe=PlotVs(Cbe, Vbe)
PL_Rbe=PlotVs(Rbe, Vbe)
Omega=2*pi*frequency

PTran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Rc=2
Rb=100
Re=1
Ikf=0.5
Ikr=0.5
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n

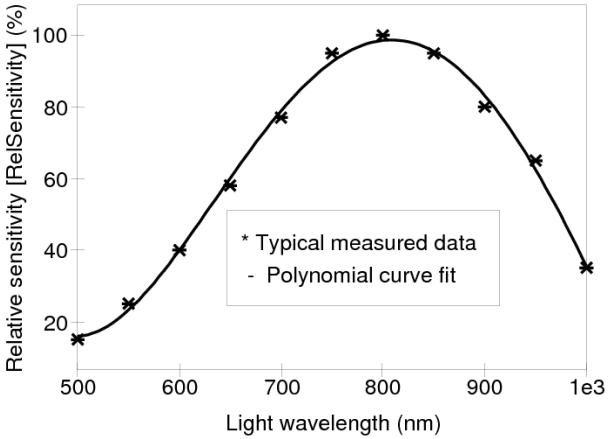




Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 5 – Adding photoelectric effects

Si phototransistor relative sensitivity data

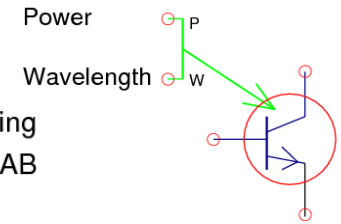


Measured data

Wavelength (nm)	Relative reponsivity (%)
500	15
550	25
600	40
650	58
700	77
750	95
800	100
850	95
900	80
950	65
1000	35

Curve fitting program

```
## Simple phototransistor responsivity fitting
## program - works with Octave or MATLAB
data = load('data.dat');
Ldata = data(:,1);
Sdata = data(:,2);
## Fit polynomial
Order = 4;
p = polyfit(Ldata, Sdata, Order);
## Evaluate the fitted polynomial
x=linspace(min(Ldata), max(Ldata), 101);
y = polyval(p,x);
## Plot
plot(x,y, '-', Ldata, Sdata,'x');
legend('Fitted polynomial', 'Original data');
```

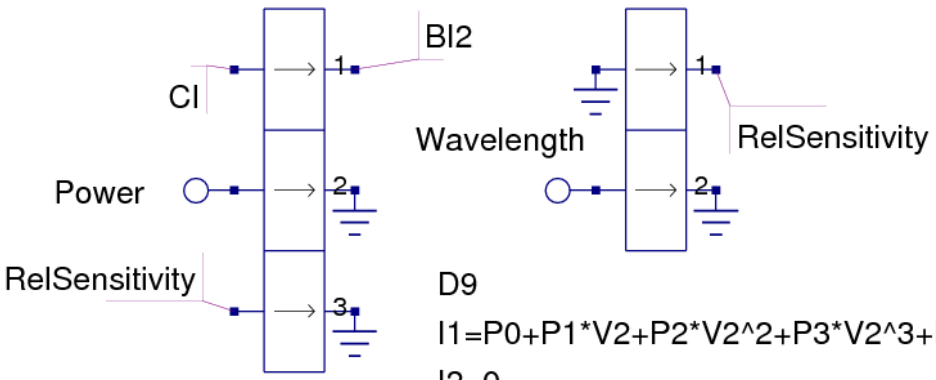



```
Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9
```

$$RelSensitivity = P_0 + P_1 \cdot \lambda + P_2 \cdot \lambda^2 + P_3 \cdot \lambda^3 + P_4 \cdot \lambda^4$$

$$Where\ P_0 = 2.6122e3, P_1 = -1.4893e1, P_2 = 3.0332e\text{-}2, P_3 = -2.5708e\text{-}5, P_4 = 7.69e\text{-}9$$

EDD light bus model



D9

I1=P0+P1*V2+P2*V2^2+P3*V2^3+P4*V2^4

I2=0

D8

I1=((Responsivity*V3)/(Bf*100))*V2

I2=0

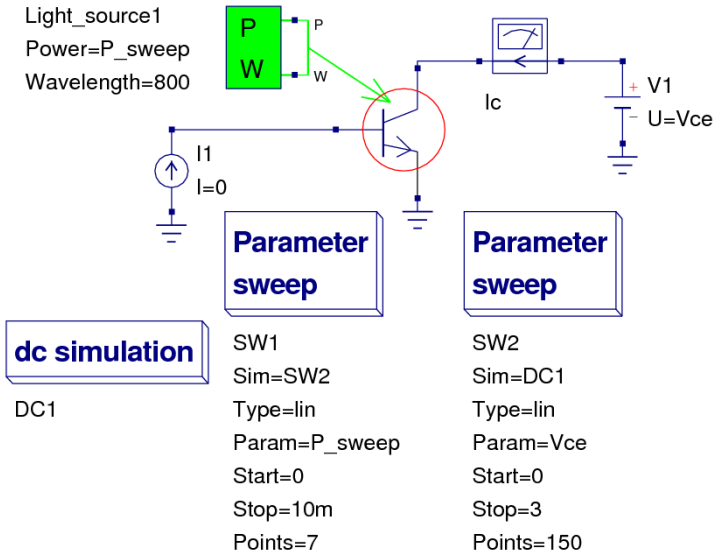
I3=V3

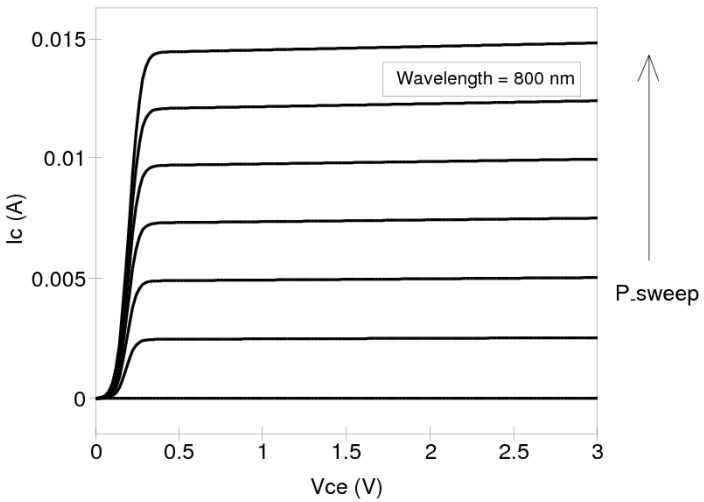
$$Iopt = \frac{RelSensitivity \cdot Responsivity \cdot power}{\beta_f \cdot 100}$$

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 6 – simulating photoelectric effects

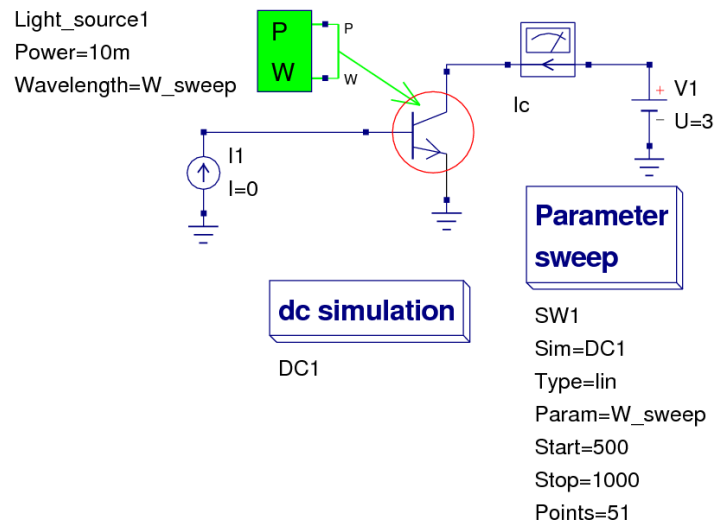
Phototransistor output characteristics

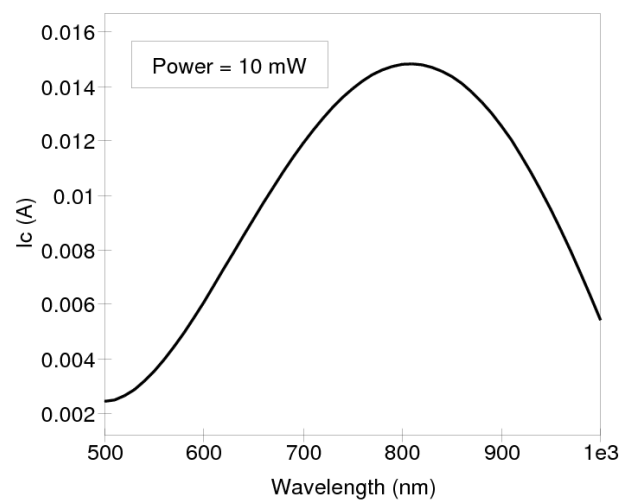




Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9

Phototransistor responsivity characteristics





Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 7 – adding noise to the phototransistor model

Qucs noise voltage source

V1
u=1e-6
e=0
c=1
a=0

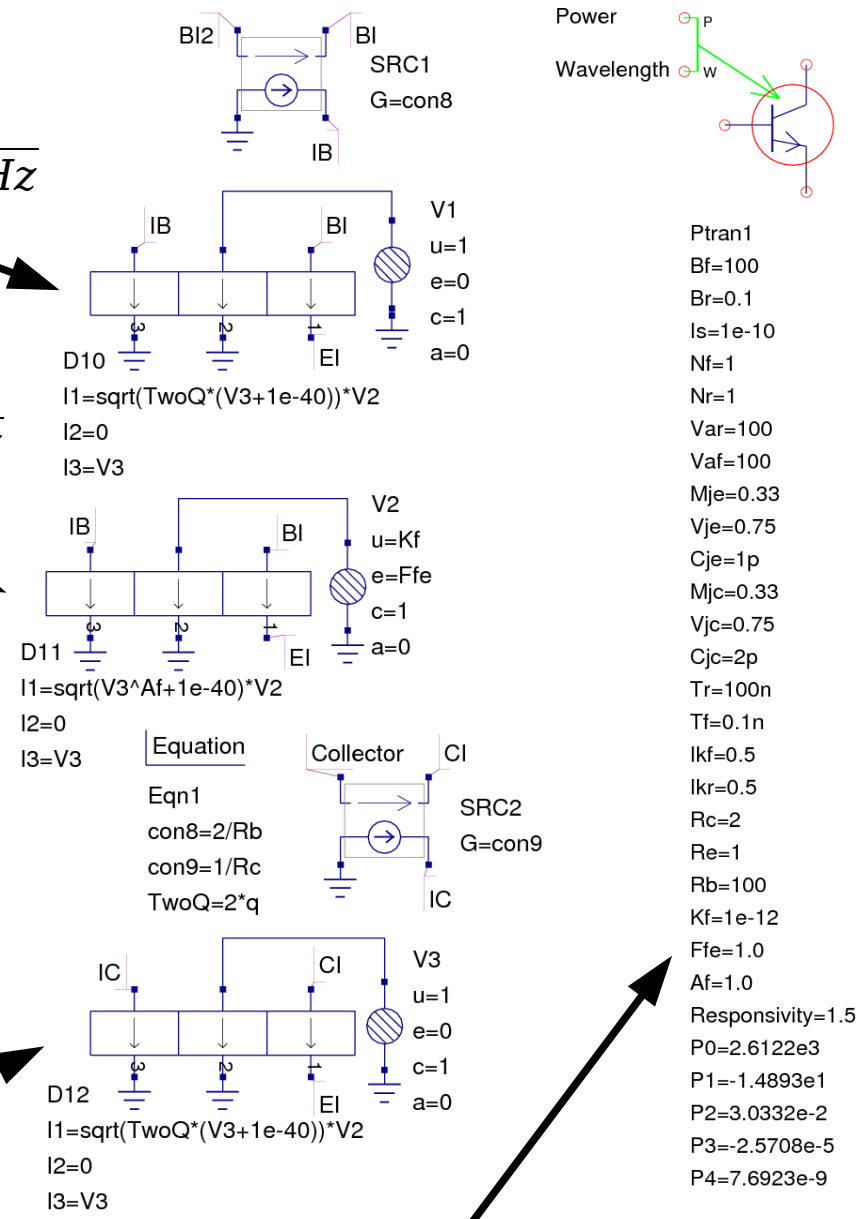
$$VPSD(f)=\frac{U}{a+c\cdot f^{e}}$$

Where *VPSD(f)* is the voltage spectral density at frequency *f* in V^2 / Hz, *U* is the voltage spectral density as *f* goes to zero, *a*, *c* and *e* are coefficients that determine the type of noise generated; white and shot noise with *U*=1, *e*=0, *c*=1 and *a*=0, and flicker noise with *U*=Kf, *e*=Ffe, *c*=1 and *a*=0.

$$ibsn=\sqrt{2\cdot q\cdot IB}\quad A/\sqrt{Hz}$$

$$ibfn=\sqrt{\frac{Kf^{Af}}{f^{Ffe}}}\quad A/\sqrt{Hz}$$

$$iICTsn=\sqrt{2\cdot q\cdot IC}\quad A/\sqrt{Hz}$$



Noise parameters

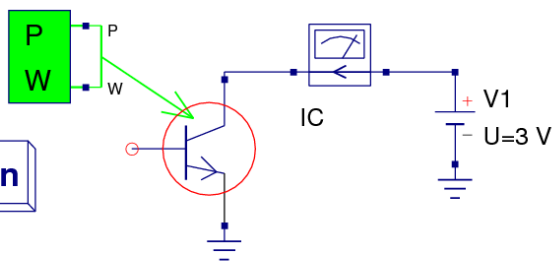
Note: Resistor thermal noise is automatically generated by Qucs

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 8 – simulating phototransistor noise

Variable light power

Light_source1
Power=sw_power
Wavelength=800



dc simulation

DC1

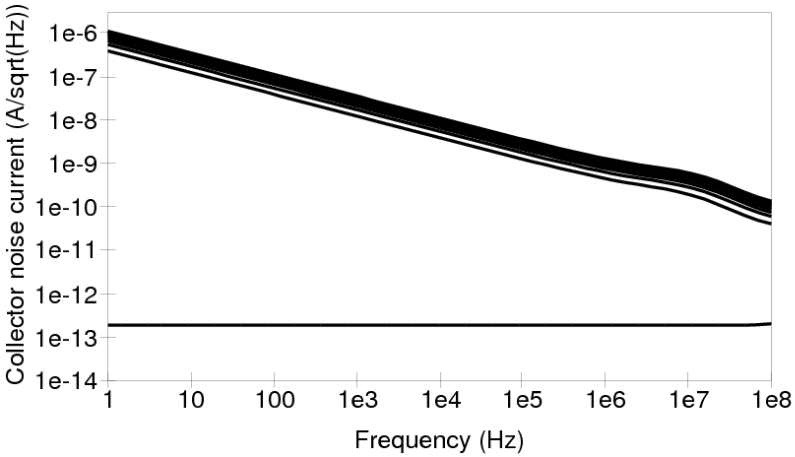
Parameter sweep

SW1
Sim=AC1
Type=lin
Param=sw_power
Start=0
Stop=20m
Points=11

ac simulation

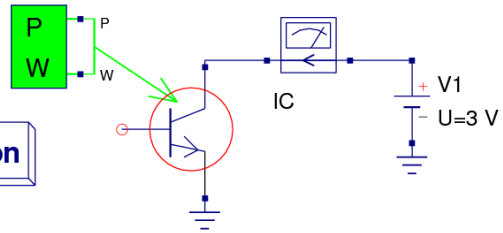
AC1
Type=log
Start=1 Hz
Stop=100MHz
Points=51
Noise=yes

Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9



Variable wavelength

Light_source1
Power=10m
Wavelength=sw_wavelength



dc simulation

DC1

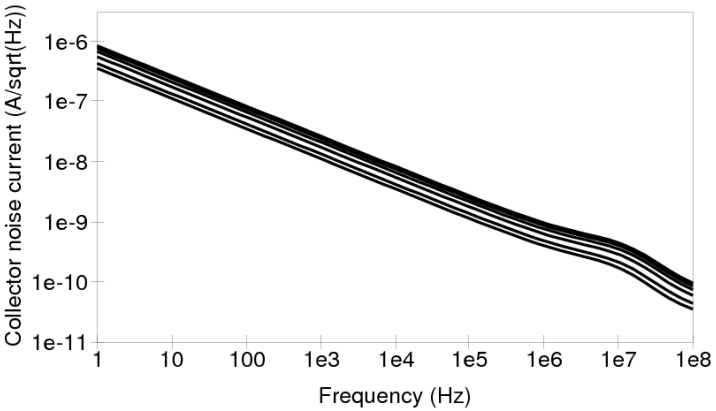
Parameter sweep

SW1
Sim=AC1
Type=lin
Param=sw_wavelength
Start=500
Stop=800
Points=7

ac simulation

AC1
Type=log
Start=1 Hz
Stop=100MHz
Points=51
Noise=yes

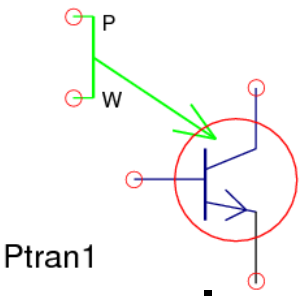
Ptran1
Bf=100
Br=0.1
Is=1e-10
Nf=1
Nr=1
Var=100
Vaf=100
Mje=0.33
Vje=0.75
Cje=1p
Mjc=0.33
Vjc=0.75
Cjc=2p
Tr=100n
Tf=0.1n
Ikf=0.5
Ikr=0.5
Rc=2
Re=1
Rb=100
Kf=1e-12
Ffe=1.0
Af=1.0
Responsivity=1.5
P0=2.6122e3
P1=-1.4893e1
P2=3.0332e-2
P3=-2.5708e-5
P4=7.6923e-9



Building a compact semiconductor model of an npn bipolar phototransistor

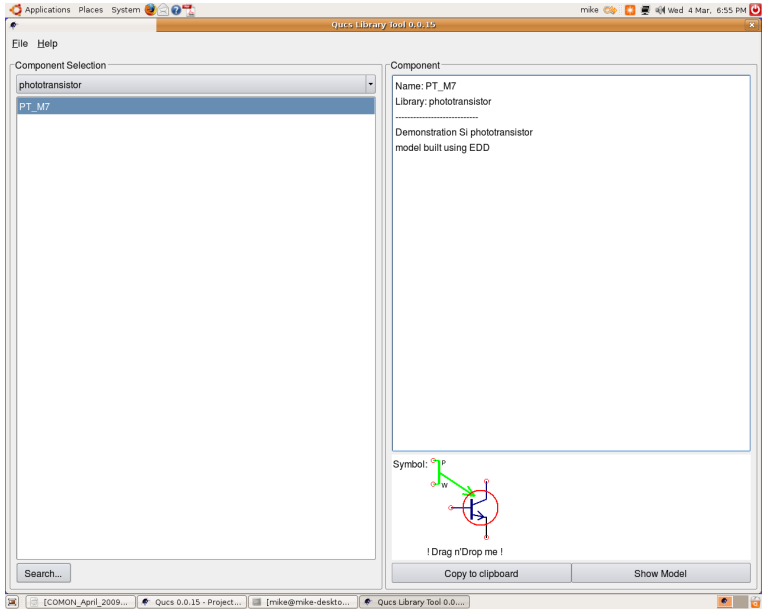
Construction: stage 9 – producing a distribution standard model

Model



Generate Verilog-A code

Store in Qucs library



Use “Create Library” from Project menu

Phototransistor.va file

```
`include "disciplines.vams"
`include "constants.vams"
module phototransistor (Collector, Base, Emitter, Power, Wavelength);
 inout Collector, Base, Emitter, Power, Wavelength;
 electrical Collector, Base, Emitter, Power, Wavelength;
 // Definition of internal local nodes
 electrical CI, BI, BI2, EI;
 // Parameter values and description text
 `define attr(txt) (*txt*)
 parameter real Bf=100 from [1:inf] `attr(info="forward beta");
 parameter real Br=0.1 from [1e-6:inf] `attr(info="reverse beta");
                                        :
 parameter real P3=-2.5708e-5 from[-inf:inf]
    `attr(info="relative selectivity polynomial coefficient" );
 parameter real P4=7.6923e-9  from[-inf:inf]
    `attr(info="relative selectivity polynomial coefficient" );
 // Definition of internal variables and quantities
 real VT, con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, TwoQ, FourKT,
 GMIN;
 real ICC, IEC, q1, q2, IB, IC, IE, Q1, RelSensitivity;                    // Quantities
 analog begin
 // Module initialisation code
 @(initial_model)
  begin
   VT = `P_K*300/`P_Q; con1=1/(Nf*VT); con2=1/(Nr*VT); con3=1-Mje; // VT = vt(300)
   con4=1-Mjc, con5=pow(2, Mje); con6=pow(2, Mjc); con7=Rb/2; con8=2/Rb;
   con9=1/Rc; con10=1/Re; TwoQ=2*`P_Q;   GMIN=1e-12;                  // TwoQ = 2*q
   FourKT=4*`P_K*$temperature;                                        // FourKT = 4*K*T
  end;
  // Model quantity equations and current contributions
  ICC=Is*(limexp(V(BI,EI)*con1)-1);   IEC=Is*(limexp(V(BI,CI)*con2)-1);
  q1=1+V(BI,CI)/Vaf + V(BI,EI)/Var;   q2=(ICC/Ikf) + (IEC/Ikr);  IB=V(BI2,BI)*con8;
  IC=V(Collector,CI)*con9;   IE=V(EI,Emitter)*con10;    I(Collector,CI) <+ IC;
  I(Base,BI2) <+ V(Base, BI2)*con8;    I(BI2, BI) <+ IB;    I(EI, Emitter) <+ IE;
  I(BI,CI) <+ (IEC/Br) + GMIN*V(BI,CI);    I(BI,EI) <+ (ICC/Bf) + GMIN*V(BI,EI);
  I(CI,EI) <+ (ICC-IEC)/(1e-20+(q1/2)*(1+sqrt(1+4*q2)));
  Q1=(V(BI,CI) >Vjc/2) ? Tr*IEC+Cjc*con6*(Mjc*V(BI,CI)*V(BI,CI)/Vjc+con4*V(BI,CI))
                       : Tr*IEC+Cjc*((Vjc/con4)*(1-pow((1-V(BI,CI)/Vjc),con4)));
  I(BI,CI) <+ ddt(Q1);
  Q1=(V(BI,EI) >Vje/2) ? Tf*ICC+Cje*con5*(Mje*V(BI,EI)*V(BI,EI)/Vje+con3*V(BI,EI))
                       : Tf*ICC+Cje*((Vje/con3)*(1-pow((1-V(BI,EI)/Vje),con3)));
  I(BI,EI) <+ ddt(Q1);
  RelSensitivity = P0+P1*V(Wavelength)+P2*pow(V(Wavelength),2)
                 +P3*pow(V(Wavelength),3)+P4*pow(V(Wavelength),4);
  I(CI,BI2) <+ ( (Responsivity*RelSensitivity)/(Bf*100) )*V(Power);
  // Noise contributions
  I(Collector,CI)   <+  white_noise(FourKT*con9, "thermal");
  I(Base,BI2)       <+  white_noise(FourKT*con8, "thermal");
  I(BI2,BI)         <+  white_noise(FourKT*con8, "thermal");
  I(EI,Emitter)     <+  white_noise(FourKT*con10, "thermal");
  I(CI,EI)          <+  white_noise(TwoQ*IC, "shot");
  I(BI,EI)          <+  white_noise(TwoQ*IB, "shot");
  I(BI,EI)          <+  flicker_noise(Kf*pow(IB, Af), Ffe, "flicker");
 end
endmodule
```

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 10 – Compiling Verilog-A code to C++

Compile phototransitor.va file

with ADMS* using

1. admsXml phototransistor.va -e qucsVersion.xml -e qucsMODULEcore.xml
   : generates files phototransistor.core.cpp and phototransistor.core.h

2. admsXml phototransistor.va -e qucsVersiom.xml -e qucsMODULEdefs.xml
   : generates file phototransistor.defs.h

3. admsXml phototransistor.va -e qucsVersion.xml -e qucsMODULEgui.xml
   : generates files phototransistor.gui.cpp phototransistor.gui.h

4. admsXml phototransistor.va -e analogfunction.xml
   : generates files phototransistor.analogfunction.cpp and
   phototransistor. analogfunction.h

File phototransistor.va and the generated compiled files must be in Qucs directory

qucs-core/src/components/verilog

* L. Lemaitre and B. Gu, “ADMS-a fully customizable Verilog-AMS compiler approach”, 2006.
Available from http://www.mos-ak.org/montreux/posters/17_Lemaitre_MOS-AK06.pdf .
S. Jahn and H. Parruitte, “Qucs: A description, Verilog-AMS interface, 2006.
Available from: http://qucs.sourceforge.net/docs.html .

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 11 – Merging phototransistor C++ code with the Qucsator analogue circuit simulator code

11.1 Modify Makefile.am in directory qucs-core/src/components/Verilog

(a) Add phototransitor.analogfunction.cpp and phototransistor.core.cpp to entry libverilog_a_SOURCES

(b) Add phototransistor.analogfuncction.h, phototransistors.defs.h and phototransistor.core.h to entry noinst_HEADERS

(c) Add phototransistor.va to entry VERILOG_FILES

11.2 Add #include "verilog/phototransistor.core.h" to file qucs-core/src/components/components.h

11.3 Add under heading // circuit components

REGISTER_CIRCUIT(phototransistor)

to file qucs-core/src/module.cpp

11.4 Add under heading // verilog devices (this step is optional but recommended)

CIR_phototransistor

to file qucs-core/src/components/component_id.h

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 12 – Adding the phototransistor to the Qucs GUI code: part 1

12.1 Add to array pInfoFunc VerilogAComps[ ] in file qucs/qucs/qucs.cpp entry &phototransistor::info before 0 at and of list

12.2 Rename phototransistor.gui.cpp to phototransistor.cpp and phototransistor.gui.h to phototransistor.h, then copy renamed files to directory qucs/qucs/components

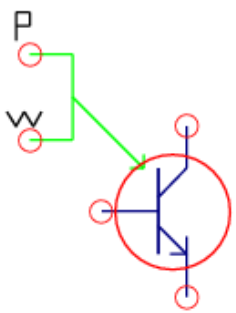
12.3 Edit phototransistor.cpp to include GUI drawing information

```
#include "phototransistor.h"
phototransistor::phototransistor()
{
  Description = QObject::tr ("phototransistor verilog device");
  Props.append (new Property ("Bf", "100", false, QObject::tr ("forward beta")));
  Props.append (new Property ("Br", "0.1", false, QObject::tr ("reverse beta")));
  Props.append (new Property ("Is", "1e-10", false,  QObject::tr ("dark current")+" ("+QObject::tr ("A")+")"));
                                                  ..
  Props.append (new Property ("Temp", "26.85", false,QObject::tr ("simulation temperature")));
  createSymbol ();
  tx = x2 + 4;
  ty = y1 + 4;
  Model = "phototransistor";
  Name  = "PT";
}
Component * phototransistor::newOne()  {
  phototransistor * p = new phototransistor();
  p->Props.getFirst()->Value = Props.getFirst()->Value;
  p->recreate(0);
  return p;
}
Element * phototransistor::info(QString& Name, char * &BitmapFile, bool getNewOne)  {
  Name = QObject::tr("photo transistor");
  BitmapFile = (char *) "phototransistor";
  if(getNewOne) return new phototransistor();
  return 0;
}
```

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 12 – Adding the phototransistor to the Qucs GUI code: part 2

Editing phototransistor.cpp continued



PT1

```
void phototransistor::createSymbol()
{
 Arcs.append(new Arc(-25,-20, 40, 40,  0,20*360,QPen(QPen::red,2)));
 Lines.append(new Line(-10,-15,-10, 15,QPen(QPen::darkBlue,3)));
 Lines.append(new Line(-30,  0,-10,  0,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(-10, -5,  0,-15,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(  0,-15,  0,-30,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(-10,  5,  0, 15,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(  0, 15,  0, 30,QPen(QPen::darkBlue,2)));
 Lines.append(new Line( -6, 15,  0, 15,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(  0,  9,  0, 15,QPen(QPen::darkBlue,2)));
 Lines.append(new Line(-55, -55, -40, -55,QPen(QPen::green,2)));
 Lines.append(new Line(-40, -55, -40, -25,QPen(QPen::green,2)));
 Lines.append(new Line(-40, -25, -55, -25,QPen(QPen::green,2)));
 Lines.append(new Line(-40, -40, -15, -15,QPen(QPen::green,2)));
 Lines.append(new Line(-15, -15, -15, -20,QPen(QPen::green,2)));
 Lines.append(new Line(-15, -15, -20, -15,QPen(QPen::green,2)));
 Lines.append(new Line(-60, -60, -60, -70,QPen(QPen::black,2)));
 Lines.append(new Line(-60, -70, -55, -70,QPen(QPen::black,2)));
 Lines.append(new Line(-55, -70, -55, -65,QPen(QPen::black,2)));
 Lines.append(new Line(-60, -65, -55, -65,QPen(QPen::black,2)));
 Lines.append(new Line(-63, -35, -60, -30,QPen(QPen::black,2)));
 Lines.append(new Line(-60, -30, -57, -35,QPen(QPen::black,2)));
 Lines.append(new Line(-57, -35, -54, -30,QPen(QPen::black,2)));
 Lines.append(new Line(-54, -30, -51, -35,QPen(QPen::black,2)));
 Ports.append(new Port(0,  -30));      // Collector
 Ports.append(new Port(-30, 0));       // Base
 Ports.append(new Port(  0, 30));      // Emitter
 Ports.append(new Port(-55,-55));      // Power
 Ports.append(new Port(-55,-25));      // Wavelength
 x1 = -35; y1 = -55;
 x2 =  35; y2 =  50;
}
```

Symbol data for schematic capture

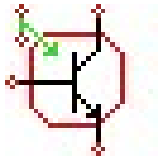
Symbol outline

Port connections

Coordinates marking area of box surrounding symbol

Building a compact semiconductor model of an npn bipolar phototransistor

Construction: stage 12 – Adding the phototransistor to the Qucs graphical user interface code: part 3



phototransistor.png

12.3 (a) Draw a 32 bit x 32 bit icon picture for the phototransistor and add it to directory qucs/qucs/bitmaps

(b) Add phototransistor.png to XPMS entry in file qucs/qucs/bitmaps/Makefile.am

12.4 Add to file qucs/qucs/components/component.cpp

(in getComponentFromName, in switch (first) statement)

as part of case 'p'

else if (cstr == “hototransistor”) c= new phototransistor();

12.5 Modify Makefile.am in directory qucs/qucs/components

(a) Add phototransistor.cpp to entry libverilog_a_SOURCES

(b) Add phototransistor.h to entry noinst_HEADERS

12.6 Add #include “phototransistor.h”

to file qucs/qucs/components/components.h

FINAL STEP → Compile qucs and qucs-core → Run qucs and test new phototransistor model

Summary

1. Qucs is a freely available circuit simulator distributed as open source software under the GNU/Linux General Public Licence (GPL).

2. This demonstration has attempted to outline the fundamental features of the package, its available components, libraries, built-in design aids, and analysis types.

3. The demonstration also introduces a number of basic approaches to circuit simulation using Qucs.

4. The presentation also showed how the compact semiconductor modeling and circuit macromodeling features implemented in the current Qucs release can be used to develop equation-defined component models of established and emerging technology devices.

5. The latter sections of the demonstration concentrated on using ADMS to compile Verilog-A compact device models and the steps needed to link such models with the main body of the Qucs C++ code.