INTERNATIONAL JOURNAL OF ELECTRONICS, Published online on 29 July 2017.
http://dx.doi.org/10.1080/00207217.2017.1357764. Volume 105, 2018, issue 3, pp. 412-425.

RESEARCH ARTICLE

Extended behavioural device modelling and circuit simulation with Qucs-S

M. E. Brinson[a] and V. Kuznetsov[b]

[a]Centre for Communications Technology, London Metropolitan University UK;
[b]Bauman Moscow State Technical University, Kaluga branch, Russia

ABSTRACT
Current trends in circuit simulation suggest a growing interest in open source software which allows access to more than one simulation engine while simultaneously supporting schematic drawing tools, behavioural, Verilog-A and XSPICE component modelling and output data post-processing. This paper introduces a number of new features implemented in the "Quite universal circuit simulator - SPICE variant" (Qucs-S), including structure and fundamental schematic capture algorithms, at the same time highlighting their use in behavioural semiconductor device modelling. Particular importance is placed on interaction between Qucs-S schematics, Equation-Defined Devices, SPICE B behavioural sources and HDL scripts. The multi-simulator version of Qucs is a freely available tool that offers improved modelling and simulation features compared to those provided by legacy circuit simulators. The performance of a number of Qucs-S modelling extensions are demonstrated with a GaN HEMT compact device model and data obtained from tests using the Qucs-S/Ngspice Xyce, SPICE OPUS multi-engine circuit simulator.

KEYWORDS
Qucs-S; SPICE; Ngspice; Xyce; SPICE OPUS; circuit simulation; compact device modelling; non-linear behavioural models; XSPICE CodeModels; Verilog-A

1. Introduction

General Public Licence (GPL) circuit simulators that can be traced back to SPICE 3f5 (Johnson, Quarles, Newton, Pederson & Sangiovanni-Vincentelli 1992) include Ngspice (Ngspice, 2016), SPICE OPUS (SPICE OPUS, 2016) and Xyce (Xyce, 2016). Although these offer a high level of compatibility with SPICE 3f5 they have evolved as separate entities, implying differences in simulation capabilities and device models, for example Xyce is a circuit simulator written independently from SPICE 3f5 which includes single and multi-tone RF Harmonic Balance circuit simulation, SPICE OPUS provides extensive optimization capabilities and Ngspice offers mixed-level/mixed-signal circuit simulation. To take advantage of these developments in GPL circuit simulation a new version of the "Quite universal circuit simulator - SPICE variant" (Qucs-S) has been released (Brinson & Kuznetsov, 2016; Kuznetsov, 2016). The Qucs-S package allows access to Ngspice, SPICE OPUS and Xyce via simulation control instructions added to a circuit schematic. In the "S" variant these instructions

CONTACT M. E. Brinson. Email: mbrin72043@yahoo.co.uk

either take the form of icons or high level programming style scripts. Central to the use of a multi-engine circuit simulator is the idea that a schematic acts as a specification of the circuit under test and a launching pad for simulation, regardless of which of the available simulators is chosen. This in turn implies that Qucs-S must be capable of translating the information held on a schematic into any of the Ngspice, SPICE OPUS, Xyce or Qucs Qucsator netlist formats. This process is more complex than simply the translation of individual component symbols into equivalent netlist statements due to the fact that not all simulation models and capabilities are common to each GPL simulator. Moreover, there are specific non-linear models, particularly the Qucs Equation-Defined Device, which are not implemented by GPL SPICE simulators. In such cases the translation process becomes more involved, requiring the synthesis of equivalent circuit functions from the available SPICE components. This paper introduces the technology needed for the simulation of Qucs-S circuit schematics with the Ngspice, SPICE OPUS and Xyce circuit simulators, including an outline of the synthesis of the different SPICE netlist dialects. It also describes an extended range of behavioural modelling components which become available through the combination of Qucs schematics with SPICE simulators, stressing the use of non-linear component models and hardware description scripts in the synthesis of compact device models. A GaN semiconductor HEMT model is described and a series of test circuits presented to illustrate the interaction between Qucs-S extended behavioural modelling, circuit simulation and output data processing.

2. The structure of Qucs-S schematics and interaction of simulation backend and frontend

Qucs-S synthesises a Qucs circuit schematic into an Ngspice, SPICE OPUS or Xyce SPICE netlist or into a Qucs Qucsator netlist. As a starting point a schematic is drawn using component symbols from three groups:

- "Unified circuit elements and simulation symbols". These are common to all SPICE kernels and the Qucsator simulation engine. Within this group are passive element, current and voltage sources, semiconductor devices, Equation-Defined Devices (EDD), and a number of simulation icons.
- "SPICE circuit elements and SPICE command symbols". These symbols provide access to all Ngspice, SPICE OPUS and Xyce component and simulation features, including passive components, sources, non-linear devices, command statements, XSPICE non-linear devices. and SPICE only simulation icons. This symbol group includes new special script simulation types. These are designed to allow passing of SPICE code to output data post-processing routines.
- "The Qucsator element symbol set". This contains Qucs specific RF and microwave devices and RF simulation icons (for example S-parameter analysis). These symbols are only available to the Qucsator simulation engine.

The relationship between a Qucs-S schematic and the different component categories is illustrated in the Figure 1. Figure 1 presents a schematic which includes a mixture of SPICE devices and parameters and unified simulation icons and components, including .MODEL, .PARAM and Nutmeg post-processor equation statements.

The data flow diagram illustrated in Figure 2 outlines the sequence followed by a Qucs-S software subsystem, called spice4qucs, when simulating the performance of a circuit; firstly the Qucs-S graphical user interface (GUI) generates a circuit netlist,

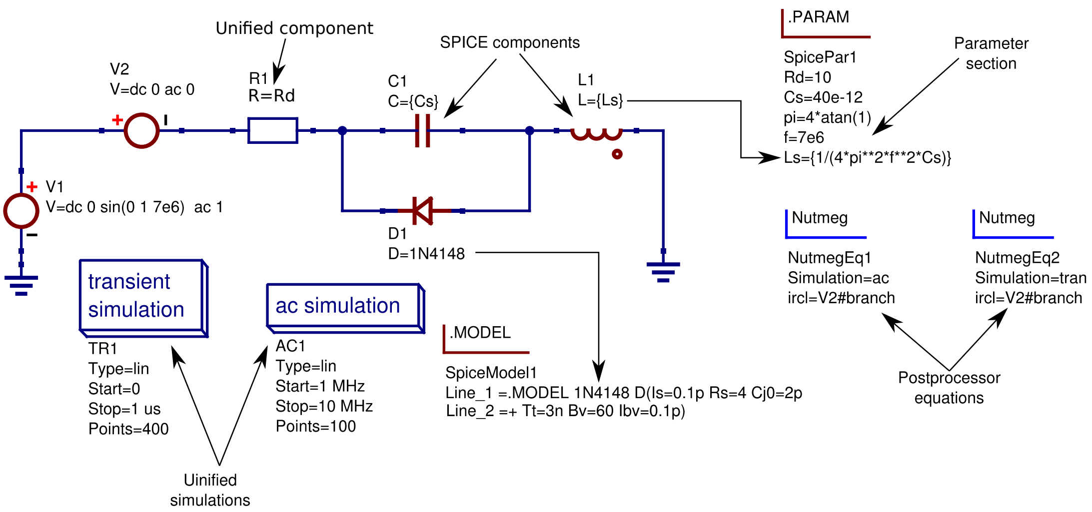

Figure 1. A Qucs-S RCL and diode circuit schematic with netlist sections labelled: SPICE component set.

secondly a simulation engine is launched, as a separate program, and the netlist simulated, and finally circuit performance data are output as a set offiles. These files are converted into a Qucs-S XML data set in preparation for post-simulation processing and visualisation using the Qucs-S visualization subsystem (or by the Octave Eaton, Bateman, Hauberg & Wehbring, 2016) scientific programming package). The format of the output data files depends on which circuit simulator is selected, for example raw-SPICE3f5 text or binary format files for Ngspice and SPICE OPUS and the XYCE STD format files for Xyce.

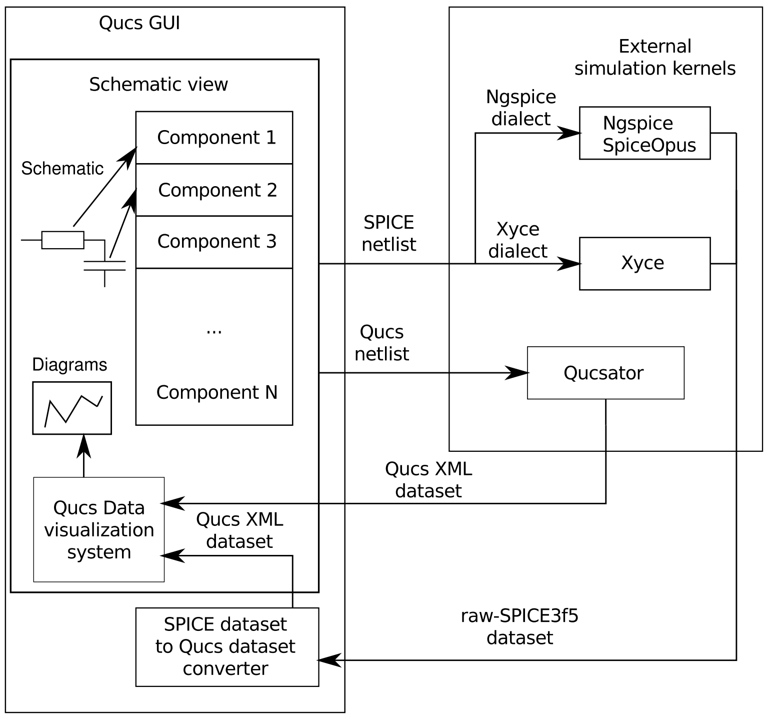

Figure 2. Spice4qucs subsystem dataflow block diagram.

3. Qucs-S netlist synthesis algorithms and script controlled simulations

A synthesised SPICE netlist consists of four sections where the first three are similar in structure but different content. The primary purpose of section four is to introduce new device modelling and simulation features specific to each of the Ngspice, SPICE OPUS and Xyce netlist dialects. These four sections can be summarized by:

- ”Section 1; parameters”. This section lists circuit parameters and simulation directives. These consist of a list of expressions formed from numeric and algebraic quantities. Simulation options and initial conditions are placed in this section.
- ”Section 2; device netlist”. This section contains the netlist for passive components and active devices, plus user defined subcircuits. Each entry describes a device type, connection and characteristic parameters.
- ”Section 3; .MODEL directives”. This section lists all .MODEL statements referenced in ”Section 2”.
- ”Section 4; simulation control and post-processing” This section introduces the commands for starting the simulation process and actions post-simulation output data processing equations. Xyce has no postprocessor and this section is omitted.

Qucs-S synthesizes a SPICE netlist by scanning the information drawn on a circuit schematic. This is done as a sequential process. Algorithm 1 introduces the sequence employed to generate a Ngspice netlist. Part of this process involves finding and attaching simulation output data post-processing instructions to each of the different Ngspice, or SPICE OPUS, simulation icons embedded on a circuit schematic.

The Ngspice/SPICE OPUS and Xyce netlist generation process are very different, as indicated by Algorithm 2. The Xyce circuit simulator does not support Nutmeg post-simulation data processing but uses an extended form of SPICE .PRINT statement.

Embedding Ngspice, SPICE OPUS and Xyce simulation control commands with circuit schematics ensures that users have access to all the simulation features implemented by the different simulation engines. Such an approach adds new simulation capabilities that can be implemented and controlled by simulator scripts/directives. Two types of script controlled simulation are allowed by Qucs-S:

- ”Type 1; Nutmeg scripts for use with Ngspice and SPICE OPUS”. These are based on the Ngspice and SPICE OPUS extended versions of the original SPICE 3f5 Nutmeg script language. The Nutmeg script language syntax has features common to all high level programming/scripting languages (operators and loops etc.) and a full set mathematical functions. Generated Nutmeg code is placed between the SPICE 3f5 .control ..... .endc statements located at the end of a netlist.
- ”Type 2; Xyce scripts”. These scripts form part of a Xyce netlist, giving direct access to less used Xyce statements, like .MEASURE.

The Qucs-S simulation control scripts have special properties which allows them to hold and process SPICE simulation statements. This extended facility has been designed to allow users the opportunity to implement new, non-standard SPICE, simulation routines without having to manually patch a circuit simulator C or C++ code. The last two data flow diagrams, Figures 3 and 4, illustrate how Qucs-S script controlled simulation works, giving particular emphasis on the differences between the simulation engines adopted by Qucs-S.

Algorithm 1: Ngspice netlist building algorithm

```
Data: Qucs Schematic
Data: SPICE netlist filename
Result: SPICE netlist
begin
    foreach (Component in Schematic) do
        if (Component is Parameter or directive) then
            Netlist ← Component.getSpiceExpression()
        end
    end
    foreach (Component in Schematic) do
        if (Component is Device) then
            Netlist ← Component.getSpiceNetlist()
        end
    end
    foreach (Component in Schematic) do
        if (Component is Model    directive) then
            Netlist ← Component.getSpiceModel()
        end
    end
    // begin of .control section
    foreach (Component in Schematic) do
        if (Component is Simulation) then
            Netlist ← Component.getSimulationScript()
            foreach (Component in Schematic) do
                // find equations attached to simulation
                if (Component is Equation) then
                    Netlist ← Component.getEquation()
                end
            end
        end
    end
    // end of .control section
end
```

4. Qucs-S behavioural Equation-Defined Device (EDD) and SPICE B source modelling and circuit simulation

The Qucs-S circuit simulator implements a sixteen-terminal, eight-port EDD with non-linear current and charge properties,designed for building behavioural compact device models (Brinson & Jahn ,2009; Jahn & Brinson , 2008). The EDD are not implemented by Ngspice, SPICE OPUS or Xyce however, making direct translation from the Qucs schematic/netlist difficult to achieve. The nearest SPICE equivalent to an EDD is the SPICE B component.In reality, these two components are not very compatible because the SPICE B component does not model internal stored charge. Qucs-S resolves this limitation by synthesising a replacement component block which has the same current, voltage and charge properties as EDD, but is constructed from a number SPICE 3f5 components connected as a functional macromodel. The synthesis

Algorithm 2: XYCE netlist building algorithm

Data: Qucs Schematic
Data: SPICE netlist filename
Result: SPICE netlist
begin
  foreach *Simulation* **do**
    foreach *(Component in Schematic)* **do**
      if *(Component is Parameter or directive)* **then**
        Netlist ← Component.getSpiceExpression()
      end
    end
    foreach *(Component in Schematic)* **do**
      if *(Component is Device or Model)* **then**
        Netlist ← Component.getSpiceNetlist()
      end
    end
    Netlist ← Simulation.getSpiceNetlist()
  end
end

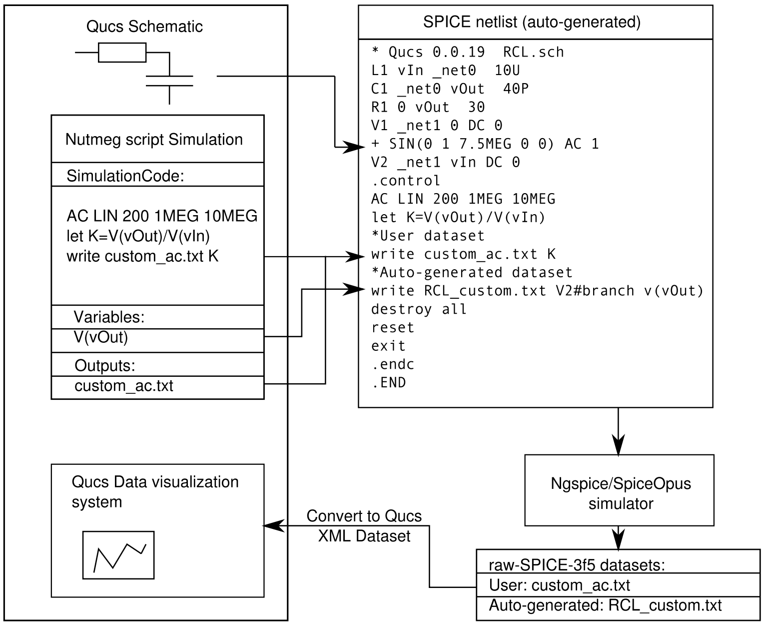

Figure 3. Dataflow diagram for the Nutmeg script simulation mode.

of a Qucs-S schematic to a SPICE style netlist is transparent to Qucs-S users, taking place at the start of circuit simulation. As the Ngspice, SPICE OPUS and Xyce netlist formats are also not identical the synthesised SPICE netlist reflects the requirements of each of the different simulation engines. The combination of Qucs-S schematics, EDD and SPICE B voltage sources, linear components, Nutmeg and Xyce scripts, and the Qucsator, Ngspice, SPICE OPUS and Xyce circuit simulator engines makes possible a freely available open source device modelling and circuit simulation tool which gives access to software with significantly improved performance when compared to SPICE

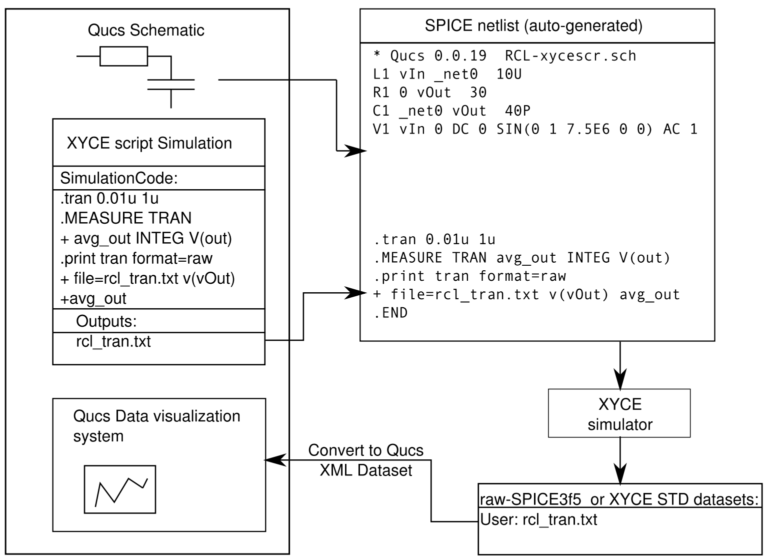

Figure 4. Dataflow diagram for the Xyce script simulation mode.

2g6 and 3f5lt is particularly useful for building experimental interactive compact models for new or emerging technologies. The remainder of this paper introduces a number ofthe most important new modelling and simulation techniques that are now possible with Qucs-S. To demonstrate these innovative approaches to interactive compact device modelling a GaN HEMT model is introduced (Angelov, Zirath & Rorsman, 1992; Angelov, Bengtsson & Garcia , 1996), and its performance discussed, with particular emphasis being placed on the modelling and simulation capabilities added by the new Qucs-S extension features. Table 1 lists the GaN HEMT model parameters and their default values, for a simplified Angelov GaN HEMT compact device model built around a schematic constructed from Qucs EDD, SPICE B voltage sources and linear components. The GaN HEMT static I/V and dynamic charge (Q)

Table 1. Typical model parameters for a simplified Angelov GaN HEMT model

Name	Description	Default	Name	Description	Default
IPK0	Current at max transcon (A)	0.05	VPKS	Voltage at max transcon (V)	-0.2 V
P1	Ids poly coefficient	0.8	P2	Ids poly coefficient	0.0
P3	Ids poly coefficient	0.0	B1	p1 unsaturated coefficient	0.1
B2	P2 unsaturated coefficient	4.0	ALPHAR	Saturation parameter	0.1
ALPHAS	Saturation parameter	1.0	VBS2	Surface breakdown parameter	0.0
LAMBDA	Channel length modulation	0.01	DVPKS	Gate voltage at peak gm (V)	0.2
VTR	Threshold voltage (V)	50.0	LSB0	Soft breakdown parameter	0.0
Rg	Gate resistance (Ω)	0.05	Rs	Source resistance (Ω)	0.05
Rd	Drain resistance (Ω)	0.05	Lg	Gate inductance (H)	1e-10
Ls	Source inductance (H)	1e-10	Ld	Drain inductance (H)	1e-10
P10	Cap poly coefficient	0.48	P11	Cap poly coefficient	0.21
P20	Cap polynomial coefficient	0.03	P21	Cap poly coefficient	0.21
P40	Polynomial coefficient	0.48	P41	Polynomial coefficient	0.25
P111	Polynomial coefficient	0.008	RI	G-S resistance (Ω)	1.00
RGD	G-D resistance (Ω)	0.01	CGDPI	G-D pinch-off cap (F)	200e-15
CGSPI	G-S pinch-off cap (F)	15e-15	CGS0	G-S cap (F)	3500e-15
CGD0	G-D cap (F)	378e-15	CDS	D-S cap (F)	800e-15
IJ	Gate fwd saturation I (A)	5e-4	PG	Gate current parameter	15.0
VJG	Gate current parameter	0.7			

physical characteristics are determined by the compact model equations:

$$P1m = P1 \cdot (1 + B1/\cosh^2(B2 \cdot V(nds)))$$

$$V_{pkm} = const1 + DVPKS \cdot \tanh(ALPHAS \cdot V(nds)) - VSB2 \cdot (V(ndg) - VTF)^2$$
$$\Psi = P1m \cdot V(ndiff) + P2 \cdot V(ndiff)^2 + P3 \cdot V(ndiff)^3$$
$$\alpha = ALPHAR + ALPHAS \cdot (1 + \tanh(\Psi))$$
$$Ids = IPK0 \cdot (1 + \tanh(\Psi)) \cdot \tanh(\alpha \cdot V(nds)) +$$
$$(1 + LAMBDA \cdot V(nds) + LSB0) \cdot \exp(V(ndg) - VTR)$$
$$Igd = IJ * (exp(PG * \tanh(2 * (V(nRgdD, gate) - VJG)))) -$$
$$exp(PG * \tanh(-2 * VJG))$$
$$Igs = IJ * (\exp(PG * \tanh(2 * (V(nRiS, gate) - VJG)))) - exp(PG * \tanh(-2 * VJG))$$
$$\Psi 1 = P10 + P11 * V(nRiS, gate) + P111 * V(nds)$$
$$\Psi 2 = P20 + P21 * V(nds)$$
$$\Psi 3 = P30 + P31 * V(nds)$$
$$\Psi 4 = P40 + P41 * V(nRgdD, gate) + P111 * V(nds)$$
$$lc1 = \log(\cosh(V(\Psi 1)))/(P11 + 1e-20)$$
$$lc4 = \log(\cosh(V(\Psi 4)))/(P41 + 1e-20)$$
$$th2 = \tanh(V(\Psi 2)), th3 = \tanh(V(\Psi 3))$$
$$Qgs = CGSPI * V(nRiS, gate) + CGS0 \cdot (V(nRiS, gate) + V(lc1)) \cdot V(th2)$$
$$Qgd = CGDPI * V(nRgdD, gate) + CGD0 \cdot (V(nRgdD, gate) + V(lc4) \cdot V(th3)$$
$Qds = CDS * V(drain, source)$, where
$V(nds) = V(drain, source)$, $V(ndg) = V(drain, gate)$, $V(ngs) = V(gate, source)$,
$Vconst1 = VPKS - DVPKS$, and $V(ndiff) = V(ngs) - Vnpkm$.

The structure and properties of Qucs-S EDD and SPICE B voltage sources are designed to allow device equations of the form listed above, to be easily converted into a Qucs-S subcircuit schematic, see Figure 5. At the start of circuit simulation Qucs-S converts the schematic of the circuit under test into a netlist with a format suitable for input to the chosen simulation engine. As indicated previously the Qucs-S software translates/synthesises the information encoded by a schematic into the required netlist. The current software allows both Qucs EDD and SPICE B device models to be included in a circuit schematic at the same time, allowing the best Qucs or SPICE compact modelling features to be combined, yielding efficient functional models of new or existing devices. The resulting synthesised netlist has a SPICE 3f5 like format which varies according to the extended specification adopted by each simulator. To illustrate a typical example the Qucs-S generated netlist for the GaN HEMT subcircuit model shown in Figure 5 is given in Figure 6. Notice that the Qucs-S EDD charge functions (DxxQyy) are synthesised from SPICE non-linear and linear components. A typical GaN HEMT DC test bench circuit and simulated DC output characteristics are shown in Figure 7, where the DC simulation sequence is controlled by the Qucs-S DC and Parameter sweep icons. These icons work in the same way regardless of which one of the available Qucs-S simulation engines is chosen. Full control is assumed by the Qucs-S package. The results in Figure 8 show both I_{ds} and transconductance g_m plotted against Vgs. Notice that this data agrees with the model parameters IPK0 = 0.05A at VPKS=-0.2V. In this example the simulation data were obtained using the same circuit test bench as the one drawn in Figure 7 but with the simulation controlled by the Nutmeg script CUSTOM1. The use of this type of Nutmeg script gives users much more control of a simulation sequence and the extraction of parameters from the resulting output data. Similarly, Figure 8 introduces a very low level script for controlling a DC simulation using the test bench drawn in Figure 7. Full control of the simulation process is given to a user. However, as a consequence users become responsible for setting up the simulation parameters and the data extraction process.

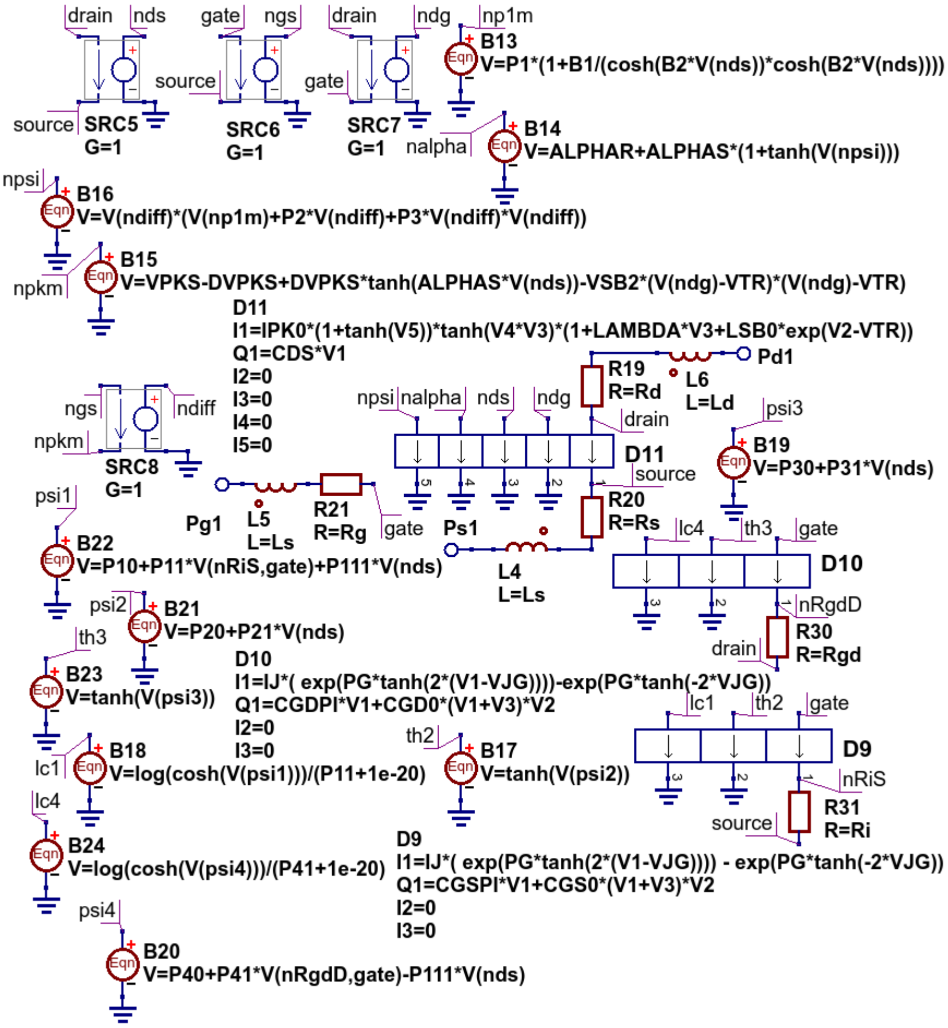

Figure 5. The internal circuit of a Qucs-S compact device model subcircuit for a simplfied Angelov GaN HEMT: Output quantities set by the model compact equations are represented by EDD and SPICE B component charges, currents and voltages respectively.

5. Extending Qucs-S/Ngspice/SPICE OPUS simulation capabilities with hardware description language scripts

The GaN HEMT model and examples give a good idea of the fundamental simulation and data processing features that result when combining schematics and HDL scripts as input to a modern circuit simulator. Although text scripts are normally employed as a SPICE entry medium,there appears to be little published reference to their use with schematics. Moreover, this is very surprising because the inclusion of blocks of script language statements embedded within a circuit schematic allows users to easily control complex simulation sequences and to construct post-simulation data processing routines for tasks not covered by the basic SPICE built-in commands. One such example is the well known omission of specific routines for S parameter RF circuit simulation in SPICE 3f5. Qucs-S however, does implement two-port scattering parameter simulation (S) using the Ngspice, SPICE OPUS and Xyce SPICE engines

```
.SUBCKT GaN_HEMT _net3 _net0 _net5 IPK0=0.05 VPKS=-0.2 P1=0.8 P2=0.0 P3=0.0
+ ALPHAR=0.1 ALPHAS=1.0 B1=0.1 B2=4.0 VSB2=0.0 LAMBDA=0.01 DVPKS=0.2 VTR=50.0
+ Rg=0.05 Rs=0.05 Rd=0.05 LSB0=0.0 Ld=1e-10 Lg=1e-10 Ls=1e-10 P11=0.25 P10=0.48
+ P20=0.03 P21=0.21 P30=0.03 P31=0.21 P40=0.48 P41=0.25 P111=0.008 CGDPI=200e-15
+ CGSPI=15e-15 CGS0=3500e-15 CGD0=15e-15 CDS=800e-15 IJ=5e-4 PG=15 VJG=0.7 Ri=1 Rgd=0.01
L5 _net0  _net1 Ls
GSRC4 0 ndiff ngs npkm 1
R15 0  ndiff 1
GSRC3 0 ndg drain gate 1
R3 0  ndg 1
GSRC2 0 ngs gate source 1
R2 0  ngs 1
GSRC1 0 nds drain source 1
R1 0  nds 1
B16 npsi  0  V = V(ndiff)*(V(np1m)+P2*V(ndiff)+P3*V(ndiff)*V(ndiff))
B15 npkm  0  V = VPKS-DVPKS+DVPKS*tanh(ALPHAS*V(nds))
+                         -VSB2*(V(ndg)-VTR)*(V(ndg)-VTR)
B14 nalpha  0  V = ALPHAR+ALPHAS*(1+tanh(V(npsi)))
B13 np1m  0  V = P1*(1+B1/(cosh(B2*V(nds))*cosh(B2*V(nds))))
R21 gate  _net1 Rg
B22 psi1  0  V = P10+P11*V(nRiS,gate)+P111*V(nds)
B21 psi2  0  V = P20+P21*V(nds)
B19 psi3  0  V = P30+P31*V(nds)
B20 psi4  0  V = P40+P41*V(nRgdD,gate)-P111*V(nds)
B18 lc1  0  V = log(cosh(V(psi1)))/(P11+1e-20)
B17 th2  0  V = tanh(V(psi2))
R31 source  nRiS Ri
BD9I0 gate nRiS I=IJ*(exp(PG*tanh(2*(V(gate,nRiS)-VJG))))-exp(PG*tanh(-2*VJG))
GD9Q0 gate nRiS nD9Q0 nRiS 1.0
LD9Q0 nD9Q0 nRiS 1.0
BD9Q0 nD9Q0 nRiS I=-(CGSPI*V(gate,nRiS)+CGS0*(V(gate,nRiS)+V(lc1))*V(th2))
BD9I1 th2 0 I=0
BD9I2 lc1 0 I=0
B24 lc4  0  V = log(cosh(V(psi4)))/(P41+1e-20)
B23 th3  0  V = tanh(V(psi3))
R20 _net2  source Rs
BD11I0 drain source I=IPK0*(1+tanh(V(npsi)))*tanh(V(nalpha)*
+             V(nds))*(1+LAMBDA*V(nds)+LSB0*exp(V(ndg)-VTR))
GD11Q0 drain source nD11Q0 source 1.0
LD11Q0 nD11Q0 source 1.0
BD11Q0 nD11Q0 source I=-(CDS*V(drain,source))
BD11I1 ndg 0 I=0
BD11I2 nds 0 I=0
BD11I3 nalpha 0 I=0
BD11I4 npsi 0 I=0
L4 _net2  _net3 Ls
R19 drain  _net4 Rd
L6 _net4  _net5 Ld
R30 drain  nRgdD Rgd
BD10I0 gate nRgdD I=IJ*(exp(PG*tanh(2*(V(gate,nRgdD)-VJG))))-exp(PG*tanh(-2*VJG))
GD10Q0 gate nRgdD nD10Q0 nRgdD 1.0
LD10Q0 nD10Q0 nRgdD 1.0
BD10Q0 nD10Q0 nRgdD I=-(CGDPI*V(gate,nRgdD)+CGD0*(V(gate,nRgdD)+V(lc4))*V(th3))
BD10I1 th3 0 I=0
BD10I2 lc4 0 I=0
.ENDS
```

Figure 6. Qucs-S Ngspice netlist synthesised from the information drawn on the GaN HEMT subcircuit schematic given in Figure 5.

plus the conversion of the simulated S parameter data to other forms of two port representation, like admittance parameters (Y) or impedance parameters (Z) (Bowick, Blyler & Ajluni , 2008). Figure 10 illustrates how Nutmeg scripts can be used to obtain

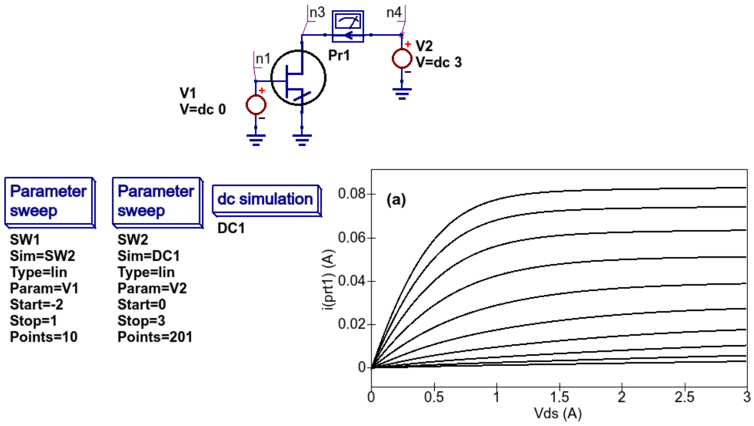

Figure 7. Ngspice,SPICE OPUS or Xyce GaN HEMT DC output characteristics test bench and typical simulation generated Ids/Vds plotted results.

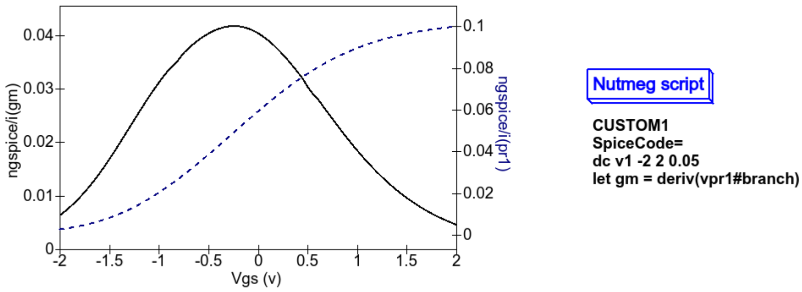

Figure 8. Ngspice and SPICE OPUS GaN HEMT Ids (A) and gm (A/V) simulated data obtained with the test bench in Figure 7: $Vds = 3V$, and a high level Nutmeg script.

two port S parameters with Qucs-S/Ngspice and introduces scripts for S to Y and S to Z parameter conversion. In Figure 10 a GaN HEMT is shown connected as a single stage RF amplifier with a narrow band 50 Ω input LC matching network to give maximum power transfer at a signal frequency of roughly 60MHz.

6. Extending Qucs-S/Xyce simulation and data post-processing capabilities with .PRINT scripts

The Xyce circuit simulator is a simulation engine that accepts SPICE netlist scripts as input and outputs tabular data. The format of the output data may be selected from a list of common formats that are easily read and post-processed by external software. The Xyce software does not include a Nutmeg style post-simulation data manipulation or extraction tool. However, to simplify data processing the Xyce .PRINT has been extended to allow algebraic output equations to be included. Figure 11 illustrates how the Xyce script icon can be added to a Qucs-S schematic, providing direct user control of a simulation and subsequent data extraction or visualization by Qucs-S. Notice that in Figure 11 a separate Xyce script icon is required for each simulation requested. The final test circuit is shown in Figure 12. This shows how the combination of Qucs schematics and the recently implemented Xyce multi-tone Harmonic Balance simulation capability allows RF non-linear steady state spectral analysis to be set-up

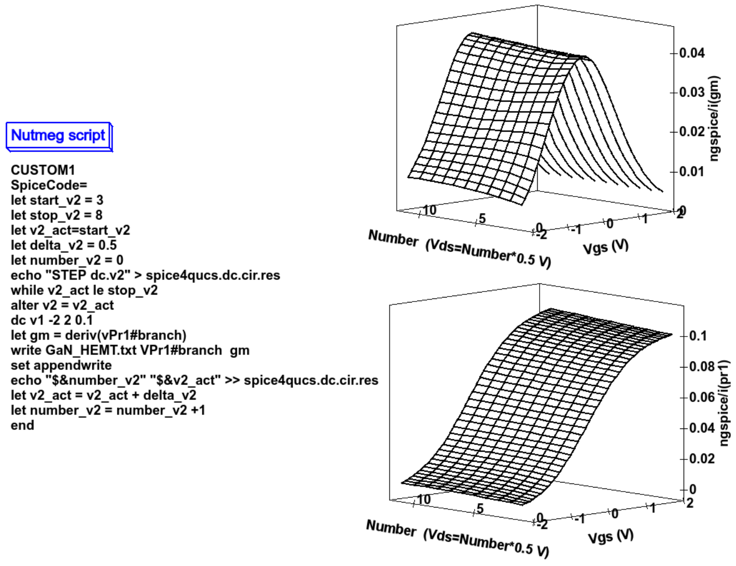

Figure 9. Ngspice and SPICE OPUS GaN HEMT Ids (A) and gm (A/V) simulated data obtained with the test bench in Figure 6: $Vds = 3V$, and a low level Nutmeg script.

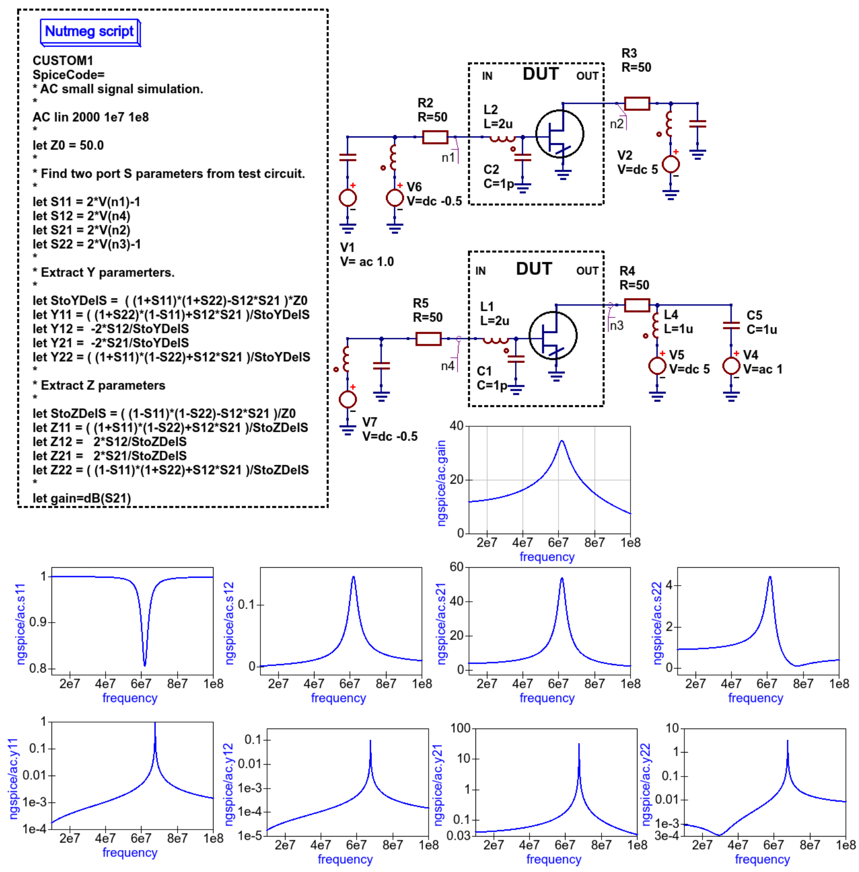

Figure 10. A Qucs-S Nutmeg script for Ngspice S parameter simulation and extraction of Y and Z two port data: (1) IN to OUT signal flow (parameters S11 and S21) and (2) OUT to IN signal flow (parameters S22 and S12); GaN HEMT parameters the same as Table 1.

using Xyce scripts. Prior to the development of the Qucs-S version of Qucs this feature was not available.

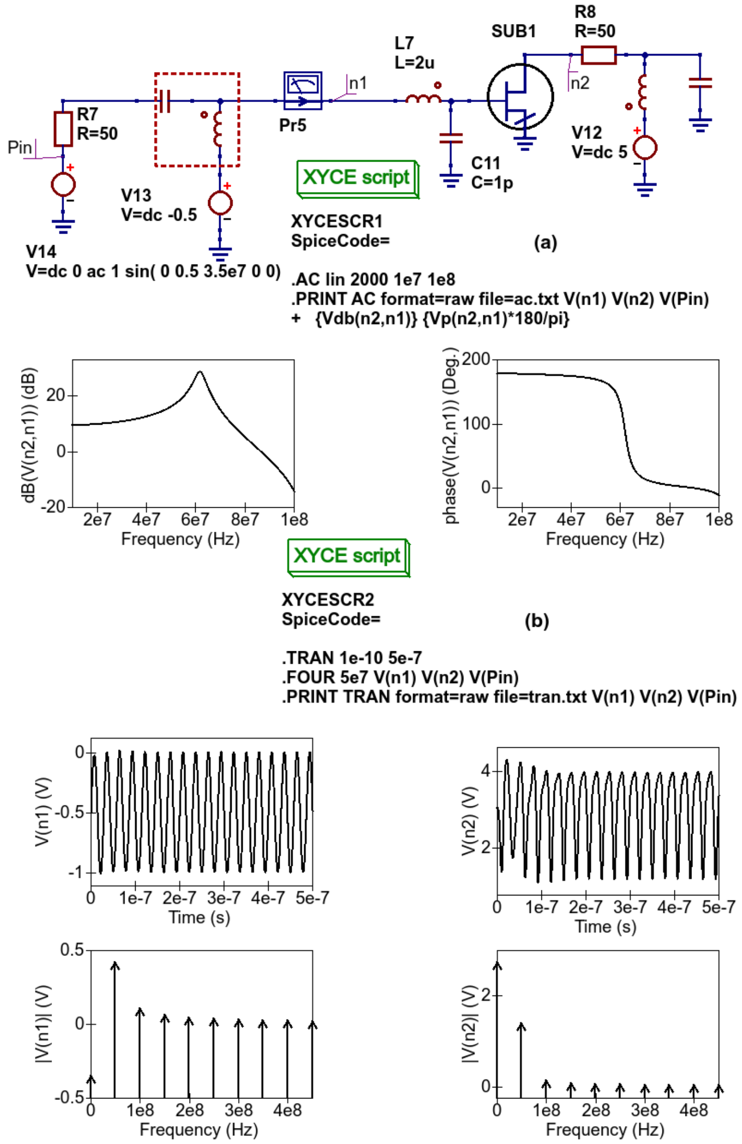

Figure 11. Two Qucs-S Xyce scripts for controlling the AC and transient simulation of a single stage GaN HEMT amplifier: script (a) AC simulation and extraction of amplifier gain db(V(n2,n1)),and phase shift phase(V(n2,n1)); (b) Transient simulation input (node n1) and output voltage Fourier data for V(Pin)=0.5V peak and the sinusoidal input frequency = 50 MHz.

7. Conclusion

With the growing interest in new semiconductor technologies there is a corresponding demand for device modelling and simulation software that allows free access to GPL CAD tools which aid the construction and testing of high performance compact device models. The extended approach to behavioural device modelling introduced in this paper represents one more step along the path to improving the next generation

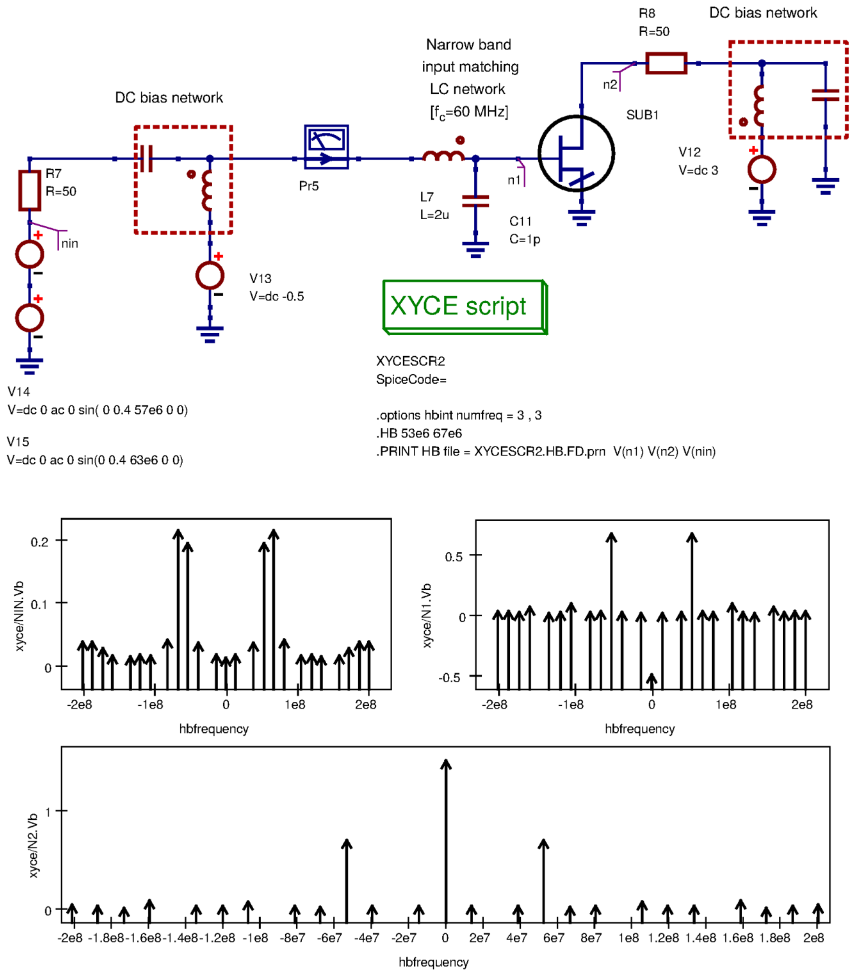

Figure 12. Qucs-S Xyce Two-tone Harmonic Balance test circuit: test bench, Xyce script and typical node voltage spectra.

of freely available modelling and circuit simulation tools. Such improvements are critical in the future development of emerging technology device models and new circuit design methods. This paper also demonstrates how the combination of schematics and HDL scripts, acting as input to a multi-engine circuit simulator, allows access to a range of modelling and simulation tools not previously available in one GPL circuit analysis and design package. To illustrate the power and utility of the new Qucs-S extensions a fundamental compact semiconductor device model for a GaN HEMT is introduced in the text and its performance confirmed by a number of example GaN circuit simulations.

References

Angelov et al. (1992). A new empirical non-linear model for HEMT and MESFET devices. *Microwave Theory and Techniques, IEEE Transactions on*, *40*, 2258–2266.

Angelov et al. (1996). Extensions of the Chalmers non-linear HEMT and MESFET model. *Microwave Theory and Techniques, IEEE Transactions on*, *44*, 1664–1674.

Bowick et al. (2008), *RF circuit design, 2nd Edition*, ISBN:978-0-7506-8518-4, Elsevier, London and Amsterdam. Retrieved from http://store.elsevier.com/RF-Circuit-Design/Christopher-Bowick/isbn-9780750685184/.

Brinson M.E. & Jahn S. (2009), Qucs: A GPL software package for simulation, compact device modelling and circuit macromodelling from DC to RF and beyond, *International Journal of Numerical Modelling: Electrical Networks, Devices and Fields*, John Wiley & Sons, Ltd, DOI:10.1002/jnm.702, *22*, 297–319.

Brinson M.E. & Kuznetsov V. (2016). Qucs-S: Spice4qucs-help documentation; user manual and reference material. *Qucs project team*. Retrieved from https://qucs-help.readthedocs.org/en/spice4qucs.

Eaton et al. (2016), *GNU Octave Version 4.20.* Octave project team. Retrieved from https://www.gnu.org/software/octave/.

Kuznesov V. (2016), *Qucs-S: Unofficial build with spice4qucs features enabled; release candidate 8*, Qucs project team. Retrieved from https://github.com/ra3xdh/qucs/release/tag/0.0.19S-rc8.

Jahn S. & Brinson M.E. (2008), Interactive compact device modelling using Qucs equation-defined devices, *International Journal of Numerical Modelling: Electrical Networks, Devices and Fields*, John Wiley & Sons, Ltd, DOI:10.1002/jnm.676, *21*, 333–349.

Johnson B., Quarles T., Newton A.R, Pederson D.O & Sangiovanni-Vincentelli A. (1992), *SPICE3 Version 3f User's Manual*. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California.

Ngspice (2016), *Ngspice: mixed-level/mixed-signal circuit simulator based on Berkeley's SPICE 3f5*. Ngspice project team. Retrieved from http://ngspice.sourceforge.net.

SPICE OPUS (2016). *SPICE OPUS: analog circuit simulator engine specially suited for optimization tools, based on SPICE 3f5 and XSPICE*. Faculty of Electrical Engineering at the University of Ljubljana, Slovenia. Retrieved from http://fides.fe.uni-lj.si/spice/download/.

Xyce (2016), *Xyce Parallel electronic simulator, Version 6.6*. Sandia National Laboratories, USA.Retrieved from https://xyce.sandia.gov/.