Qucs: improvements and new directions in GPL compact device modelling and circuit simulation tools

Mike Brinson, Centre for Communications Technology, London Metropolitan University, UK, mbrin72043@yahoo.co.uk.
Richard Crozier, The University of Edinburgh, UK, richard.crozier@yahoo.co.uk.
Vadim Kuznetsov, Bauman Moscow Technical University, Russia, ra3xdh@gmail.com.
Clemens Novak, Qucs Developer, clemens@familie-novak.net.
Bastien Roucaries, Laboratoire SATIE – CNRS UMR 8929, Université de Cergy-Pontoise, ENS Cachan, FR, bastien.roucaries@satie.ens-cauchan.fr.
Frans Schreuder, Nikhef, Amsterdam, NL, fransschreuder@gmail.com.
Guilherme Brondani Torri, Qucs Developer, guitorri@gmail.com.

Plus contributions from members of the Qucs user community.

- From Qucs-0.0.18 to Qucs-0.0.19: the way forward
- Octave support for Qucs circuit simulation
- Expanded compact device modelling capabilities with the Berkeley Model and Algorithm Prototyping Platform (MAPP)
- Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
- Linking ngspice Xyce to Qucs
- New circuit design aids
- Improvements to Qucs documentation and code control features
- Possible directions for Qucs development after release 0.0.19

Presented at the MOS-AK Spring Workshop at DATE, Grenoble, France, March 12, 2015.
From Qucs-0.0.18 to Qucs-0.0.19: the way forward
• Part 1 - Qucs-0.0.18

GPL software used by Qucs-0.0.18
** PS2SP – PSPICE to SPICE preprocessor, http://members.acon.at/fschmid7/
# From Qucs-0.0.18 to Qucs-0.0.19: the way forward

- Part 2 - Qucs-0.0.18 download statistics

## Qucs-0.0.18 Download statistics between 1 August 2014 to 15 Jan 2015

<table>
<thead>
<tr>
<th>Downloads</th>
<th>74,922</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Top Country</strong></td>
<td>United States 15%</td>
</tr>
<tr>
<td><strong>Top OS Windows</strong></td>
<td>Windows 78%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Android</th>
<th>BSD</th>
<th>Linux</th>
<th>Macintosh</th>
<th>Solaris</th>
<th>Unknown</th>
<th>Windows</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>272</td>
<td>13</td>
<td>520</td>
<td>1,392</td>
<td>0</td>
<td>1,572</td>
<td>7,741</td>
<td>11,510</td>
</tr>
<tr>
<td>India</td>
<td>172</td>
<td>1</td>
<td>445</td>
<td>89</td>
<td>0</td>
<td>322</td>
<td>4,189</td>
<td>5,218</td>
</tr>
<tr>
<td>Germany</td>
<td>23</td>
<td>4</td>
<td>349</td>
<td>443</td>
<td>0</td>
<td>1,784</td>
<td>2,520</td>
<td>5,123</td>
</tr>
<tr>
<td>France</td>
<td>21</td>
<td>7</td>
<td>178</td>
<td>223</td>
<td>0</td>
<td>26</td>
<td>2,929</td>
<td>3,384</td>
</tr>
<tr>
<td>Spain</td>
<td>19</td>
<td>0</td>
<td>204</td>
<td>347</td>
<td>0</td>
<td>204</td>
<td>2,398</td>
<td>3,172</td>
</tr>
<tr>
<td>Russia</td>
<td>34</td>
<td>3</td>
<td>182</td>
<td>130</td>
<td>0</td>
<td>115</td>
<td>2,624</td>
<td>3,088</td>
</tr>
<tr>
<td>Italy</td>
<td>34</td>
<td>0</td>
<td>287</td>
<td>281</td>
<td>0</td>
<td>14</td>
<td>2,405</td>
<td>3,021</td>
</tr>
<tr>
<td>Japan</td>
<td>12</td>
<td>8</td>
<td>73</td>
<td>184</td>
<td>0</td>
<td>19</td>
<td>2,439</td>
<td>2,735</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>28</td>
<td>0</td>
<td>112</td>
<td>258</td>
<td>4</td>
<td>122</td>
<td>1,973</td>
<td>2,497</td>
</tr>
<tr>
<td>Brazil</td>
<td>21</td>
<td>1</td>
<td>155</td>
<td>91</td>
<td>0</td>
<td>33</td>
<td>1,634</td>
<td>1,935</td>
</tr>
<tr>
<td>Mexico</td>
<td>9</td>
<td>0</td>
<td>92</td>
<td>333</td>
<td>0</td>
<td>6</td>
<td>1,402</td>
<td>1,842</td>
</tr>
<tr>
<td>Canada</td>
<td>7</td>
<td>2</td>
<td>65</td>
<td>240</td>
<td>0</td>
<td>15</td>
<td>1,380</td>
<td>1,709</td>
</tr>
<tr>
<td>Argentina</td>
<td>3</td>
<td>0</td>
<td>77</td>
<td>38</td>
<td>0</td>
<td>9</td>
<td>1,502</td>
<td>1,629</td>
</tr>
<tr>
<td>Poland</td>
<td>9</td>
<td>6</td>
<td>73</td>
<td>40</td>
<td>0</td>
<td>5</td>
<td>1,361</td>
<td>1,494</td>
</tr>
<tr>
<td>Colombia</td>
<td>6</td>
<td>0</td>
<td>55</td>
<td>116</td>
<td>0</td>
<td>7</td>
<td>1,020</td>
<td>1,204</td>
</tr>
<tr>
<td>Indonesia</td>
<td>11</td>
<td>0</td>
<td>49</td>
<td>32</td>
<td>0</td>
<td>9</td>
<td>971</td>
<td>1,072</td>
</tr>
<tr>
<td>Australia</td>
<td>9</td>
<td>4</td>
<td>44</td>
<td>138</td>
<td>0</td>
<td>6</td>
<td>843</td>
<td>1,044</td>
</tr>
<tr>
<td>Portugal</td>
<td>8</td>
<td>0</td>
<td>30</td>
<td>77</td>
<td>0</td>
<td>3</td>
<td>910</td>
<td>1,028</td>
</tr>
<tr>
<td>Netherlands</td>
<td>6</td>
<td>3</td>
<td>55</td>
<td>105</td>
<td>0</td>
<td>111</td>
<td>737</td>
<td>1,017</td>
</tr>
<tr>
<td>South Africa</td>
<td>18</td>
<td>0</td>
<td>23</td>
<td>31</td>
<td>0</td>
<td>22</td>
<td>905</td>
<td>999</td>
</tr>
</tbody>
</table>
From Qucs-0.0.18 to Qucs-0.0.19: the way forward

- Part 3 - Introducing the Qucs-0.0.19 structure

Qucs models and modelling features

- Equation-defined device models
- Components Subcircuits Macromodels
- Library components
- Circuit design data
- SPICE netlist models

Qucs algebraic equations

Qucs GUI Schematic capture

Simulation Qucsator

Simulation ngspsie

Simulation Xyce

OCTAVE

Qucs post-simulation data processing

Output data plots and tables

OCTAVE post-simulation data processing

Numerical post-simulation data processing

MAPP post-simulation data processing

MAPP* Berkeley Model and Algorithm Prototyping Platform (MAPP)

http://draco.eecs.berkeley.edu:8765/jr/MAPP/wikis/home

Numerical post-simulation data processing + MAPP device modelling and simulation
Octave support for Qucs circuit simulation
Part 1 – post simulation data processing

Octave
Post-simulation data processing

Qucs
Post-simulation data processing
Octave support for Qucs circuit simulation
Part 2 – Qucs-Octave linked transient simulation

Qucs netlist: boostconverter.net

Boost Converter

Branch Currents

externally driven transient simulation

ETR1

ECVS1
U=0 V
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 1: Launching MAPP from Qucs

Background to MAPP:
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 2: Qucs/MAPP output

Qucs circuit schematic and output data

Background to MAPP:
http://draco.eecs.berkeley.edu/dracotiki/tiki-index.php?page=MAPPfeatures
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 3: From Qucs circuit synthesis to MAPP output

---

This is the Berkeley Model and Algorithm Prototyping Platform (MAPP)
- git branch 2014-12-10--alpha-release

---
Qucs evaluation of tunnel diode model equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parameters</td>
<td>Model equations</td>
</tr>
<tr>
<td>Vv=0.4</td>
<td>Vdiode=ls*(exp(Vpn/VT)-1.0)</td>
</tr>
<tr>
<td>VT=0.025</td>
<td>lexcess=lv<em>exp(K</em>(Vpn-VV))</td>
</tr>
<tr>
<td>Ip=1e-5</td>
<td>Ip=Vdiode+lexcess+ltunnel</td>
</tr>
<tr>
<td>Vp=0.1</td>
<td>ltunnel=lp*(Vpn/Vp)*exp((Vp-Vp)/Vp)</td>
</tr>
<tr>
<td>lv=1e-6</td>
<td></td>
</tr>
<tr>
<td>K=5</td>
<td></td>
</tr>
<tr>
<td>ls=1e-12</td>
<td></td>
</tr>
</tbody>
</table>

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 4: Compact device modelling; the tunnel diode model continued

MAPP tunnel diode compact device model

function MOD = tunnelDiode_ModSpec_wrapper()
    MOD = ee_model();
    MOD = add_to_ee_model(MOD, 'external_nodes', {'p', 'n'});
    MOD = add_to_ee_model(MOD, 'explicit_outs', {'ipn'});
    MOD = add_to_ee_model(MOD, 'parms', {'Is', 1e-12, 'VT', 0.025});
    MOD = add_to_ee_model(MOD, 'parms', {'Ip', 1e-5, 'Vp', 0.1});
    MOD = add_to_ee_model(MOD, 'parms', {'Iv', 1e-6, 'Vv', 0.4, 'K', 5});
    MOD = add_to_ee_model(MOD, 'parms', {'C', 0});
    MOD = add_to_ee_model(MOD, 'f', @f);
    MOD = add_to_ee_model(MOD, 'q', @q);
    MOD = finish_ee_model(MOD);
end

function out = f(S)
    v2struct(S);
    I_diode = Is*(exp(vpn/VT)-1);
    I_excess = Iv * exp(K * (vpn - Vv));
    I_tunnel = (Ip/Vp) * vpn * exp(-1/Vp * (vpn - Vp));
    out = I_diode + I_tunnel + I_excess;
end

function out = q(S)
    v2struct(S);
    out = C*vpn;
end

Tunnel diode I/V curve

MAPP post-simulation output

Figure 1
I/V curve of a tunnel diode
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 5: Compact device modelling; the tunnel diode model continued

function out = plotGV_tunnelDiode_ModSpec_wrapper()
    MOD = tunnelDiode_ModSpec_wrapper();
    vs = -0.05:0.001:0.4;
    gs = zeros(size(vs));
    for idx = 1:size(gs,2)
        gs(1,idx) = MOD.dfe_dvecX(vs(idx), [], [], [], MOD);
    end
    figure();
    plot([min(vs), max(vs)], [0, 0], 'Color', 'red', 'LineWidth', 1.25, 'LineStyle', '--');
    hold on;
    plot([0, 0], [min(gs), max(gs)], 'Color', 'red', 'LineWidth', 1.25, 'LineStyle', '--');
    h = plot(vs, gs, 'Color', 'blue', 'LineWidth', 1.75);
    axis tight;
    box on;
    grid on;
    set(gca,'FontName','Times New Roman','FontSize',15,'FontWeight','bold');
    xlabel('vpn (V)','FontName','Times New Roman','FontSize',18,'FontWeight','bold');
    ylabel('transconductance G (S)','FontName','Times New Roman','FontSize',18,'FontWeight','bold');
    title(['G/V curve of a tunnel diode'],'FontName','Times New Roman', 'FontSize',18,'FontWeight','bold');
    set(gcf,'color','white');
end
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 6: Tunnel diode EDD model and test circuit

Equation

Model_parameters
Wv=0.4
VT=0.025
Ip=1e-5
Vp=0.1
Iv=1e-6
K=5
Is=1e-12

Equation

Model_equations
I_{diode}=I_s*(\exp(V_{pn}/VT)-1.0)
I_{excess}=I_v*\exp(K*(V_{pn}-V_v))
I_{pn}=I_{diode}+I_{excess}+I_{tunnel}
I_{tunnel}=I_p^*(V_{pn}/V_p)*\exp((V_p-V_{pn})/V_p)

\[\begin{align*}
P1 & \quad TD\_EDD1 \\
V_T &= 0.025 \\
I_s &= 1e-12 \\
I_p &= 1e-5 \\
I_v &= 1e-6 \\
V_p &= 0.1 \\
V_v &= 0.4 \\
K &= 5 \\
C &= 0.01p \\
\end{align*}\]
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 7: Tunnel diode Verilog-A model and test circuit

// Tunnel diode Verilog-A compact device model.
// Verilog-A code translated, by hand, from
// MAPP model.
//
// `include "disciplines.vams"
// `include "constants.vams"
//
module TD(p,n);
inout p,n;
electrical p,n;
//
parameter Is=1e-12 from [1e-20 : inf);
parameter Ip=1e-5 from [1e-20 : inf);
parameter Iv=1e-6 from [1e-20 : inf);
parameter Vp=0.1 from [1e-20 : inf);
parameter Vv=0.4 from [1e-20 : inf);
parameter KTD = 5 from [1e-20 : inf);
parameter VT=0.025 from [1e-20 : inf);
parameter C=1e-15 from [1e-20 : inf);
//
real Idiode, Iexcess, Itunnel;
//
analog begin
Idiode = Is*(exp(V(p,n)/VT)-1.0);
Iexcess = Iv*exp(KTD*(V(p,n)-Vv));
Itunnel = Ip*(V(p,n)/Vp)*exp(Vp-V(p,n)/Vp);
I(p,n) <+ Idiode+Iexcess+Itunnel;
I(p,n) <+ ddt(C*V(p,n));
end
endmodule
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools

Part 8: Tunnel diode SPICE B model and test circuit

Typical ngspice, Xyce and LTspice tunnel diode netlist constructed with a subcircuit and B type non-linear current sources

The tunnel diode conductance = \( \text{diff}(I(V_m), V_{dc}) \).

SPICE equivalent of diff appears not to be implemented?
Linking ngspice and Xyce to Qucs: Part 1 - Background

Concept

The primary purpose of the proposed integration is (1) to provide a facility which allows Qucs schematics to be simulated with GPL SPICE compatible simulation engines, in particular ngspice (http://ngspice.sourceforge.net) and Xyce (https://xyce.sandia.gov/), and (2) to check compact device model performance and accuracy across different simulators.

Current state of Work:

- Implemented Ngspice/Xyce netlist builder
- Implemented Ngspice/Xyce simulation output to Qucs XML-dataset converter
- Supported components:
  - RCL components
  - Nonlinear devices (Diode, BJT, MOSFET, JFET)
  - AC, DC, pulse voltage sources
  - Controlled sources
  - Relay
- Supported simulations
  - DC simulation
  - Transient simulation
  - AC simulation
  - Harmonic Balance (Xyce)
- Qucs library components support (ngspice)

Source code available from https://github.com/ra3xdh/qucs/tree/spice4qucs
Linking ngspice and Xyce to Qucs: Part 2 - A simple RC circuit example

Qucs menu Simulate (F2)

Simulate with SPICE (ngspice or Xyce)

**Diagram:**
- Circuit diagram with labels:
  - L1 = 10u
  - C1 = 40p
  - R1 = 30

**Qucs Simulation Console:**
- Circuit: *qucs 0.0.19
- /home/wik/qucs/lcr-1.dpl
- Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
- ngspice-26 done

**Graph Input:**
- Graph input dialog with dataset and options

**Graph Output:**
- Graph showing AC and transient simulation results

**AC Simulation:**
- AC1
- Type = lin
- Start = 1 MHz
- Stop = 10 MHz
- Points = 1000

**Transient Simulation:**
- TR1
- Type = lin
- Start = 0
- Stop = 1 us
Linking ngspice and Xyce to Qucs: Part 3 - A larger circuit example

Audio amplifier schematic

AC gain against frequency

Transient output voltage against time

Ngspice and Xyce audio amplifier test circuit and typical simulation results
Linking ngspice and Xyce to Qucs: Part 4 – Comparing Qucs and Xyce Harmonic Balance simulation

**Harmonic balance simulation**

HB1
f=1 MHz
n=17
iabstol=1 pA
vabstol=1 uV
reltol=0.001

---

**Xyce**

---

**Qucs**
Linking ngspice and Xyce to Qucs: Part 4 – Small signal S parameter simulation with ngspice and Qucs

DC Simulation:
- SRC1: V1, U=1 V, R5 = 50 Ω
- R1: R = 50 Ω
- U0: U = 1 V, f = 1 kHz

AC Simulation:
- AC1: Type = log, Start = 10 kHz, Stop = 100 MHz, Points = 161

DUT:
- IN to OUT
- L2: L = 15.92 μH
- C3: C = 3.183 nF
- C4: C = 3.183 nF
- R2: R = 50 Ω
- R6: R = 50 Ω
- SRC2: G = 2

DUT:
- IN to OUT
- L3: L = 15.92 μH
- C5: C = 3.183 nF
- C6: C = 3.183 nF
- R4: R = 50 Ω
- U1: U = 1 V, f = 1 kHz
- SRC3: G = 2

DUT:
- IN to OUT
- V2: V = 1 V
- U2: U = 1 V, R = 50 Ω
- SRC4: G = 2
Linking ngspice and Xyce to Qucs: Part 4 – Small signal S parameter simulation with ngspice and Qucs continued
Resistor colour codes

Main features
- Main window of the Qucs-active filter design tool.
- Butterworth, Chebyshev (Type I and II), Bessel, and Cauer low-pass, high-pass, band-pass, and band-stop active filter design capabilities.
- Sallen-Key, Multifeedback and Cauer filter section topologies are available.
- User-defined filter transfer functions.
- Full Qucs integration, including a copy-and-paste interface.
- To be released with Qucs-0.0.19.
Butterworth Low Pass Active filter design

**Filter parameters**
- Passband attenuation, $A_p$ (dB): 3
- Stopband attenuation, $A_s$ (dB): 10
- Cutoff frequency, $f_c$ (Hz): 1000
- Stopband frequency, $f_s$ (Hz): 3000
- Passband ripple $R_p$ (dB): 0
- Passband gain, $K_v$ (dB): 0
- Filter order: 5

**Transfer function and Topology**
- Approximation type: Butterworth
- Filter type: LowPass
- Filter topology: Sallen-Key (S-K)

**Filter calculation console**
- Filter order: 2
- Poles list: $P_k = e^{j\phi} \cdot \text{Re}^{\frac{j\pi}{n}}$
  - $-0.707107 + j0.707107$
  - $-0.707107 + j-0.707107$
- Part list: $C_1$ (mF), $C_2$ (mF), $R_1$ ($k\Omega$), $R_2$ ($k\Omega$), $R_3$ ($k\Omega$), $R_4$ ($k\Omega$), $R_5$ ($k\Omega$), $R_6$ ($k\Omega$)
  - 1: $5000.000\mu F$, 10000.000µF, 22.504, 22.511, 10000.000, 0.000, 0.000, 0.000

---

- **ac simulation**: $K = \text{dB}(\text{out} / \text{in})$
- **Equation**: $K = -3.01796$
- **Frequency (Hz)**: 1000.9

---

**DC simulation**
Filter transfer functions

- For odd order filters without transfer function zeros (Butterworth, Chebyshev Type-I and Bessel):

\[ H(s) = \frac{N(s)}{D(s)} = H_1(s) \prod_{i=0}^{N_2} H_2(s) = H_0 \frac{1}{s + C_N} \prod_{i=0}^{N_2} \frac{C_i}{s^2 + B_is + C_i} \]  

(1)

- For odd order filter with transfer function zeros (Cauer and Chebyshev-Type-II):

\[ H(s) = H_1(s) \prod_{i=0}^{N_2} H_2(s) = H_0 \frac{1}{s + C_N} \prod_{i=0}^{N_2} \frac{s^2 + A_i}{s^2 + B_is + C_i} \]  

(2)

- For even order filter without transfer function zeros:

\[ H(s) = \prod_{i=0}^{N_2} H_2(s) = H_0 \prod_{i=0}^{N_2} \frac{C_i}{s^2 + B_is + C_i} \]  

(3)

- For even order filter with transfer function zeros:

\[ H(s) = \prod_{i=0}^{N_2} H_2(s) = H_0 \prod_{i=0}^{N_2} \frac{s^2 + A_i}{s^2 + B_is + C_i} \]  

(4)
Qucs: Active filter design algorithms - 2

- Input magnitude response parameters:
  - Cutoff frequency
  - Stopband frequency
  - Passband attenuation
  - Stopband attenuation
  - Passband ripple (for Chebyshev and Cauer filters only)
  - Passband gain

- User defined transfer function representation

\[
H(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + \ldots + b_1 s + b_0}{a_n s^n + \ldots + a_1 s + a_0}
\]  

(5)

- Filter circuit building algorithm

Data: Filter Magnitude response parameters or transfer function numerator and denominator coefficients

Result: Active filter circuit in Qucs XML format in system clipboard

begin
  if UserDefinedTransferFunction then
    SolveEquations \(N(s) = 0, D(s) = 0\);
    Poles,Zeros ← FindPolesAndZeros();
  else
    SelectTransferFunctionApproximation();
    DetermineFilterOrder();
    Poles,Zeros ← CalculatePolesAndZeros();
  end
  SectionsCount ← EvaluateFilterSectionsCount();
end
  for \(i \leftarrow 1\) to SectionsCount do
    FindABCcoefficients(Poles[i],Zeros[i]);
    CalculateRCElementsValues();
  end
  BuildFilterCircuit();

Transfer function definition window
1. New style HTML “Qucs-Help” documentation

Background

The 'Quite universal circuit simulator' Qucs (pronounced: kju:ks) is an open source circuit simulator developed by a group of engineers, scientists and mathematicians under the GNU General Public License (GPL). Qucs is the brain-child of German Engineers Michael Margraf and Stefan Jahn. Since its initial public release in 2003 around twenty contributors, from all regions of the world, have invested their expertise and time to support the development of the software. Both binary and source code releases take place at regular intervals. Qucs numbered releases and day-to-day development code snapshots can be downloaded from (http://qucs.sourceforge.net). Versions are available for Linux (Ubuntu and other distributions), Mac OS X and the Windows 32 bit operating system.

In the period since Qucs was first released it has evolved into an advanced circuit simulation and device modelling tool with a user friendly "graphical user interface" (GUI) for circuit schematic capture, for investigating circuit and device properties from DC to RF and beyond, and for launching other circuit simulation software, including the FreeHDL (VHDL) and Icarus Verilog digital simulators. Qucs includes built-in code for processing and visualising simulation output data. Qucs also allows users to process post-simulation data with the popular Octave numerical data analysis package. Similarly, circuit performance optimisation is possible using the A SPICE Circuit Optimizer (ASCO) package or Python code linked to Qucs.

Between 2003, and January 2015, the sourceforge Qucs download statistics show that over one million downloads of the software have been recorded. As well as extensive circuit simulation capabilities Qucs supports a full range of device modelling features, including non-linear and RF equation-defined device modelling and the use of the Verilog-A hardware description language (HDL) for compact device modelling and macromodelling. Recent extensions to the software aim to diversify the Qucs modelling facilities by running the Berkeley “Model and Algorithm Prototyping Platform” (MAPP) in parallel with Qucs, using Octave launched from the Qucs GUI. In the future, as the Qucs project evolves, the software will also provide circuit designers with a choice of simulation engine selected from the Qucs built-in code, ngspice and Xyce ©.
Improvements to Qucs documentation and code control features - 2

2. New style HTML Qucs Tutorial/Report documents

Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors

Mike Brinson

Copyright 2014, 2015 Mike Brinson, Centre for Communications Technology, London Metropolitan University, London, UK. (mailtombrin72043@yahoo.co.uk)

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

Introduction

Resistors are one of the fundamental building blocks in electronic circuit design. In most instances conventional resistor circuit simulation models are characterized by I/V characteristics specified by Ohm's law. In reality the impedance of RF resistors is frequency dependent, being determined by component physical properties, component manufacturing technology and how components are connected in a circuit. At low frequencies fixed resistors have a nominal value at room temperature and can be modelled accurately by Ohm's law. At RF frequencies the fact that a resistor acts more like an inductance or capacitance can play a crucial role in determining whether or not a circuit operates as designed. Similarly, if a resistor is modelled as an ideal component at a frequency where it exhibits significant reactive properties then the resulting simulation data are likely to be incorrect. The subcircuit and Verilog-A compact resistor models introduced in this Qucs note are designed to give good performance from low frequencies to RF frequencies not greater than a few GHz.

Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47 Ω axial RF resistor.

Figure 8 - Qucs device model parameter extraction system applied to a nominal 47 Ω resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values: Rs = Rm = 47.3 Ω, CShunt = 0.08pF; Optimised values: Ls = L = 10.43nH, Llead = LL = 1.47nH, Cp = C = 0.69pF. To reduce simulation time the ASCO cost variance was set to 1e-3. The ASCO method was set to...
Improvements to Qucs documentation and code control features - 3

3. Source code documentation generated with Doxygen
Improvements to Qucs documentation and code control features - 4

**Concept**

With each release of Qucs the program code becomes more complex and the number of built in device models increases. In an attempt to check that new device models and code changes do not, inadvertently, introduce bugs or simulation errors a set of software tests are under development. Eventually, these tests will exercise all Qucs passive component and active device models across relevant simulation domains.

**Current state of work:**

- Tests operation of Qucs GUI (qucs) and simulator engine (qucsator)
- Test process uses Python, testing Qucs projects held in a “testsuit” directory ($python run.py --prefix /home/user/local/qucs-master/bin/ --skip skip.txt --qucsator)
- Projects which are known to fail can be skipped
- Test options:
  - `--qucs` runs the schematic to netlist conversion.
  - `--qucsator` runs the simulator.
  - `--add-test [schematic].sch` adds a schematic as a test-project into the test-set.
  - `--skip [file]` will skip the projects listed in the [file].
  - `--project [project directory from testset]` will run a single test from the set.
  - `--compare-qucsator [prefix/one prefix/two prefix/three]` runs multiple qucsator simulators.
  - `--verbose [0|1]` increase verbosity: 0 = progress and errors, 1 = all info.

**Typical Qucs simulation report for passed tests**

<table>
<thead>
<tr>
<th>Project</th>
<th>Schem. Version</th>
<th>Sim. Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC_SW_resonance_prj</td>
<td>0.0.4</td>
<td>0.016968</td>
</tr>
<tr>
<td>AC_SW_sw_meter_prj</td>
<td>0.0.4</td>
<td>0.017099</td>
</tr>
<tr>
<td>AC_bandpass_prj</td>
<td>0.0.17</td>
<td>0.115111</td>
</tr>
<tr>
<td>AC_groupdelay_ac_prj</td>
<td>0.0.12</td>
<td>0.034698</td>
</tr>
<tr>
<td>DC_AC_SP_stab_prj</td>
<td>0.0.4</td>
<td>0.067629</td>
</tr>
<tr>
<td>DC_AC_active_bp_prj</td>
<td>0.0.5</td>
<td>0.016422</td>
</tr>
<tr>
<td>DC_AC_active_bp_prj</td>
<td>0.0.5</td>
<td>0.017316</td>
</tr>
<tr>
<td>DC_AC_bbv_prj</td>
<td>0.0.5</td>
<td>0.032346</td>
</tr>
<tr>
<td>DC_AC_gain_prj</td>
<td>0.0.5</td>
<td>0.031954</td>
</tr>
<tr>
<td>DC_AC_gyrorator_prj</td>
<td>0.0.4</td>
<td>0.008243</td>
</tr>
<tr>
<td>DC_AC_notch_prj</td>
<td>0.0.5</td>
<td>0.016785</td>
</tr>
<tr>
<td>DC_AC_selective_AMP_prj</td>
<td>0.0.5</td>
<td>0.017569</td>
</tr>
<tr>
<td>DC_SP_LPF-Balun2_prj</td>
<td>0.0.15</td>
<td>0.216681</td>
</tr>
<tr>
<td>DC_SP_LPF-Balun3_prj</td>
<td>0.0.15</td>
<td>0.165655</td>
</tr>
<tr>
<td>DC_SP_opamp_gyrorator_prj</td>
<td>0.0.5</td>
<td>0.018379</td>
</tr>
<tr>
<td>DC_SW_bridge_prj</td>
<td>0.0.3</td>
<td>0.034325</td>
</tr>
<tr>
<td>DC_SW_bsim4v30MOS_lbs_Vgs_prj</td>
<td>0.0.18</td>
<td>0.217675</td>
</tr>
<tr>
<td>DC_SW_bsim4v30MOS_lbs_Vgs_prj</td>
<td>0.0.18</td>
<td>0.164787</td>
</tr>
<tr>
<td>DC_SW_charac_prj</td>
<td>0.0.4</td>
<td>0.117094</td>
</tr>
<tr>
<td>DC_SW_diode_prj</td>
<td>0.0.5</td>
<td>0.167375</td>
</tr>
<tr>
<td>DC_SW_diode_prj</td>
<td>0.0.18</td>
<td>0.421467</td>
</tr>
<tr>
<td>DC_SW_fummel_prj</td>
<td>0.0.10</td>
<td>0.034314</td>
</tr>
<tr>
<td>DC_SW_preregulator_prj</td>
<td>0.0.15</td>
<td>0.033987</td>
</tr>
<tr>
<td>DC_SW_rgummel_prj</td>
<td>0.0.10</td>
<td>0.065789</td>
</tr>
<tr>
<td>DC_TR_SW_spice_BFR520_prj</td>
<td>0.0.18</td>
<td>0.218216</td>
</tr>
<tr>
<td>DC_TR_active_mixer_prj</td>
<td>0.0.4</td>
<td>0.181998</td>
</tr>
<tr>
<td>SP_SmithChartTest_prj</td>
<td>0.0.12</td>
<td>0.065689</td>
</tr>
<tr>
<td>SP_Sparan_diagrams_prj</td>
<td>0.0.18</td>
<td>0.066238</td>
</tr>
<tr>
<td>SP_bpf_10Ghz_prj</td>
<td>0.0.4</td>
<td>0.065355</td>
</tr>
<tr>
<td>SP_chebyshev1_5th_prj</td>
<td>0.0.3</td>
<td>0.034585</td>
</tr>
<tr>
<td>SP_elliptic_5th_prj</td>
<td>0.0.3</td>
<td>0.067000</td>
</tr>
<tr>
<td>SP_fet_noise_prj</td>
<td>0.0.5</td>
<td>0.034303</td>
</tr>
</tbody>
</table>

4. Qucs model and program code test system
Summary and possible future directions

- Qucs 0.0.19 is likely to be a major release of the circuit simulator package with many of the features introduced in this presentation included. At this stage in the Qucs development cycle it is difficult to say what the final structure of the software will be, or indeed how complete the new features will be. However, as a minimum Qucs 0.0.19 will have benefited from the significant amount of work done by the 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.

MUCH WORK STILL NEEDS TO BE DONE BEFORE FURTHER NEW FEATURES ARE ADDED TO THE SOFTWARE CODE.

- Medium to long term possible improvements to Qucs have been published in the revised Qucs Wiki roadmap. This can be found at

https://github.com/Qucs/qucs/wiki/Roadmap

Stable and development versions of Qucs and MAPP can be downloaded from: