Verilog-A Compact Semiconductor Device Modelling and Circuit Macromodelling with the QucsStudio-ADMS "Turn-Key" Modelling System

M. E. Brinson
Centre for Communications Technology
London Metropolitan University
London N78DB, UK
mbrin72043@yahoo.co.uk

M. Margraf
Qucs and QucsStudio Project Founder
Berlin
Germany
Michael.margraf@alumni.tu-berlin.de

- Background
- Verilog-A compact device modelling with Qucs-ADMS
- QucsStudio-ADMS "Turn-Key" Verilog-A compact device modelling
- A simplified Verilog-A npn RF BJT model
- QucsStudio C++ compiled model programming interface
- Adding Verilog-A natures to QucsStudio
- Transient simulation of a class A RF npn BJT amplifier
- Conclusions
- Future directions

Presented on 25 May 2012 at the special session on "Compact Modelling Support for Nanoscaled IC Technology and Design": 19 International Conference on Mixed Design of Integrated Circuits and Systems, Warsaw, Poland.

Background

- Until the adoption of Verilog-A as the preferred analogue hardware description language for compact semiconductor device modelling by the Compact Model Council, C was the standard modelling language.
- However, hand coding models in C was often found to be very tedious, time consuming and subject to error, particularly when determining partial derivatives of device currents and charges needed in non-linear circuit simulation.
- In contrast to C Verilog-A provides built-in tools which automatically generate partial derivatives, making compact device modelling a much more straight forward process.
- Current trends indicate that there is a growing acceptance by the compact modelling community of the Verilog-AMS subset Verilog-A as the preferred compact modelling language.
- The standardization of Verilog-AMS and specifically the addition of a number of compact modelling enhancements to its analogue Verilog-A subset have also greatly influenced Verilog-A usage.
- The release of the Verilog-A “Analogue Device Model Synthesizer” (ADMS) software under the GNU General Public License has also accelerated the rate at which Verilog-A has been accepted and used by the modelling community as a replacement for C.

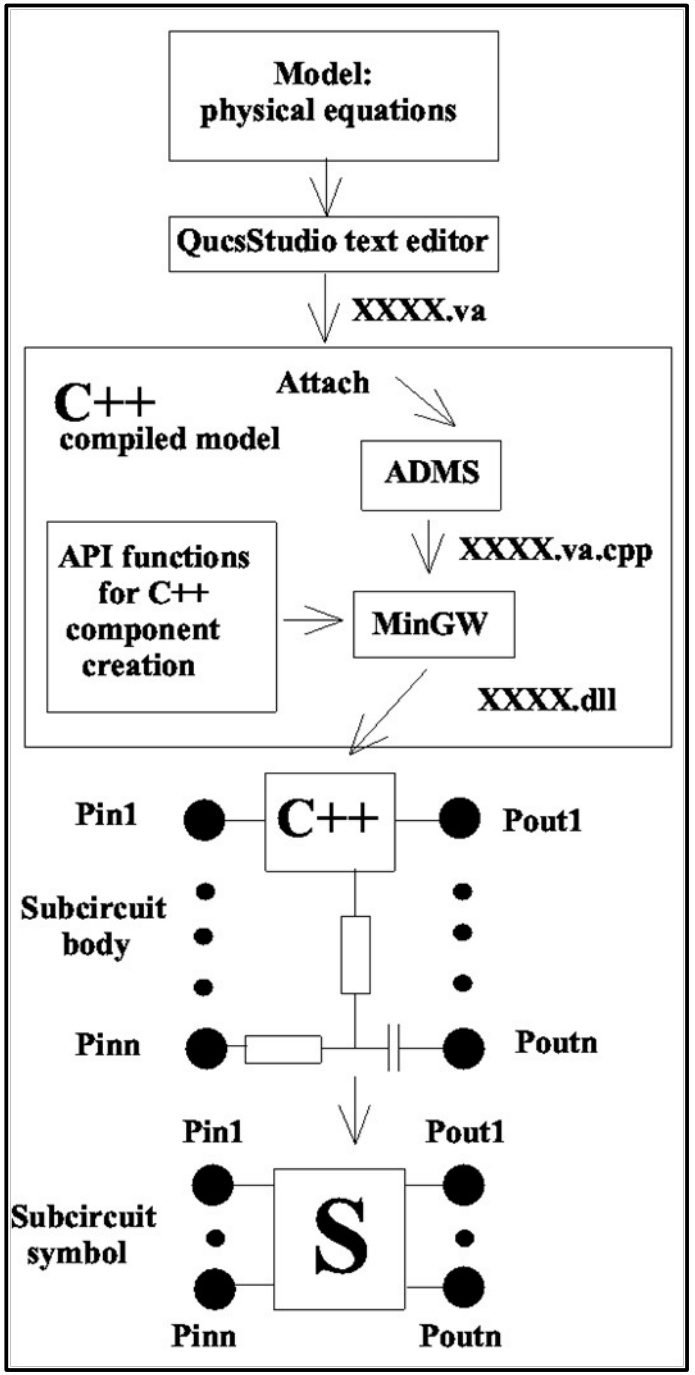
Verilog-A compact device modelling with Qucs-ADMS

- Verilog-A compact device modelling was first implemented in Qucs version 0.0.11.
- In the Qucs modelling process the ADMS Verilog-A to C++ synthesizer was used to compile Verilog-A model code to C++ code manually.
- After conversion the C++ code also had to be manually merged with the main body of the Qucs circuit simulator.
- Similarly, the Qucs graphical user interface code needed to be patched to add a new model symbol to the simulators library of built-in component symbols.
- Finally, due to the fact that the Qucs simulator uses static C++ model libraries the entire simulator C++ code had to be re-compiled and re-linked to generate an extended simulator each time a compact model was added to the software.
- In principle, it was possible to add compact device models to Qucs using the above procedure.
- In practice, the modelling process requires users to have an advanced knowledge of C++ programming coupled with a good understanding of the Qucs model application programming interface.
- The Qucs compact device modelling process was further complicated in that it used software tools supplied with Linux rather than the more universally available Microsoft Windows® operating system.

QucsStudio-ADMS 'Turn-Key' Verilog-A compact device modelling

- A primary aim of the new QucsStudio Verilog-A compact device modelling system is to provide the circuit simulation software with a simple tool that does not require the main body of the simulator C++ code to be patched by hand when adding compact models.
- In contrast to the Qucs modelling scheme the QucsStudio version is based on dynamic linked model libraries rather than static model libraries.
- The use of dynamic libraries suggests that the new modelling process be called 'Turn-Key' to emphasise the fact that the software takes over responsibility for determining when a compact device model needs to be re-compiled and re-linked.
- Changes in Verilog-A model code act like a key turning on the compilation and linking of changed code.
- When changes take place, edited models are automatically updated at the start of the next user requested circuit simulation.

Flow chart showing QucsStudio Verilog-A compact modelling stages: Verilog-A code to subcircuit.



A simplified QucsStudio-ADMS Verilog-A npn RF BJT model: Part 1 the model Verilog-A code

- The compact model presented here and the next slide consists of a large signal Ebers-Moll bipolar transistor equivalent circuit with second order high-level current injection effects and internal capacitance included, plus external lead inductance and capacitance.

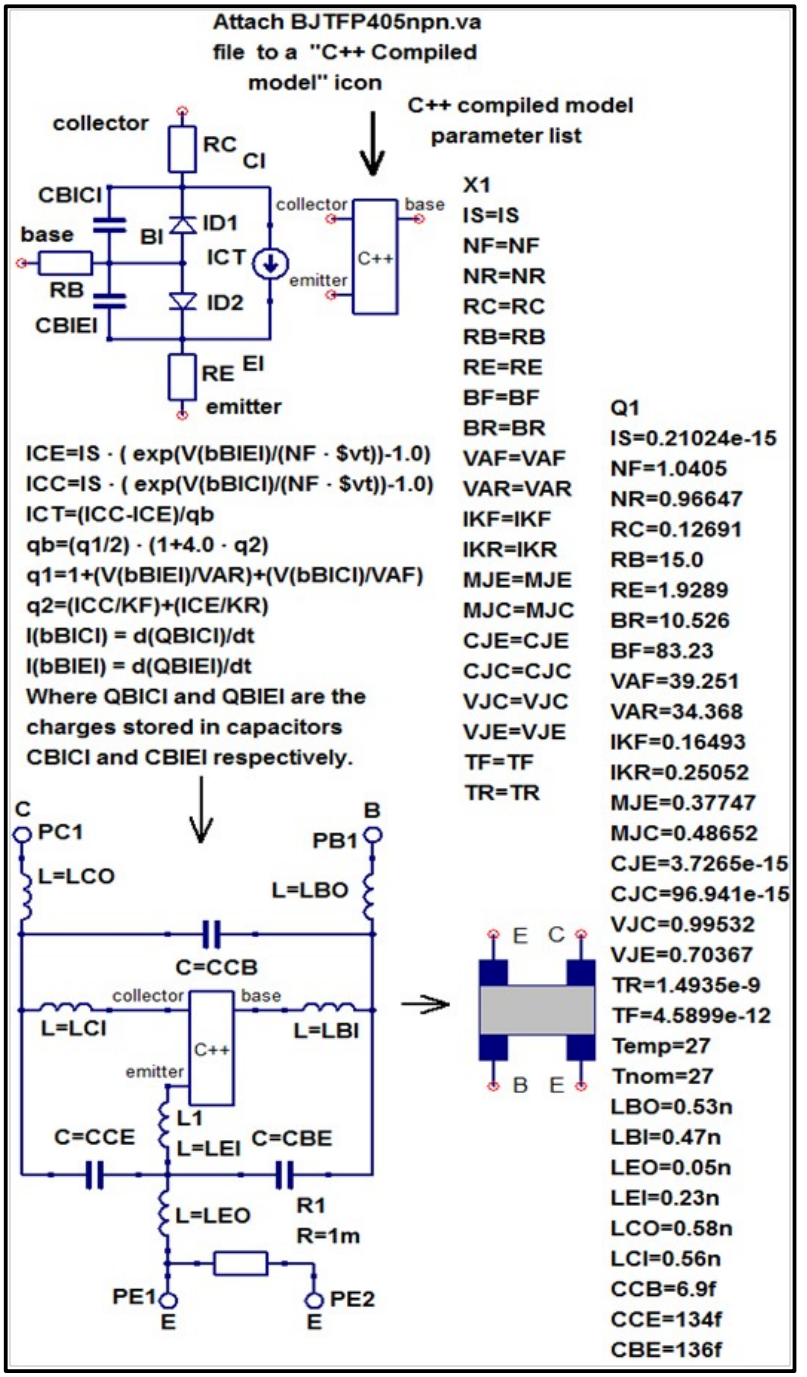
Verilog-A code for a simplified RF npn BJT: the Model parameters have the same meaning as those defined in the SPICE 3f5 BJT model.

```
//   Verilog-A npn RF BJT model
`include "disciplines.vams"
`include "constants.vams"
module BJTFP405npn(collector,base,emitter);
inout collector,base,emitter ;  electrical collector,base,emitter;
electrical CI, BI, EI, nI1; // Internal nodes.
`define CTOK 273.15
`define GMIN 1e-12
`define TWOQ 2*`P_Q
parameter real IS = 0.21024e-15 from [1e-20 : inf);
parameter real NF = 1.0405 from [0.5 : inf) ;
parameter real NR = 0.96647 from [0.5 : inf) ;
parameter real RC = 0.12691 from [1e-20 : inf);
parameter real RB = 15.0 from [1e-20 : inf);
parameter real RE = 1.9289 from [1e-20 :inf;)
parameter real BF = 83.23 from [1e-20 : inf) ;
parameter real BR = 10.526 from [1e-20 : inf);
parameter real VAF = 39.251 from [1e-20 : inf);
parameter real VAR = 34.368 from [1e-20 : inf);
parameter real IKF = 0.16493 from [1e-20 : inf);
parameter real IKR = 0.25052 from [1e-20 : inf);
parameter real MJE = 0.37747 from [1e-20 : inf);
parameter real MJC = 0.48652 from [1e-20 : inf);
parameter real CJE = 3.7265e-15 from [1e-20 : inf);
parameter real CJC = 96.941e-15 from [1e-20 : inf);
parameter real VJC = 0.99532 from [1e-20 : inf);
parameter real VJE = 0.70367 from [1e-20 : inf);
parameter real TF  = 4.5898  from [1e-20 : inf);
parameter real TR  = 1.4935  from [1e-20 : inf);
parameter real Temp = 27 from [-273.15 : inf);
real con1, con2, con3, con4, con5, con6, con7;
real  x1, y1, x2, y2, z1, z2, con8, con9, QBICI, QBIEI, q1O2;
real IEC, ICC, q1, q2, T1, T2,  VJCO2, VJEO2;
// Model branches
branch (collector, CI)  bcollectorCI;   branch (base, BI)  bbaseB1;
branch (EI, emitter)  bEIemitter;  branch (BI, CI)  bBICI;
branch (BI, EI) bBIEI;  branch (CI, EI) bCIEI;
analog begin
  con1 = 1.0/(NF*$vt);  con2 = 1.0/(NR*$vt);
  VJCO2 = VJC/2;  VJEO2 = VJE/2;  con3 = 1.0-MJE;
  con4 = 1.0-MJC; con5 = exp(MJE*ln(2)); con6 = exp(MJC*ln(2));
 // Current contributions
  I(bcollectorCI) <+ V(bcollectorCI)/RC;  I(bbaseBI) <+ V(bbaseBI)/RB;
   I(bEIemitter)  <+ V(bEIemitter)/RE;
  IEC = IS*(limexp(V(bBICI)*con2)-1.0);  ICC = IS*(limexp(V(bBIEI)*con1)-1.0);
  q1=1.0 + V(bBICI)/VAF + V(bBIEI)/VAR;   q2  = ICC/IKF + IEC/IKR;
  I(bBICI) <+ IEC/BR + `GMIN*V(bBICI);  I(bBIEI) <+ ICC/BF + `GMIN*V(bBIEI);
  q1O2 = q1/2;  I(bCIEI)  <+ (ICC-IEC)/(1e-20+q1O2*sqrt(1.0+4*q2));
  y1 = 1.0-V(bBICI)/VJ  y2 = 1.0-V(bBIEI)/VJE;
  z1 = exp(con4*ln(y1));  z2 = exp(con3*ln(y2));
  QBICI = (V(bBICI)>=VJCO2)
          ? TR*IEC+CJC*con6*(V(bBICI)*V(bBICI)+con4*V(bBICI))
          : TR*IEC+CJC*((VJC/con4)*(1.0-z1));
  QBIEI = (V(bBIEI)>=VJEO2)
          ? TF*ICC+CJE*con5*(V(bBIEI)*V(bBIEI)+con3*V(bBIEI))
           : TF*ICC+CJE*((VJE/con3)*(1.0-z2));
 I(bBICI) <+ ddt(QBICI);  I(bBIEI) <+ ddt(QBIEI);
end
endmodule
```

A simplified QucsStudio-ADMS Verilog-A npn RF BJT model: Part 2 the QucsStudio subcircuit model

- On attaching the BJT Verilog-A code to the QucsStudio C++ compiled model icon the software tries to extract the external node names and parameters
- If successful QucsStudio draws a group of named terminals attached to the C++ compiled model icon

The QucsStudio example RF npn BJT subcircuit device model: top; C++ compiled model icon: bottom model showing external parasitic components and subcircuit symbol plus parameters



QucsStudio C++ compiled model programming interface

- Central to the operation of the QucsStudio Verilog-A modelling system is a C++ compiled model component.
- QucsStudio built-in components are defined by a C++code template which lists model properties.
- This list contains amongst other things the number of external and internal nodes as well as pointers to the model parameter list and to the model schematic symbol.
- It also contains function calls, for example (tEvaluate)Matrix that determine the physical operation of a component.
- Many function calls are optional.
- In order to synthesize the C++ code needed for simulation of an analogue model the QucsStudio-ADMS-MinGW tools undertake the required operations in terms of a model API.

```
// component definition
EXPORT tComponentInfo compInfo = {
  isNonLinear,        // component type ('isNonLinear' or 'isLinear')
  "BJTFP405npn",   // model identifier
  "VerilogAMS model of BJTFP405npn", // component description
  3,                  // number of external nodes
  3,                  // number of internal nodes
  0,                  // number of inputs (system simulations only)
  20,                 // number of parameters
  params,             // pointer to list of parameters
  0,                  // size of global variable buffer in bytes
  0,                  // pointer to component icon (0 = unused)
  0,                  // size of component icon
  -1,                 // index of parameter determining schematic symbol (-1 = unused)
  0,                  // pointer to list of schematic symbols
  (tEvaluate)fillMatrix,     // function calculating analog model (0 = no model exists)
  0,                  // function calculating noise model (0 = noise free)
  0,                  // function calculating system model (0 = no model exists)
  0,                  // string with digital Verilog model (0 = no model exists)
  0                   // string with digital VHDL model (0 = no model exists)
};
```

QucsStudio C++ component definition template

```
1. setA(Node1, Node2, num);
      sets a single linear matrix element
2. setG(Node1, Node2, conductance);
      sets a linear conductance
3. setR(Node1, Node2, resistance);
      sets a linear resistance
4. setC(Node1, Node2, capacitance, initial voltage);
      sets a linear capacitance
5. setL(Node1, Node2, internal Node, inductance, initial current);
      sets a linear inductance
6. setM(Node1, Node2, mutual inductance, initial current);
      sets a linear mutual inductance
7. setI(Node1, Node2, current);
      sets a linear current
8. setDelayedI(Node1, Node2, line constant, delay, buffer pointer);
      sets a linear time-delayed current
9. setIQ(Node1, Node2, current, charge);
      sets a non-linear current and charge
10. setGC(Node1, Node2, Node3, Node4, current derivative, charge derivative);
      sets a non-linear conductance and capacitance
11. setNoiseA(Node1, Node2, noise current density);
      sets a single linear noise matrix element
12. setNoiseG(Node1, Node2, noise current density);
      sets a linear noise current
13. setNoiseNG(Node1, Node2, noise current density);
      sets a non-linear noise current
```

QucsStudio C++ model Application Programming Interface functions

Adding Verilog-A natures to QucsStudio

- A high percentage of compact device models include a mixture of linear and non-linear R, C and L components.
- The 2.3.0 version of ADMS appears to treat all R and C components as non-linear elements.
- Also in this version of ADMS it is not permitted to express the connection of inductance L between circuit nodes p and q as V(p,n) <+ L*ddt(I(p,n));.
- This limitation is overcome by representing L as a combination of a grounded capacitor with a gyrator.
- In compact models with a significant of R, C and L components simulation run times can be reduced, especially transient simulation, by ensuring that the QucsStudio "Turn-Key" modelling system uses linear R, C and L whenever possible rather than non-linear R, C or non-linear models.
- In QucsStudio the selection of linear or nonlinear R, C and L components has been implemented by adding three new Verilog-A "natures" to the standard disciplines and natures file "disciplines.vams".
- A parameter, called insideQucsStudio, is defined by QucsStudio to allow automatic linear or non-linear component selection from within Verilog-A model code.

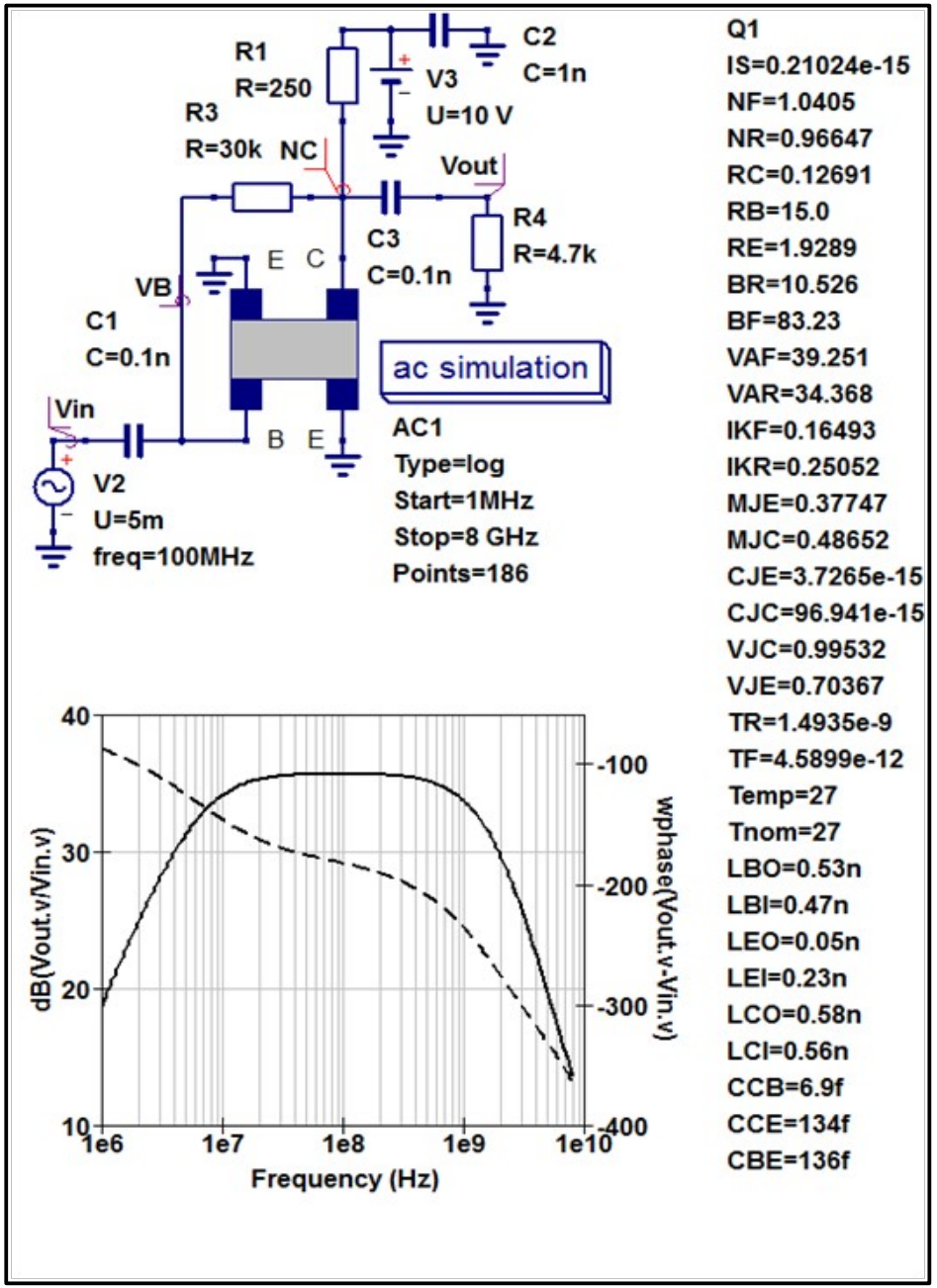
```
nature Resistance          nature Conductance          nature Capacitance
 units = "ohms";             units = "S";                units = "F";
 access = R;                 access = G;                 access = C;
endnature                  endnature                   endnature

                          discipline electrical
                                potential Voltage;
                                flow Current;
                                flow Conductance;
                                flow Resistance;
                                flow Capacitance;
                          enddisciplne
```

```
`ifdef  insideQucsStudio
    R(b1) <+ Rvalue;
`else
   I(b1) <+ V(b1)/Rvalue;
`endif
```

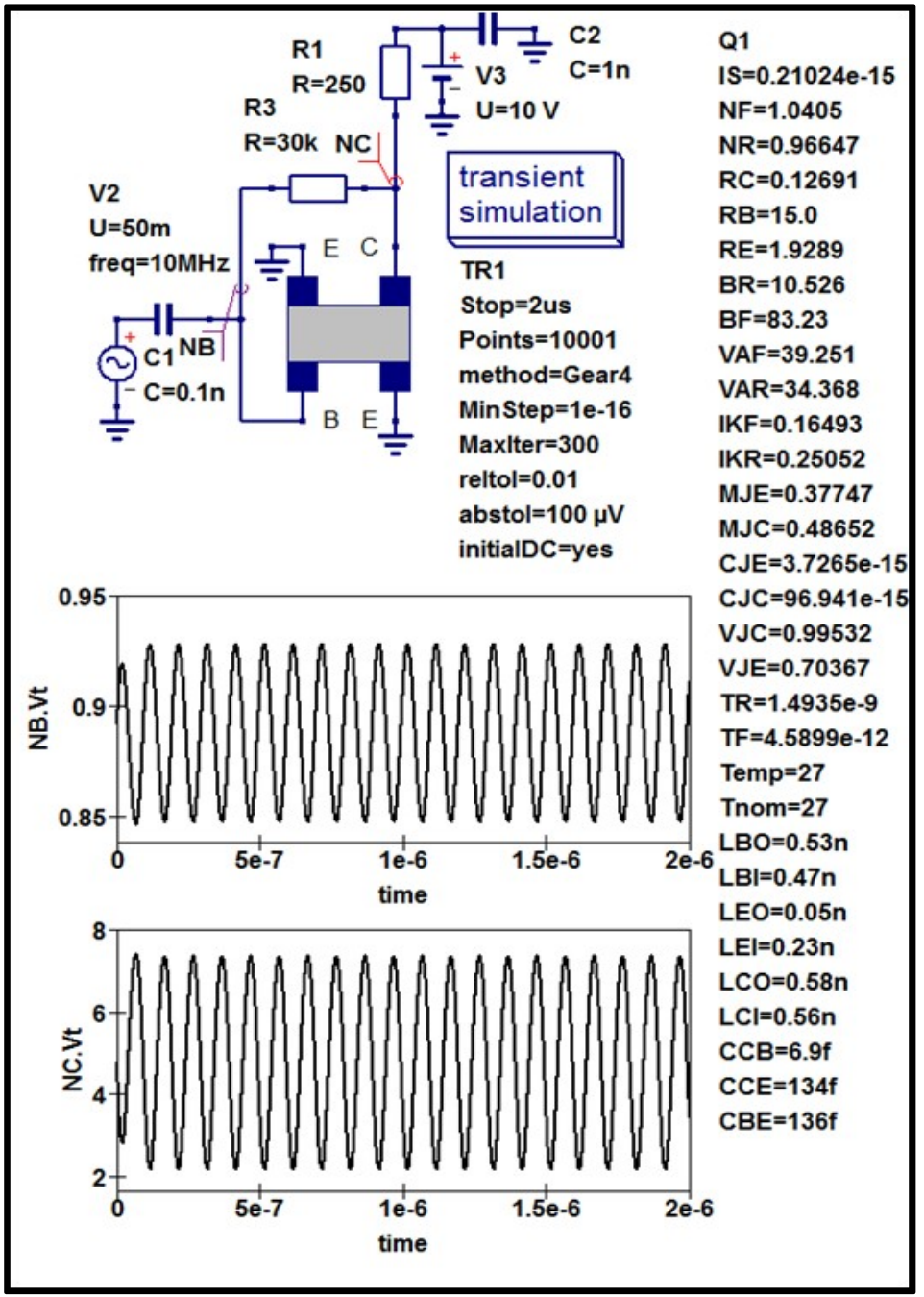
Example R selection
Using insideQucsStudio.

QucsStudio natures plus modified electrical discipline for linear R and C components.

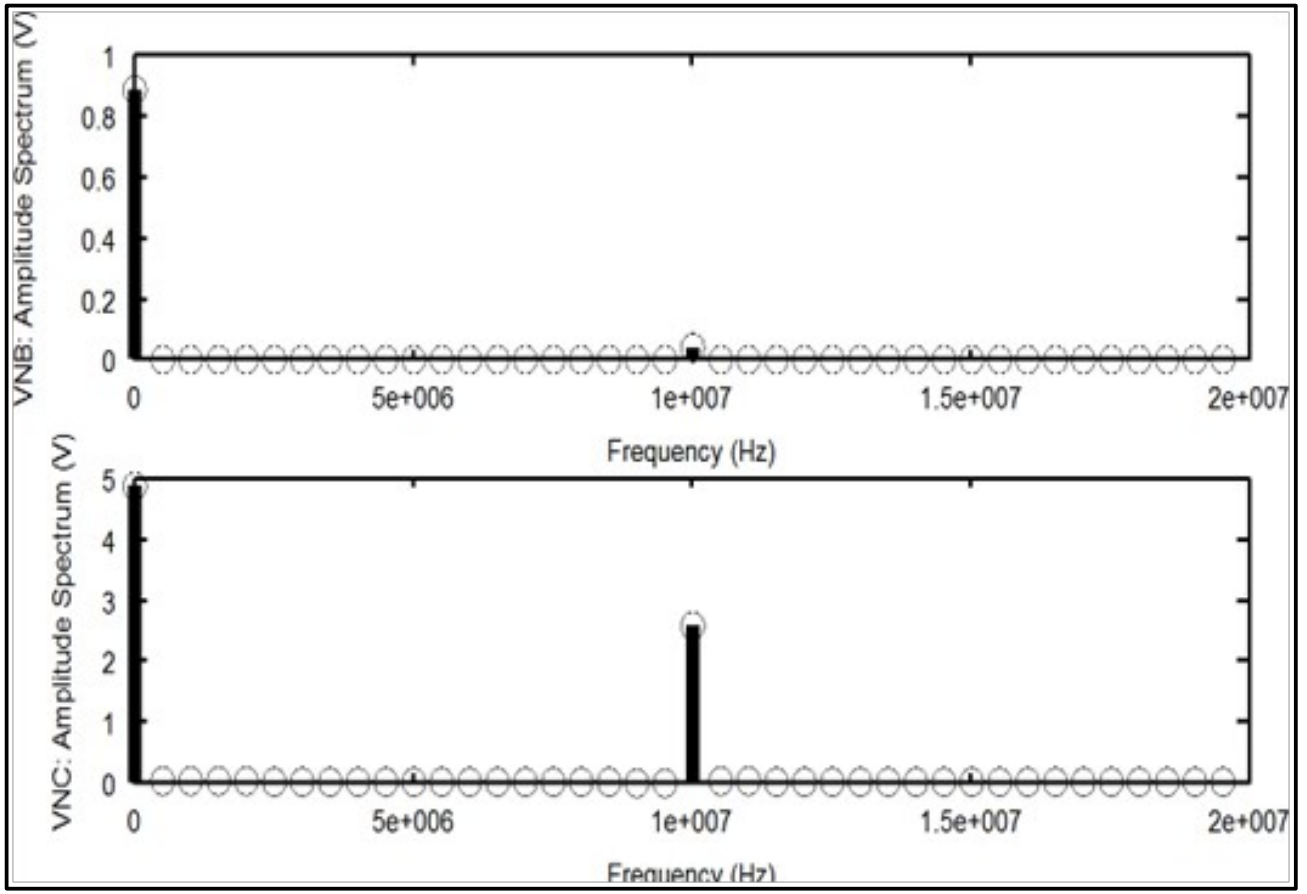
A class A npn BJT RF amplifier with collector feedback: small signal AC test circuit and gain and phase curves; legend: solid line = dB(Vout.v/Vin.v) And dotted line = wphase(Vou.v/Vin.v) [the unwrapped phase in degrees]



A class A npn BJT RF amplifier with collector feedback: transient simulation test circuit with directly coupled voltage test probes NB and NC and time response output waveforms for a sinusoidal input signal of 50mV peak, 10MHz and 0 phase



Voltage amplitude spectra plotted against frequency for amplifier probe signals NB.Vt and NC.Vt



Octave “m” script for post-simulation data processing: functions “loadQucsDataset” and “loadQucsVariable” are provided with QucsStudio for conversion of simulation output data to Qctave Internal format

```
% File testfeedbacknpnTRAN.m file
% Control file called on completion of transient simulation.
% Calls functions loadQucsDataset,loadQucsVariable
% and stemfft.
function stemfft(data, points, FinTim)
 yfft  = abs(fft(data/points));
 no2   = length(yfft)/2;
 yvec(1) = yfft(1);
 yvec([2:no2])  = 2*yfft([2:no2]);
 fc    = linspace(0, no2, no2)/FinTim;
 stem(fc, yvec, "linewidth", 4 , "color", "black");
endfunction

qucsFilename = 'TestfeedbacknpnTRAN.dat';
loadQucsDataset;
whos
clf()
newplot()
[VNB,Dep]=loadQucsVariable("TestfeedbacknpnTRAN.dat","NB.Vt");
[VNC,Dep1]=loadQucsVariable("TestfeedbacknpnTRAN.dat","NC.Vt");
[Time,Dep3]=loadQucsVariable("TestfeedbacknpnTRAN.dat","time");

subplot(2,1,1)
stemfft(VNB, 2048, 2e-6);
set(gca, "linewidth", 4, "fontsize", 20,  "fontname", "TimesRoman",
 "fontweight", "bold",  "xlim", [0,20e6], "xlabel", text("string",
 "Frequency (Hz)", "fontsize", 20, "fontname", "TimesRoman",
 "fontweight", "bold"), "ylabel",  text("string",
 "VNB: Amplitude Spectrum (V)", "fontsize", 20,
 "fontname", "TimesRoman", "fontweight", "bold", "rotation", 90));

subplot(2,1,2);
stemfft(VNC, 2048, 2e-6);
set(gca, "linewidth", 4, "fontsize", 20,  "fontname", "TimesRoman",
 "fontweight", "bold", "xlim", [0,20e6], "xlabel", text("string",
 "Frequency (Hz)", "fontsize", 20, "fontname", "TimesRoman",
 "fontweight","bold"), "ylabel",  text("string",
 "VNC: Amplitude Spectrum (V)", "fontsize", 20,
 "fontname", "Arial", "fontweight", "bold", "rotation", 90));
print("TestfeedbacknpnTRAN.png","-dpng");
```

Conclusions

- This presentation outlined the background and practical use of a new device modelling system which has been specifically designed for straightforward Verilog-A compact model construction by users who do not have an advanced knowledge of C++ programming or who are not familiar with the details of circuit simulator C++ model interfaces.
- The new QucsStudio Verilog-A compact modelling system uses dynamic linked libraries, which suggests the term “Turn-Key” modelling to describe the presented modelling process, where changes in Verilog-A model code act as a key turning on re-compiling and re-linking of changed model code.
- As far as the authors are aware QucsStudio is the first GNU open source GPL circuit simulator with a Verilog-A “Turn-Key” style compact device modelling system.
- Coupling the new modelling system with equation-defined device modelling, circuit macromodelling and Octave post-simulation data processing provides QucsStudio with a set of modelling and simulation tools previously not freely available within a single circuit simulation GPL software package.

QucsStudio is freely available under the open source GNU General Public Licence
Download QucsStudio from:

http://mydarc.de/DD6UM/QucsStudio/qucsstudio.html

[Currently QucsStudio supports Windows® only: QucsStudio-1.3.2.zip or QucsStudio-1.3.2_light.zip {without Octave and model compiler}]

Future Directions

In 2013 Qucs/QucsStudio will celebrate 10 years since Qucs version 0.0.1 was released for general use under the GNU open source General Public License. To mark this important mile stone in the development of open source circuit simulators a new generation of QucsStudio simulation and modelling tools is to be launched under the title QucsStudio 2.0.0. This new software package will build on the success of the current QucsStudio release adding extensions centred on the following topics:

- Advanced binary linked component libraries to QucsStudio
- Faster analogue simulator based on a direct sparse linear equation solver
- Extended range of compact device models and circuit macromodels
- Improved system simulation facilities

NEW BOOK: Open Source/GNU CAD for Compact Modelling, Editors: Wladek Grabinski and Daniel Tomaszewski. Publisher: Mark de Jongh [Mark.de.Jongh@springer.sbm.com], www.springer-sbm.com.
Chapter 5: M.E. Brinson, **Schematic entry and circuit simulation with Qucs.**
Chapter 6: M.E. Brinson, **Qucs modelling and simulation of analogue/RF devices and circuits.**