Recent Developments in Qucs-S Equation-Defined Modelling of Semiconductor Devices and IC's

Mike Brinson, and Vadim Kuznetsov

***Abstract*—The Qucs Equation-Defined Device was introduce roughly ten years ago as a versatile behavioural simulation component for modelling the non-linear static and dynamic properties of passive components, semiconductor devices and IC macromodels. Today, this component has become an established element for building experimental device simulation models. It's inherent interactive properties make it ideal for device and circuit modelling via Qucs schematics. Moreover, Equation-Defined Devices often promote a clearer understanding of the factors involved in the construction of complex compact semiconductor simulation models. This paper is concerned with recent advances in Qucs-S/Ngspice/XSPICE modelling capabilities that improve model construction and simulation run time performance of Equation-Defined Devices using XSPICE model syntheses. To illustrate the new Qucs-S modelling techniques an XSPICE version of the EPFL EKV v2.6 long channel transistor model together with other illustrative examples are described and their performance simulated with Qucs-S and Ngspice.**

***Index Terms*—Qucs, Qucs-S, Ngspice, XSPICE Code Models, compact semiconductor device modelling, Equation-Defined Devices (EDD), macromodels.**

I. Introduction

QUCS Equation-Defined Device (EDD) modelling of existing and emerging technology devices and IC's has become an established technique since the Qucs EDD was first released roughly ten years ago [1][2]. The popularity of the Qucs EDD component can be largely traced back to three of it's primary characteristics; firstly it can model the static and dynamic properties of physical devices expressed as a set of explicit compact model equations, secondly the structure and properties of the Qucs EDD have a direct relationship to Verilog-A hardware device language (HDL) statements [3], making conversion of EDD models to Verilog-A straight forward [4], and finally EDD model technology is both interactive and integrates easily with conventional circuit simulation components. At this time Qucs and it's SPICE variant Qucs-S [13] [14] appear to be the only open source simulators to have implemented the EDD. Although EDD modelling is easily applied to the construction of subcircuit models of semiconductor devices and IC's these models often tend to simulate much slower than compiled C/C++ code models. Hence, conversion of EDD derived models to HDL C/C++ models is recommended, particularly in those situations where a model is to be distributed as part of a circuit simulator package. Unfortunately, the current popular SPICE derived General Public License (GPL) circuit simulators do not implement the Analogue Device Model Synthesizer (ADMS) [5] software in a consistent way, making automatic conversion of behavioural device models to compiled C/C++ versions either difficult, or indeed sometimes not feasible. Although the Ngspice [6] and Xyce © [7] simulators both employ ADMS for Verilog-A to C++/C code translation they require that the entire simulator code is compiled and linked manually when adding new models. The Qucs-S variant of Qucs overcomes EDD simulation speed limitations by the addition of an "XSPICE CodeModel synthesizer", which has been specifically developed for converting EDD model schematics into "turn-key" C code models. The purpose of this paper is to introduce a number of recent improvements in Qucs-S EDD modelling of semiconductor devices and IC's. The improvements in Qucs-S EDD modelling capabilities are also demonstrated by an outline of (1) an XSPICE synthesized version of a basic EPFL EKV v2.6 long channel compact transistor model [8], (2) an RF inductance XSPICE CodeModel, and (3) other example models and simulation data.

M. Brinson is with the Centre for Communications Technology, London Metropolitan University, UK, e-mail: mbrin72043@yahoo.co.uk

V. Kuznetsov is with the Department of Electronic Engineering, Bauman Moscow State Technical University, Kaluga branch, Russia; e-mail: ra3xdh@gmail.com

II. The Structure of Qucs-S Equation-Defined Device (EDD) Models

Qucs-S behavioural models of semiconductor devices and IC's can be constructed from Qucs EDD, SPICE B non-linear voltage and current sources combined with linear components to form user defined subcircuits. This approach provides a convenient interactive way of building and testing new experimental compact device models and IC macromodels. When first introduced the Qucs EDD was conceived as an eight branch two terminal non-linear modelling block where individual terminal currents (I_j) and internal stored charge (Q_j) are given by equations 1 to 3.

$$I_j = I_j(V_j), g_j = \frac{dI_j}{dV_j} \tag{1}$$

$$Q_j = Q_j(V_j, I_j) \tag{2}$$

$$C_j = \frac{dQ_j}{dV_j} = \frac{\partial Q_j(V_j)}{\partial V_j} + \frac{\partial Q_j(I_j)}{\partial I_j} \cdot g_j \tag{3}$$

where $1 <= j <= 8$, I_j is the current flowing through branch j, V_j is the voltage across branch j, Q_j is branch j internal stored charge, g_j, and C_j are branch conductance and capacitance respectively. One of the primary uses of Qucs EDD is to evaluate non-linear algebraic/differential equations which represent the physical properties of a device being modelled. These fall into two main forms, firstly explicit algebraic equations which represent static quantities, for example DC current, and secondly dynamic quantities, for example capacitor current. In practice intermediate algebraic equations

are also often used in the calculation of the values of model variables. As a general rule it is straight forward, and convenient, to calculate model equation values by representing the individual model variables as the current flowing in one of the EDD branches (I_j). Similarly, conversion of such currents to voltages, prior to use in calculating further model variables, is easily achieved by passing I_j through a one Ohm resistor. EDD can also automatically calculate device dynamic currents from the algebraic equations specifying the charge (Q_j) stored by each of the EDD branches. An example of this style of fundamental EDD modelling is shown in Figure 1. Experience has shown that the limit of eight branches per EDD is insufficient when constructing complex compact semiconductor device models. As a consequence the original Qucs EDD branch limit of 8 has been increased to 20.

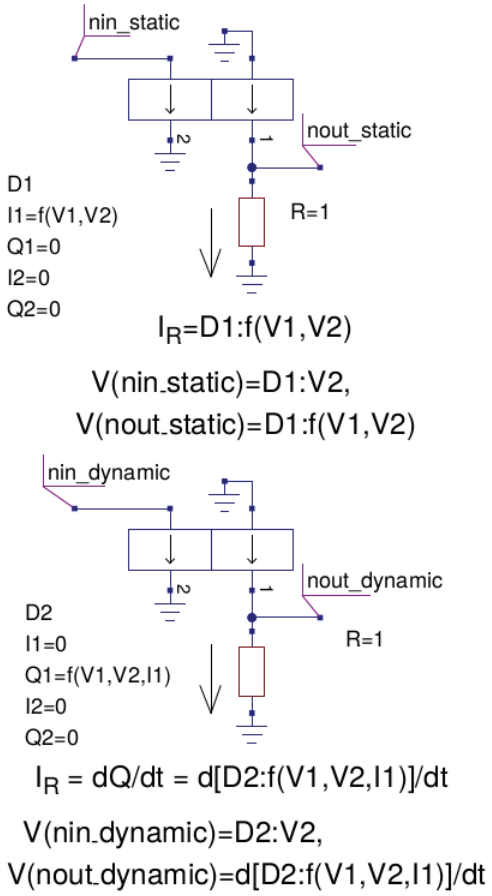

Fig. 1. A template showing the application of EDD for the evaluation of non-linear algebraic/differential equations that represent static and dynamic compact model physical quantities which are a function of two or more voltage or current variables.

III. The Synthesis of XSPICE CodeModels

Versions of the XSPICE CodeModel development software are distributed with the Ngspice and SPICE OPUS [9] circuit simulators, including implementations of the standard ”analog”, ”digital” and ”hybrid” CodeModel libraries. The standard XSPICE CodeModel libraries are distributed as file libname.cm with each model built from sets of files called ”name.ifs” and ”name.mod”. These contain model interface and source code, respectively. The Qucs-S version of the XSPICE CodeModel software is different to Ngspice and SPICE OPUS distributions in that an XSPICE CodeModel synthesizer now forms part of the circuit simulation package. This extension automatically generates files ”name.ifs” and ”name.mod” files from an EDD schematic and an attached Qucs Equation block. The synthesis process is similar to that adopted by the Qucs-S Verilog-A synthesizer [10]. Moreover, it is a full ”turn-key” package which does not require users to manually patch the Qucs-S C++/C code. To synthesize an XSPICE CodeModel an EDD schematic is drawn and branch variables $I_1...I_n$ and $Q_1...Q_n$ entered as algebraic/numeric expressions, followed by any required Equation block values. Selecting Qucs-S GUI commands ”Create XSPICE IFS” and ”Create XSPICE MOD” initiates automatic synthesis of the XSPICE CodeModels under development.

The arrows shown in Figure 2(a) indicate the sequence of the individual steps undertaken by Qucs-S/Ngspice to synthesize an EDD C model from an initial EDD drawing to attaching the synthesized C code to a Qucs-S schematic component symbol. The C code generated by the XSPICE CodeModelling software for the non-linear resistor example is shown in Listing 1 (file Vcontrolc.mod), where the code built by this process is in accordance with the XSPICE model template structure, see the XSPICE documentation [11] for the details. One point to note regarding the model C code in Listing 1 is the fact that the partial derivatives of the EDD branch currents have been automatically generated; in the case of the non-linear resistor example, the first order partial derivatives $\partial I_1/\partial V_1$ and $\partial I_1/\partial V_2$ are required. Unlike the Qucs-S Verilog-A modelling tool the ADMS software package is not used for the synthesis of model C code but has been replaced by the Ginac C++ library [12]. The Ginac GPL package is designed to allow the creation of integrated software systems that embed symbolic manipulations, like for example the generation of partial derivatives, with computational intensive numeric routines, see Listing 1.

```
/* XSPICE codemodel VcontrolR */
/* auto-generated template */
#include <math.h>
#define D_0_step(x) (0)
#define step(x) ((x)>0.0
  ? 1.0
  : (((x)==0)?0.5:0.0))
#define Xpow(x,p) pow(fabs((x)),(p))
void cm_VcontrolR(ARGS) {
 Complex_t ac_gain00, ac_gain01, ac_gain10,
           ac_gain11;
 static double R0,k1,k2,k3;
 static double V1,V2,V1_old,V2_old;
 double Q0, cQ0, Q1, cQ1, delta_t;
 if(INIT) {
  R0 = PARAM(r0); k1 = PARAM(k1);
  k2 = PARAM(k2); k3 = PARAM(k3);
 }
 if (ANALYSIS != AC) {
  if (TIME == 0) {
   V1_old=V1=INPUT(nRp_nRn);
   V2_old=V2=INPUT(nV2_gnd);
   Q0=0.0; cQ0=0.0;
  } else {
   V1 = INPUT(nRp_nRn); V2 = INPUT(nV2_gnd);
   delta_t=TIME-T(1); V1_old=V1; V2_old=V2;
  }
```

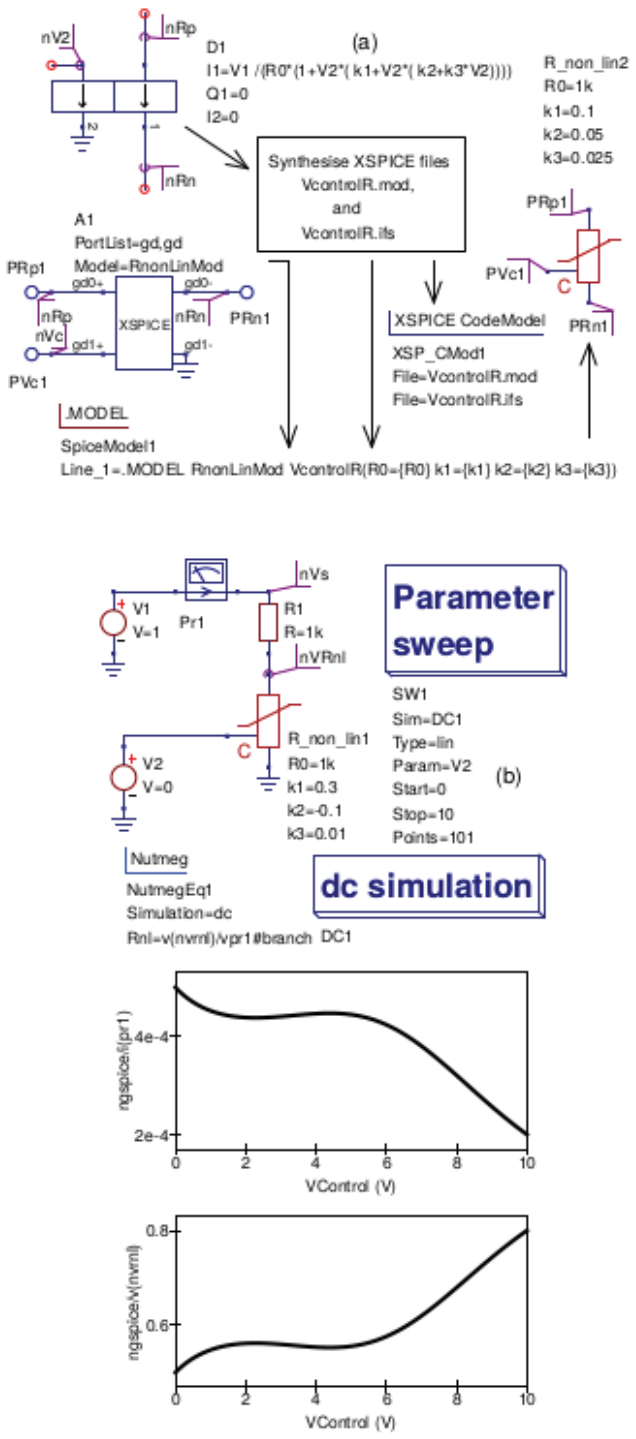

Fig. 2. Qucs-S EDD XSPICE CodeModel for s non-linear resistor model: (a) synthesis flow diagram showing modelling stages, (b) test circuit and typical simulation data.

```
  OUTPUT(nRp_nRn)=1.0/R0/(V2*((k2+k3*V2)*
    V2+k1)+1.0)*V1+Q0;
  OUTPUT(nV2_gnd)=0.0+Q1;
  PARTIAL(nRp_nRn,nRp_nRn)=1.0/
    ((k1+(k2+k3*V2)*V2)*V2+1.0)/R0+cQ0;
  PARTIAL(nRp_nRn,nV2_gnd)=-(V2*(k3*V2+k2)
    + k1+V2*(2.0*k3*V2+k2))*V1/
    Xpow((V2*(k3*V2+k2)+k1)*
    V2+1.0,2.0)/R0;
 } else {
  ac_gain00.imag=(0.0)*RAD_FREQ;
  AC_GAIN(nRp_nRn,nRp_nRn)=ac_gain00;
  ac_gain01.real=-(V2*(k3*V2+k2)+k1+
    V2*(2.0*k3*V2+k2))*V1/Xpow((V2*
    (k3*V2+k2)+k1)*V2+1.0,2.0)/R0;
  ac_gain01.imag=0.0;
  AC_GAIN(nRp_nRn,nV2_gnd)=ac_gain01;
 }
}
```

Listing 1. The XSPICE synthesised C code for a non-linear resistor model (file VcontrolR.mod).

Qucs-S EDD handles non-linear capacitors $C(V)$ by evaluating capacitor current from EDD branch charge, expressed as

$$I(t_1) = \frac{\partial Q(V(t_1 - \Delta t))}{\partial V(t_1 - \Delta t)} \cdot \frac{V(t_1) - V(t_1 - \Delta t)}{\Delta t} \quad (4)$$

where $\Delta t = TIME - T(1)$, $TIME$ is the current simulation time and $T(1)$ the time at the previous simulation time step. The simple test circuit shown in Figure 3 illustrates the effect that a non-linear capacitance has on the transfer function of an RC low pass filter network when the DC bias is swept through a series of values. Notice in this example that the DC voltage across the capacitor is used to change $C(V1))$ rather than a separate control branch. Listing 2 gives the core of the synthesized XSPICE CodeModel for the non-linear capacitor.

```
/* XSPICE codemodel nonLinQ */
/* auto-generated template */
#include <math.h>
#include "xspice_mathfunc.h"
void cm_nonLinQ(ARGS)
{
 Complex_t ac_gain00;
 static double c0,k1,k2;
 static double V1,V1_old;
 double Q0, cQ0;
 double delta_t;
 if(INIT) {
  c0 = PARAM(c0);
  k1 = PARAM(k1);
  k2 = PARAM(k2);
 }
 if (ANALYSIS != AC) {
  if (TIME == 0) {
   V1_old = V1 = INPUT(P1_P2);
   Q0=0.0;
   cQ0=0.0;
  } else {
   V1 = INPUT(P1_P2);
   delta_t=TIME-T(1);
   Q0 = (( k1*V1+k2*(V1*V1)+1.0)*c0)*
        (V1-V1_old)/(delta_t+1e-20);
   cQ0 = (( k1*V1+k2*(V1*V1)+1.0)*c0)/
         (delta_t+1e-20);
   V1_old = V1;
  }
  OUTPUT(P1_P2) = 0.0 + Q0;
  PARTIAL(P1_P2,P1_P2) = 0.0 + cQ0;
  } else {
  ac_gain00.real = 0.0;
  ac_gain00.imag = (c0*( k1*V1+k2*(V1*V1)+
    1.0))*RAD_FREQ;
  AC_GAIN(P1_P2,P1_P2) = ac_gain00;
 }
}
```

Listing 2. The XSPICE synthesised C code for a non-linear capacitor model.

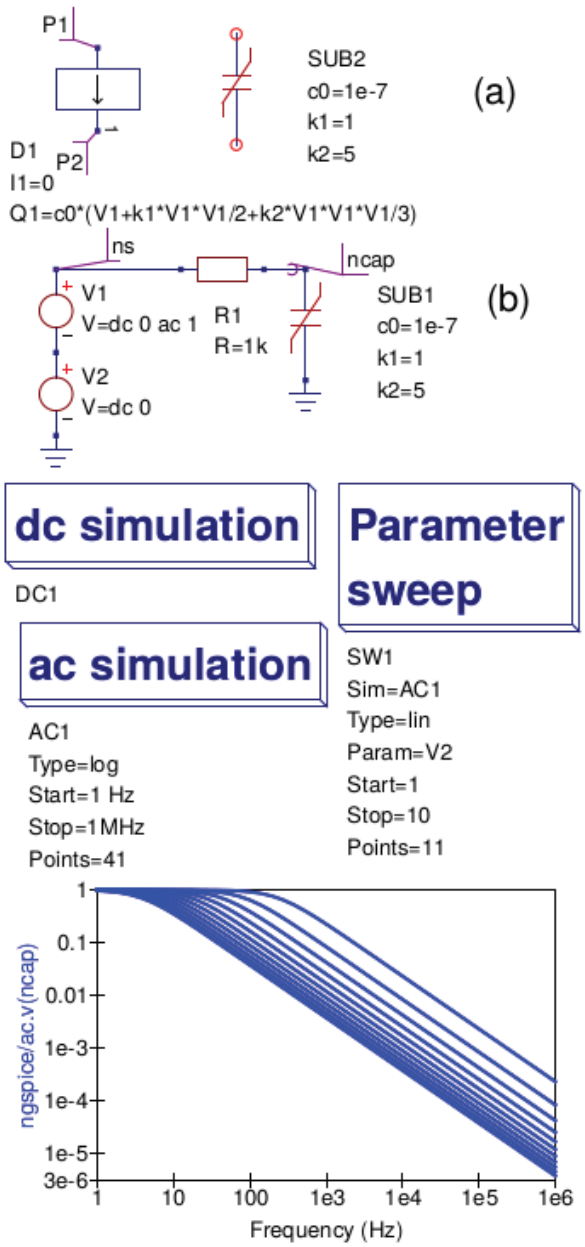

Fig. 3. EDD/CodeModel non-linear compact model example: (a) EDD model and Qucs-S schematic symbol, (b) RC low pass filter test circuit with swept capacitance bias control.

IV. A Basic Qucs-S Long Channel Version of the EPFL-EKV v2.6 Compact Device Model for an nMOS Transistor

The Qucs-S XSPICE synthesized CodeModel examples introduced in the previous sections demonstrate the fundamental concepts needed to take full advantage of the improved modelling power and flexibility now built into the Qucs-S circuit simulation and modelling package. When modelling semiconductor devices the complexity of the task requires adoption of a modular structure where individual parts of a model can be developed, tested and finally assembled to produce a finished model. Qucs-S encourages such an approach by the use different modelling features, including Qucs EDD, SPICE B type sources, XSPICE CodeModel synthesized blocks and appropriate linear components linked together via a set of internal nodes within the body of a subcicuit. A basic long channel EPFL EKV v2.6 nMOS model can be represented by the following set of simplified compact semiconductor device equations 5. A detailed description of the meaning of the variables in these equations is given in reference [8]. In this paper equations 5 to 13 are used to demonstrate how EDD, and XSPICE synthesized code models can be combined to develop a basic functional, static I/V model of an nMOS transistor:

$$Vg' = Vg - Vto + \phi + \beta \cdot \sqrt{\phi} \tag{5}$$

$$Vp = Vg' - \phi - \gamma \cdot (\sqrt{(Vg' + (\gamma/2)^2)} - \gamma/2) \tag{6}$$

$$n = 1 + \gamma/(2 \cdot \sqrt{(Vp + \phi + 4 \cdot vt)}) \tag{7}$$

$$beta = (kp \cdot w/l)/(1 + \theta \cdot Vp) \tag{8}$$

$$x1 = (Vp - Vs)/vt \tag{9}$$

$$x2 = (Vp - Vd)/vt \tag{10}$$

$$iff = \ln(1 + \exp(x1/2)) \cdot \ln(1 + \exp(x1/2)) \tag{11}$$

$$ir = \ln(1 + \exp(x2/2)) \cdot \ln(1 + \exp(x2/2)) \tag{12}$$

$$ids = 2 \cdot n \cdot \beta * vt^2 \cdot (iff - ir) \tag{13}$$

Figures 4 and 5 present the normal sequence for converting Qucs-S EDD into XSPICE C code blocks. In order to represent the EDD version of the EKVv2.6 nMOS transistor static I/V equations (5) in an easily readable format that can be fitted onto a double column page four EDD, labeled (a) to (d) are used to build the long channel I/V model.

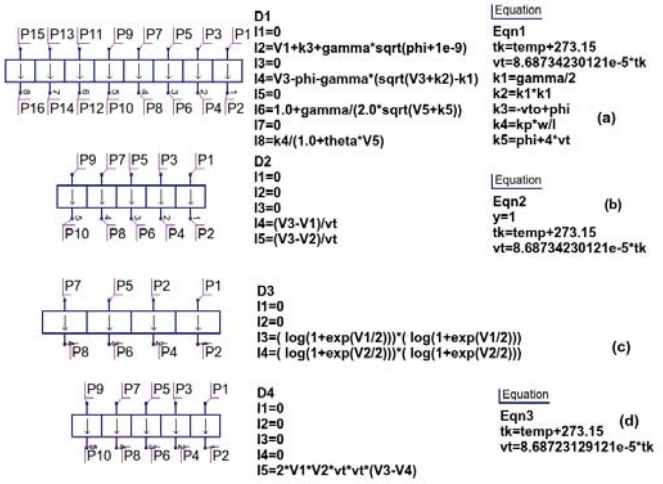

Fig. 4. EKV2.6 Qucs-S EDD long channel static I/V model for a long channel nMOS transistor: $D1$ to $D4$ currents In represent the equations listed in equations 5 to 13.

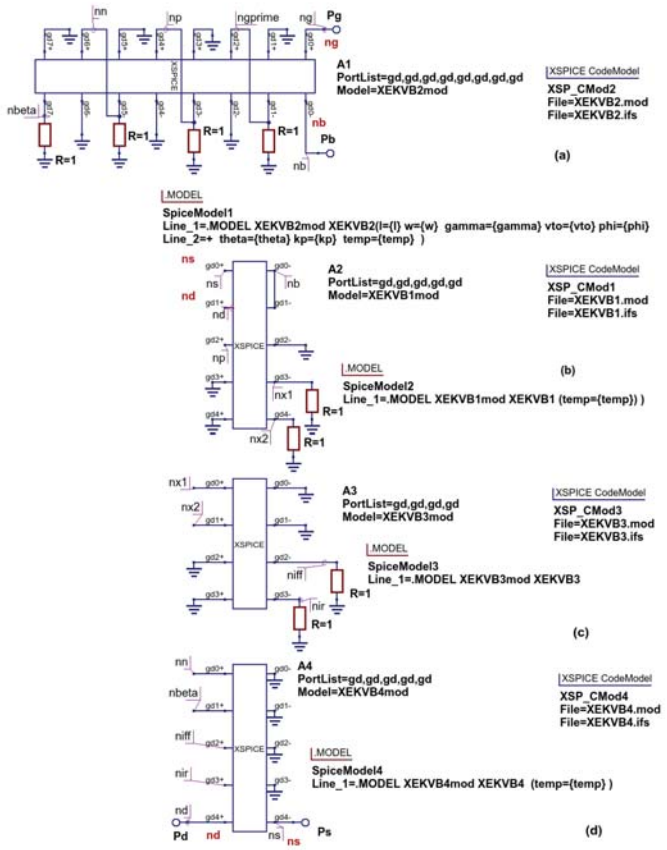

Fig. 5. EKV2.6 Qucs-S XSPICE long channel static I.V model for a nMOS transistor: nodes labeled with the same name are connected.

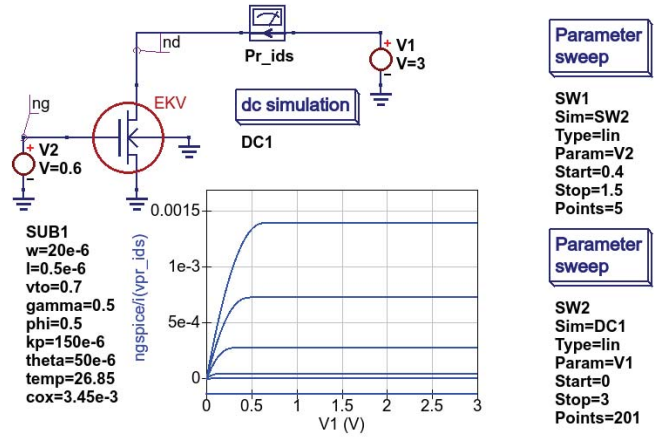

Fig. 6. EKV2.6 Qucs-S long channel static I/V model test bench and typical simulated I/V output characteristics.

V. Modelling RF Passive Components with XSPICE CodeModels

A high percentage of EDD models consist of two main sections. Firstly, an inner core which represents the primary and secondary physical effects that determine the functionality of the device, and secondly an outer part which acts as a wrapper around the inner core connecting it to the outside world via passive components. These components often represent parasitic effects which are associated with the device signal paths from the inner core to the external connection device pins. At low frequencies parasitics normally have no significant effect on simulation performance and are usually neglected. However, at RF and higher frequencies, the reverse is true, making it essential that the parasitic components are included, otherwise significant simulation errors occur. The recent XSPICE CodeModel improvements to EDD modelling allow Qucs-S, for the first time, to include difference equations in compact simulation models. Equation 14 shows an example where Qucs-S models a non-linear capacitor current derived from the time derivative of stored branch charge written as a difference equation. In the case of an inductor, an equivalent current equation is

$$I(t_1) = I(t_1 - \Delta t) + \frac{V(t_1)\Delta t}{L} \tag{14}$$

The expression for current $I(t_1)$ given in equation 14 is for an ideal component without any high frequency parasitics. In contrast the electrical network shown in Figure 7 (a) is an RF inductance with parasitics Rs, Cp, Rp added. Equation-defined models for complex RCL networks can be easily handled by Qucs-S using time dependent difference equations to transform first order differential equations to simpler algebraic equations. Equation 15 introduces the RF inductance algebraic/difference equations that express the current I flowing in the model in terms of applied voltage V and component values.

$$I(t_1) = I_L(t_1) + IC_p(t_1) + IR_p(t_1) \tag{15}$$

where

$$IL(t_1) = (V(t_1) + p1 \cdot IL(t_1 - \Delta t))/(Rs + p1) \tag{16}$$

$$p1 = L/\Delta t \tag{17}$$

$$ICp(t_1) = (V(t_1) - \Delta t)/\Delta t \tag{18}$$

$$IRp(t_1) = V(t_1)/Rp \tag{19}$$

The expression for $I(t_1)$ is used by Qucs-S as the basis for an XSPICE CodeModel. Listing 3 introduces part of the XSPICE C code derived from the schematic and current function given in Figure 7. Similarly, Figure 7 (c) presents a simple test bench and an AC performance curve for the RF inductor. Notice at frequencies above roughly 1e8Hz the inductor shows distinct capacitive characteristics. This is a primary effect caused by the parasitic components Rs, Cp, Rp.

```
if(ANALYSIS == DC) {
 vind = INPUT(nind);
 iind_dc = vind/(PRS+1e-7) +
    vind/(PRP+1e-7);
 OUTPUT(nind) = iind_dc;
 PARTIAL(nind, nind) = 1/(PRS+1e-7) +
    1/(PRP+1e-7);
 iind_dc_old = iind_dc;
}
if(ANALYSIS == AC) {
 p1 =(PRS*PRS+RAD_FREQ*RAD_FREQ*
    lvalue*lvalue);
 ac_gain.real = PRS/p1+1/PRP;
 ac_gain.imag= RAD_FREQ*PCP - (lvalue *
    RAD_FREQ)/p1;
 AC_GAIN(nind,nind) = ac_gain;
}
```

```
if(ANALYSIS == TRANSIENT) {
 vind = INPUT(nind); delta = TIME - T(1);
 p1 = lvalue/delta;  p2 = p1+PRS;
 iind = (iind_old*p1/p2 + vind)/p2;
 icp = icp_old+PCP*(vind - vind_old)/
   delta;
 irp = vind/PRP;
 OUTPUT(nind) = iind + icp + irp;
 PARTIAL(nind, nind) = 1/p2 +
   PCP/delta + 1/PRP;
 vind_old = vind;
 iind_old = iind; icp_old  = icp;
}
```

Listing 3. Part of the XSPICE synthesised C code for an RF inductance model.

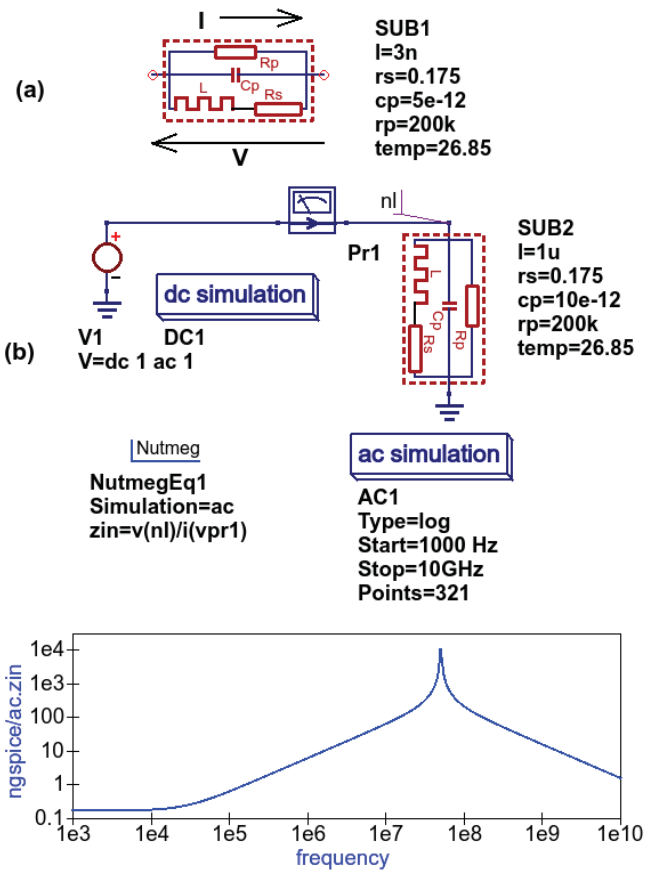

Fig. 7. RF inductance Equation-Defined model: (a) schematic symbol and (b) AC test bench and performance data

VI. Extending Equation-Defined Component and Device Models with User Defined Functions

When first released for modelling the Qucs EDD was supported by a set of operators and mathematical functions similar to those common to packages like Octave. Over the last ten years the range of these functions has been steadily improved, till today, they mirror a high percentage of the functions defined in the Verilog-A standard [3]. With the release of the Qucs-S version of EDD, and the supporting XSPICE CodeModel tools, user defined functions have also been added to the Equation-Defined modelling tool set. Qucs-S user defined functions are stored by Qucs-S as a list in a library which can be included, when needed, in XSPICE CodeModels. The examples in Listing 4 give an indication of the Qucs-S XSPICE user defined function syntax:

```
#define step(x) ((x)>0.0?1.0:(((x)==0)?
      0.5:0.0))
#define Xpow(x,p) pow(fabs((x)),(p))
#define limexp(x) ((x)<80.0?exp(x):
     (exp(80.0)*((x)-80.0)))
```

Listing 4. Example user-defined XSPICE CodeModel functions.

VII. Conclusion

Qucs EDD were introduced roughly ten years ago as a simple to use interactive non-linear modelling tool. Although Verilog-A has emerged as one of the dominant hardware description languages for compact model development the strong link between EDD and Verilog-A models has ensured that EDD modelling retains it's popularity amongst the compact modelling community. This paper introduces a number of improvements that have taken place in EDD modelling as part of the release of the Qucs-S version of Qucs. Particular emphasis being given to (1) automatic syntheses of C code compact device models from Qucs EDD schematics, with a newly developed XSPICE CodeModel synthesis tool, (2) the introduction of user defined functions and (3) subcircuit packaging of hybrid XSPICE CodeModel/EDD/linear component models. To demonstrate the utility and performance of the improved Qucs-S EDD modelling tools a number of different model fragments are described and simulated with Qucs-S/Ngspice. Given the success of the original Qucs EDD it is expected that the recent improvements in the Qucs-S non-linear modelling will over a period of time have a corresponding impact.

References

[1] S. Jahn, and M.E. Brinson, "Interactive Compact Modeling Using Qucs Equation-Defined Devices", *Int. J. Numer. Model.* 2008, vol 21, pp. 335-349.

[2] M. Brinson, and S. Jahn,"Qucs: A GPL software package for simulation, compact device modelling and circuit macromodelling from DC to RF and beyond", *Int. J. Numer. Model.* 2009, vol 22, pp. 297-319-349.

[3] Accellera, "Verilog-AMS Language Manual", Version 2.3.1, 2009. [Online] Available: http://www.accellera.org.

[4] M. E. Brinson, "SPICE to QucsStudio via Qucs: An international attempt to develop a freely available GPL RF design, compact modeling, simulation, data processing and manufacturing development environment for engineers", MOS-AK/GSA Workshop, March 16-18, 2012, India. [Online] Available: http://www.mos-ak.org/india/presentations/Brinson_MOS-AK_India12.pdf.

[5] L. Lemaitre and B.Gu, "ADMS- a fully customizable Verilog-AMS compiler approach," MOS-AK, Meeting, Montreux, 2006, [Online] Available: http://www.mos-ak.org/montreux/posters/17_Lemaitre_MOS_AK06.pdf.

[6] P Nenzi and H. Vogt, "Ngspice-26 (Next generation SPICE version 26), 2015. [Online] Available: http://ngspice.sourceforge.net.

[7] Sandia National Laboratories US., "Xyce parallel electronic simulator version 6.4," 2016, [Online] Available: http://xyce.sania.gov.

[8] M. Bucher, C. Lallememt, C. Enz. F Theodoloz and F. Krummencher, "The EPFL-ELV MOSFET Model Equations for Simulation", Technical Report, Electronics Laboratories, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, Model Version 2.6, June 1997.

[9] SPICE OPUS, "SPICE OPUS: analog circuit simulator engine specifically suited for optimization tools, based on SPICE 3jf and XSPICE.", Faculty of Electrical Engineering at the University of Ljubljana, Slovenia., 2016. [Online] Available: http://fides.fe.uni-lj.si/spice/download/.

[10] M. E.Brinson and V. Kuznetsov, "A new approach to compact semiconductor modelling with Qucs Verilog-A analogue module synthesis, "*Int. J. Numer. Model.*, 2016, vol. 29. pp. 1070-1088, November/December.

[11] F. Cox III, W.Kahn, J, Murry and S. TYnor, "Code-level modeling in XSPICE," in *Circuits and Systems, 1992 ISCAS '92, Proceedings., 1992 IEEE International Symposium on.*, vol.2 , May 1992, PP. 871-874.

[12] GiNaC, "GiNaC is Not a CAS: " C++ library, designed to allow the creation of integrated systems that embed symbolic manipulations together with more established areas of computer science.", [Online] Available: http://www.ginac.de/.

[13] M. E. Brinson and V. Kuznetsov,'Spice4qucs-help documentation, user manual and reference material. [Online] Available: http://qucs-help.readthedocs.org/en/spive-4qucs.

[14] M. E. Brinson and V. Kuznetsov, Qucs-0.0.19S: A new open-source circuit simulator and its application for hardware design", in *International Siberian Conference on Control and Communications (SIBCON)*, May 2016, pp 1-5.

Mike Brinson received a first class honours BSc degree in the Physics and Technology of Electronics from the United Kingdom Council for National Academic Awards in 1965, and a PhD in Solid State Physics from London University in 1968. Since 1968 Dr. Brinson has held academic posts in Electronics and Computer Science. From 1997 till 2000 he was a visiting Professor of Analogue Microelectronics at Hochschule, Breman, Germany. Currently, he is a visiting Professor at the Centre for Communication Technology Research, London Metropolitan University, UK. He is a Chartered Engineer (CEng) and a Fellow of the Institution of Engineering and Technology (FIET), a Chartered Physicist (CPhys), and a member of the Institute of Physics (MInstP). Prof. Brinson Joined the Qucs project development team in 2006, specializing in device and circuit modeling, testing and document preparation.

Vadim Kuznetsov was born in Kaluga, Russia in 1988. He received dipl. engineer degree from Moscow Bauman State University (BMSTU) in 2010. He received PhD degree from Higher school of Economics in 2014. He is Associate Professor of Electronic Engineering department of Kaluga Branch of BMSTU. His research field is electrostatic discharge simulation methods. His field of interest is electronic design automation (EDA) CAD open-source software development. He is core member of Qucs circuit simulator development team.