Qucs-0.0.19S: a new open-source circuit simulator and its application for hardware design

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

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

***Abstract*—Circuit simulation is widely used in communication and control equipment hardware design tasks. This article introduces an extended version of the popular Qucs circuit simulator called Qucs-0.0.19S. It is a simulation tool which supports multiple SPICE circuit simulators, including Ngspice and Xyce. The package includes a graphical user interface, component and compact device modelling tools, a choice of simulation engine, and advanced simulation data postprocessing facilities. It allows user to construct new component using XSPICE extension and construct new simulations using Nutmeg scripting. Qucs-0.0.19S is targeted at academic and industrial applications. Software implementation details and application cases are considered.**

***Index Terms*—Qucs, SPICE, Ngspice, Xyce, Nutmeg scripting, circuit simulation, EDA**

I. Introduction

Open source software offers access and cost benefits to enterprise information technology. However, not all sectors have a fully developed software base. One example is electronic design automation (EDA) where General Public Licence (GPL) circuit simulation and printed circuit board layout packages are undergoing rapid development. The ”Quite universal circuit simulator” (Qucs) [1], [2] is one of a new breed of GPL circuit simulators. Qucs was started by M. Margraf and S. Jahn in 2001. The initial intention was that Qucs should be an RF circuit analysis package which offered features not found in SPICE. Recently a new team took over responsible for Qucs development.

Qucs-0.0.19S is a freely available package with versions for Linux, Windows © and MacOS © . It includes a simulation kernel called Qucsator. Although Qucsator has acceptable performance it is not fully compatible with SPICE 2g6 or 3f5 [3], [4]. Qucs has a unique netlist syntax and model format with SPICE support implemented via a software compatibility layer. It does not allow direct access to manufacturers SPICE models and libraries. The compatibility layer also prohibits access to a number of SPICE built in models, simulation types and the Nutmeg scripting language. A ”Spice4qucs” subsystem has been added to Qucs to form Qucs-0.0.19S [5], and hence overcome these limitations. Qucs-0.0.19S was presented during MOS-AK workshop at Graz, Austria [6].

Spice4qucs is not another SPICE simulation kernel but acts as an interface to a number of established GPL SPICE engines. These have excellent performance, but usually lack a graphical user interface (GUI) for schematic capture and external simulator launch control. The reverse is true for Qucs which is distributed with mature GUI and modelling tools.

Evaluation of GPL SPICE simulators, plus feedback from Qucs users, suggested; (a) Qucs should support several SPICE GPL kernels, (b) Qucs should not simple be a schematic capture and simulation software package but must also offer advanced data processing features, and (c) provide a range of compact device modelling facilities. Factor (a) is met by the Ngspice [7] and XYCE [8] SPICE simulators. Moreover, Spice4qucs is able to launch both simulators from the Qucs GUI. Qucsator has excellent small signal AC and S-parameter simulation performance. But Qucsator time-domain simulation is not that stable. In particular, Qucsator cannot reliably simulate switching circuits. The addition of SPICE based simulation to Qucs allows this limitation to be largely eliminated, making Qucs-0.0.19S, a viable choice for research and industrial circuit design [9], [10].

II. An overview of Qucs-0.0.19S component models

The Spice4qucs subsystem is designed for the simulation of Qucs circuit schematics with Ngspice or Xyce launched as external simulation engines [11]. In general legacy Qucs circuit doesn’t require tweaking to simulate it with Qucs-0.0.19S. Qucs legacy passive components can be simulated with Qucs-0.0.19S. In addition Qucs-0.0.19S introduces a group of passive component models with SPICE format. Qucs legacy semiconductor device models are SPICE incompatible. Similar to passive components active device models have a fixed list of named parameters [1], [12]. Moreover, some of these are SPICE incompatible. Qucs-0.0.19S allows users to construct SPICE device definitions from a name, a model specifier and a SPICE style ”modelcard”. These can be attached to a schematic symbol and passed directly to a SPICE kernel.

Qucs-0.0.19S subcircuit and library components form part of a file component subclass. These allow the construction of more complex components from pre-defined model primitives and manufactures models. Qucs-0.0.19S allows users access to the following types of file component:

1) *Subcircuits*, for the construction of new components from predefined components. This form of subcircuit is identical to the original Qucs implementation [12], except that each subcircuit is stored as a .SUBCKT netlist;

2) *SPICE file* components, for attaching SPICE .SUBCKTs to a circuit schematic. This component allows to pass unmodified SPICE netlist directly to simulator. Netlist is stored in a separate file;
3) *Library components*, for the storage and recall of previously defined component and device models. Qucs/Qucs-0.0.19S libraries are encoded in text XML format. Library can store unmodified SPICE code.

III. The operation principles of multi-simulator support in Qucs-0.0.19S

Algorithm 1 outlines the Qucs netlist building method. Qucsator does not use netlist sections [12]. A Qucs schematic is represented as a C++ class, consisting of a set of netlist processing methods. A single method scans a schematic file in one pass and outputs information describing located components.

Algorithm 1:

```
Data: Qucs Schematic
Data: Qucs netlist filename
Result: Qucs netlist
begin
    foreach (Component in Schematic) do
        Netlist ← Component.getQucsNetlist()
    end
end
```

In contrast, SPICE netlists consist of separate sections for equations, post-processor directives, and component specifications. Hence, building a SPICE netlist requires a multiple pass method, see Algorithm 2.

Algorithm 2:

```
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
    // begin of .control section
    foreach (Component in Schematic) do
        if (Component is Simulation) then
            Netlist ← Component.getBeforeSimScript()
            Netlist ← Component.getSpiceNetlist()
            Netlist ← Component.getAfterSimScript()
            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
```

A Qucs schematic consists of a group of components where every item has a properties list. For example, let's consider an RC-network schematic (see Figure 1). Qucs simulation icons and equations are considered to be a special forms of component. The Qucs netlist has declarative format. During scanning Qucsator automatically separates components, equations, and simulator directives. The order has no effect on the final result.

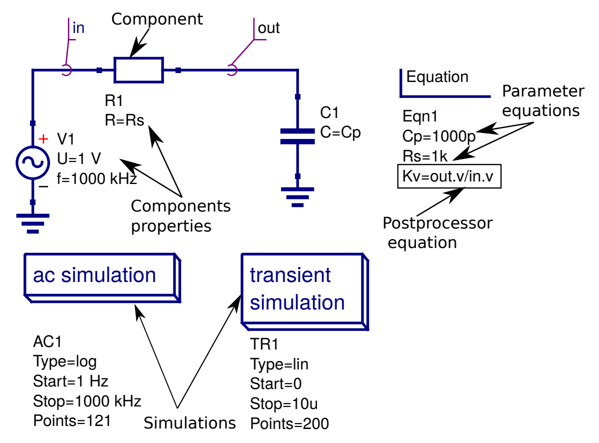


Fig. 1. A Qucs RC circuit schematic with netlist sections labelled

The Qucs netlist for the RC network is:

```
# Qucs 0.0.19  RC1.sch
Vac:V1 in gnd U="1 V" f="1000 kHz"
R:R1 in out R="Rs"
C:C1 gnd out C="Cp"
Eqn:Eqn1 Cp="1000p" Rs="1k"
Kv="out.v/in.v" Export="yes"
.AC:AC1 Type="log" Start="1 Hz"
Stop="1000 kHz" Points="121" Noise="no"
.TR:TR1 Type="lin" Start="0"
Stop="10u" Points="200"
```

The Ngspice netlist for the RC network is:

```
* Qucs 0.0.19  RC1.sch
* Parameters section
.PARAM Cp={1000p}
.PARAM Rs={1k}
* Components section
V1 in 0 DC 0 SIN(0 1 1000K 0 0) AC 1
R1 in out  {RS}
C1 0 out  {CP}
* Simulations execution section
.control
AC DEC 21 1 1000K
let Kv=V(out)/V(in)
* Write result to text file
write RC1_ac.txt v(in) v(out)  Kv
TRAN 5e-08 1e-05 0
* Write result to text file
write RC1_tran.txt v(in) v(out)
exit
.endc
* Netlist ends here
.END
```

Qucs output data are translated into an XML dataset when simulation finishes.The Ngspice netlist format is very close to an imperative programming language, with `.PARAM` directives in proper order for error free evaluation. At the end of a Ngspice netlist is a `.control ... .endc` group. This group contains a Ngnutmeg post-processor script that is executed after a netlist is scanned by Ngspice. During scanning, simulation and post-processor directives are placed between the control words `.control ... .endc`. The `.control ... .endc` group also supports Ngnutmeg file `write` directives for storing simulation datasets. Ngspice datasets are written in the SPICE-3f5 raw-ASCII format which in turn are converted and saved by Qucs-0.0.19S as part of a Qucs XML dataset.

With Xyce multiple simulations are not supported. The Xyce netlist has the following format:

```
 * Qucs 0.0.19  RC1.sch
.PARAM Cp={1000p}
.PARAM Rs={1k}
V1 in 0 DC 0 SIN(0 1 1000K 0 0) AC 1
R1 in out  {RS}
C1 0 out  {CP}
.TRAN 5e-08 1e-05 0
.PRINT  tran format=raw file=RC1_tran.txt v(in) v(out)
.END
```

Spice4qucs operates at GUI level in distinct steps; netlist building followed by simulation and finally it uses a raw-ASCII output data parser to generate a Qucs XML dataset. All schematic symbols have an XML representation which is written to memory during schematic file loading.

As the Xyce simulator does not include a data post-processor the netlist building algorithm for Xyce is much simpler, see Algorithm 3.

The block diagram drawn in Figure 2 illustrates the interaction between schematic capture, simulation and data visualization for all used simulation backends.

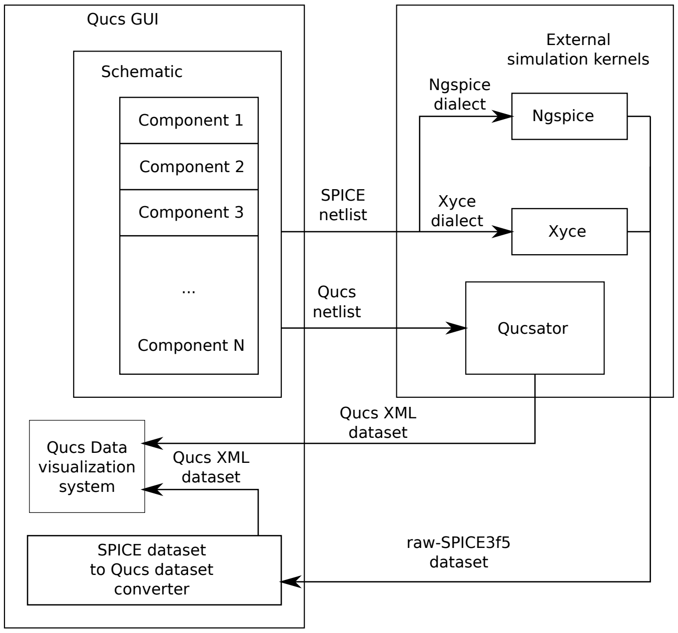


Fig. 2. Spice4qucs subsystem dataflow block diagram

A number of the SPICE simulation types generate Qucs incompatible output datasets, implying that they require unique custom parsers. The parsers implemented in the current version of Qucs-0.0.19S are for SPICE-3f5 raw-ASCII (AC, DC, TRAN, and Parameter sweep simulation), Fourier simulation, noise simulation and HB simulation (XYCE only). The Spice4qucs subsystem extracts output data from each simulation request and combines them into single Qucs XML dataset ready for processing by the Qucs data visualization system.

Algorithm 3:

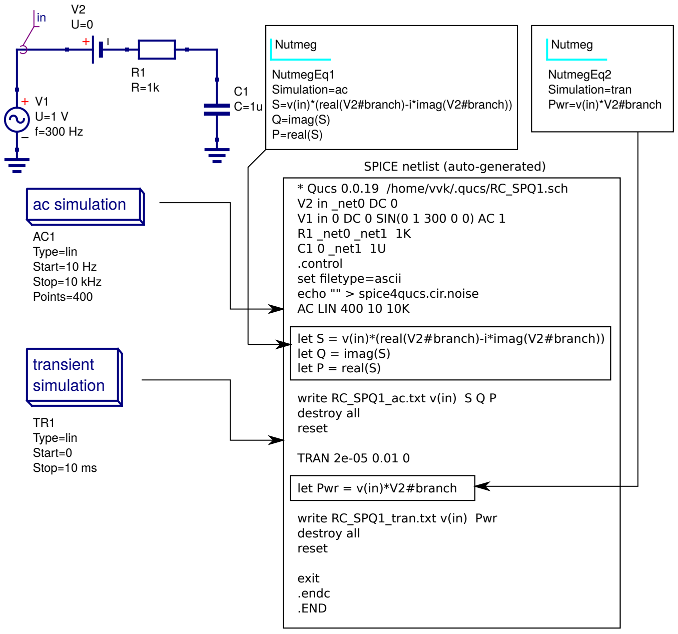
```
Data: Qucs Schematic
Data: SPICE netlist filename
Result: SPICE netlist
begin
    foreach (Component in Schematic) do
        if (Component is Parameter or directive) then
            Netlist ← Component.getSpiceExpression()
        end
    end
    foreach (Component in Schematic) do
        if (Component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

IV. Qucs-0.0.19S simulations

A. *Common simulations and simulation data postprocessing*

The following simulation types are implemented .DC, .AC, .TRAN, .FOUR, .DISTO, .NOISE, and a new ”Ngspice Custom” form. XYCE backend supports single-tone and multitone Harmonic Balance simulation. Qucs allows to get access to these simulations from the GUI.

The Qucs data post-processor has many SPICE incompatible functions. A way to overcome this is to pass post-processor directives directly to Nutmeg via a new component called ”Nutmeg equation”. Illustrated in Figure 3 is an RC network driven by an AC source. This demonstrates how .AC and .TRAN are defined and how ”Nutmeg” can be used to determine, apparent, active and reactive power, given by $S = |U \cdot \bar{I}|, P = \Re[U \cdot \bar{I}], Q = \Im[U \cdot \bar{I}]$., respectively. Similarly, real power can be calculated from transient data, using $P(t) = u(t) \cdot i(t)$.



```
NutmegEq1
Simulation=ac
S=v(in)*(real(V2#branch)-i*imag(V2#branch))
Q=imag(S)
P=real(S)
```

```
NutmegEq2
Simulation=tran
Pwr=v(in)*V2#branch
```

```
* Qucs 0.0.19  /home/vvk/.qucs/RC_SPQ1.sch
V2 in _net0 DC 0
V1 in 0 DC 0 SIN(0 1 300 0 0) AC 1
R1 _net0 _net1  1K
C1 0 _net1  1U
.control
set filetype=ascii
echo "" > spice4qucs.cir.noise
AC LIN 400 10 10K

let S = v(in)*(real(V2#branch)-i*imag(V2#branch))
let Q = imag(S)
let P = real(S)

write RC_SPQ1_ac.txt v(in)  S Q P
destroy all
reset

TRAN 2e-05 0.01 0

let Pwr = v(in)*V2#branch

write RC_SPQ1_tran.txt v(in)  Pwr
destroy all
reset

exit
.endc
.END
```

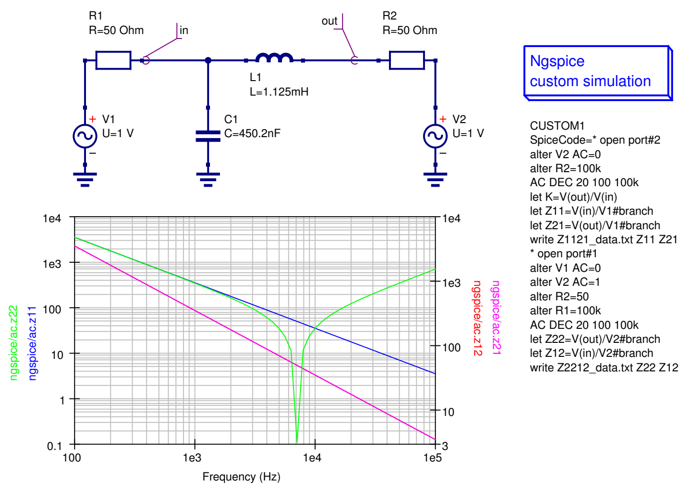
Fig. 3. An example of Nutmeg post-processor equation usage

B. *Ngnutmeg scripting*

Qucs-0.0.19S has a powerful new feature, called ”Ngspice custom simulation”, where a Nutmeg script is added to a Qucs schematic, allowing SPICE statements and Ngnutmeg scripts to be passed directly to a SPICE netlist.

It allows to get easy access to all Ngnutmeg functions from the GUI. It's able to construct nonstandard simulations using Ngnutmeg scripting (for example scattering matrix and SWR analysis, Monte-Carlo analysis).

For example, Z-parameter analysis is not available for the most of SPICE-compatible simulators including proprietary ones. But it could be easily constructed with Qucs-S, Ngspice, and Nutmeg scripting. Figure 4 illustrates this approach for a passive low-pass Butterworth LC-filter.



```
CUSTOM1
SpiceCode=* open port#2
alter V2 AC=0
alter R2=100k
AC DEC 20 100 100k
let K=V(out)/V(in)
let Z11=V(in)/V1#branch
let Z21=V(out)/V1#branch
write Z1121_data.txt Z11 Z21
* open port#1
alter V1 AC=0
alter V2 AC=1
alter R2=50
alter R1=100k
AC DEC 20 100 100k
let Z22=V(out)/V2#branch
let Z12=V(in)/V2#branch
write Z2212_data.txt Z22 Z12
```

Fig. 4. Z-parameter extraction with Nutmeg scripting

Postprocessor directives are used to extract voltage and current data form AC-simulation results and convert it into desired Z-parameter value.

V. XSPICE support in Qucs-S

XSPICE is SPICE-3f5 extension targeted on system-level circuit design tasks. It is especially important for communication equipment. XSPICE introduces a set of additional analog and mixed-signal models targeted on system-level design. Qucs-S with Ngspice backend supports a wide range of XSPICE blocks.

The following XSPICE analog devices are presented in Qucs-S out-of-box: gain block, integrator, differentiators, adder, multiplier.

These blocks allows simulate not only analog circuits, but also to solve control theory tasks. For example, PI-controller step response analysis is shown in the Figure 5.

This simulation uses XSPICE blocks (analog gain, integrator, and adder) to define PI-controller elements and transient simulation to obtain step response.

It's able to construct a new XSPICE block using "XSPICE generic device" component (Figure 7). It's sufficient to provide port list and modelcard reference to create new device. It's able to attach user symbol to a new device using standard Qucs subcircuit technique [12].

XSPICE allows to develop new devices using CodeModel technique [13]. User can compile a set of CodeModels in a single dynamic-loadable binary library. Now it's available inclusion of precompiled CodeModel libraries using special

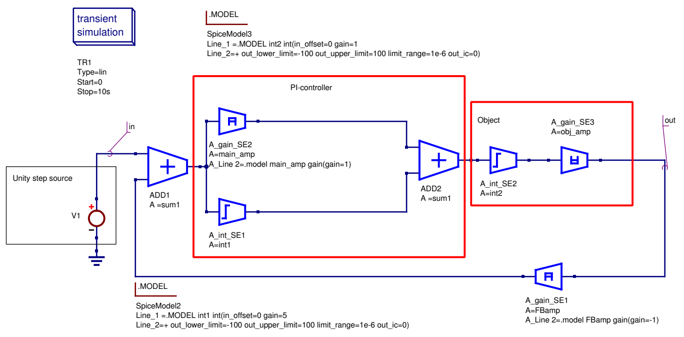

transient
simulation
TR1
Type=lin
Start=0
Stop=10s
.MODEL
SpiceModel3
Line_1 =.MODEL int2 int(in_offset=0 gain=1
Line_2=+ out_lower_limit=-100 out_upper_limit=100 limit_range=1e-6 out_ic=0)
PI-controller
in
A_gain_SE2
A=main_amp
A_Line 2=.model main_amp gain(gain=1)
Unity step source
V1
ADD1
A =sum1
A_int_SE1
A=int1
ADD2
A =sum1
Object
A_gain_SE3
A=obj_amp
A_int_SE2
A=int2
out
.MODEL
SpiceModel2
Line_1 =.MODEL int1 int(in_offset=0 gain=5
Line_2=+ out_lower_limit=-100 out_upper_limit=100 limit_range=1e-6 out_ic=0)
A_gain_SE1
A=FBamp
A_Line 2=.model FBamp gain(gain=-1)

Fig. 5. PI-controller analysis with XSPICE analog blocks

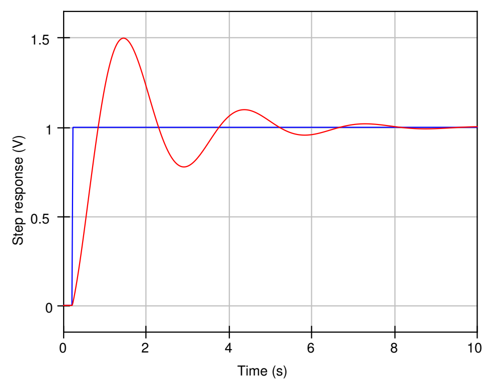


Fig. 6. Simulated step response of PI-controller

circuit symbol (Figure 7). It's sufficient to specify location of binary library file. New models form this library could be used using user-defined XSPICE block and general modelcard.

Qucs-S will allow to attach CodeModels to schematic and compile it automatically during netlist building. This feature is under construction now and it will not be considered further.

VI. Conclusion

Qucs-0.0.19S is the first step in the development of an open-source circuit simulator that combines, and extends, the best features available with GPL circuit simulators. It can simulate a wide range of different size circuits, including those designed using manufacturer's device models.

Qucs-0.0.19S allows switching of simulation backends. Qucs-S covers the following application areas:

1) Realistic analog circuit simulation in time domain with Ngspice backend. Full support of SPICE-3f5 standard allows to use wide range of component models provided by vendors;
2) RF-circuits analysis (S,Z,Y-parameters matrix) using Nutmeg scripting and Harmonic balance analysis with XYCE backend [14]. This application is not available for many other SPICE-compatible simulators;
3) Control theory applications using XSPICE analog blocks;

The main advantages of Qucs-0.0.19S are:

1) It's free and open-source. It allows users to easily modify sources and propose new features;

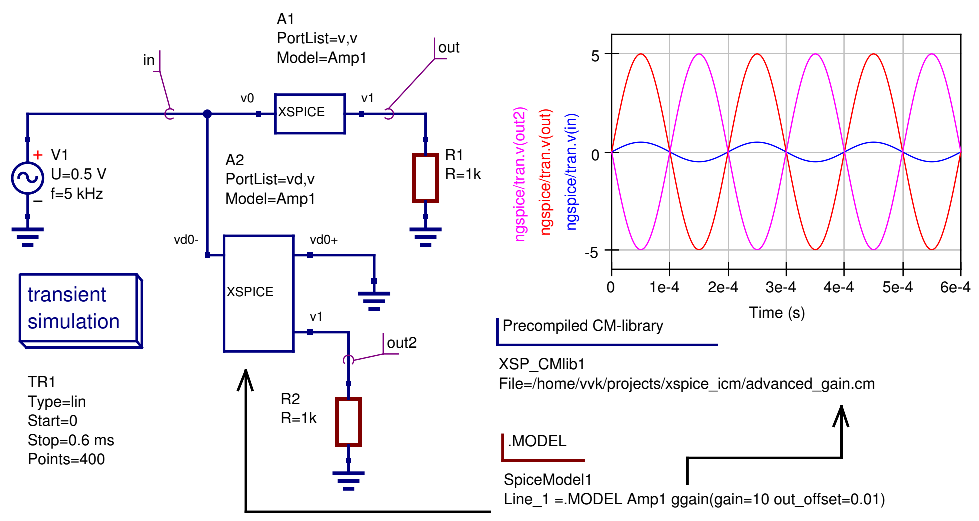


Fig. 7. User-defined XSPICE device construction

2) Switchable simulation backends allows user to select the most suitable one for every simulation task;
3) Advanced postprocessing with Nutmeg Equations;
4) GUI allows to get access to unlimited features of Nutmeg scripting. It allows user to construct new simulation types (for example RF simulation types) without modification of Qucs and simulator backends sources;
5) XSPICE allows system-level design. Also CodeModel technique allows to construct new XSPICE devices without modification of simulator sources.

Considering all above, we can conclude that Qucs-0.0.19S is not simple GUI for SPICE backends. It allows also advanced features in simulation result postprocessing, circuit parametrization, and user devices and simulation definition. And Qucs-0.0.19S could be recommended for communication and control equipment equipment hardware design tasks.

References

[1] M. E. Brinson and S. Jahn, "Qucs: A GPL software package for circuit simulation, compact device modelling and circuit macromodelling from DC to RF and beyond," *International Journal of Numerical Modelling (IJNM): Electronic Networks, Devices and Fields*, vol. 22, no. 4, pp. 297 – 319, September 2008. [Online]. Available: http://www3.interscience.wiley.com/journal/121397825/abstract

[2] W. Grabinski, M. Brinson, P. Nenzi, F. Lannutti, N. Makris, A. Antonopoulos, and M. Bucher, "Open-source circuit simulation tools for RF compact semiconductor device modelling," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 27, no. 5-6, pp. 761–779, 2014. [Online]. Available: http://dx.doi.org/10.1002/jnm.1973

[3] A. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE Version 2g User's Guide*. Department of Electrical Engineering and Computer Sciences, University of California, 1981.

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

[5] V. Kuznetsov. Unofficial build with spice4qucs features enabled. Release candidate 3. Qucs project team. [Online]. Available: https://github.com/ra3xdh/qucs/releases/tag/0.0.19S-rc3

[6] M. Brinson, R. Crozier, V. Kuznetsov, C. Novak, B. Roucaries, F. Schreuder, and G. B. Torri. Qucs: An introduction to the new simulation and compact device modelling features implemented in release 0.0.19/0.0.19Src2 of the popular GPL circuit simulator. MOS-AK Workshop, Graz. [Online]. Available: http://www.mos-ak.org/graz_2015/presentations/T_5_Brinson_MOS-AK_Graz_2015.pdf

[7] Ngspice: mixed-level/mixed-signal circuit simulator based on Berkeley's Spice3f5. Ngspice project team. [accessed August 2015]. [Online]. Available: https://www.ngspice.org/

[8] Xyce Parallel electronic simulator: version 6.2. Sandia National Laboratories. [accessed August 2015]. [Online]. Available: https://xyce.sandia.gov/

[9] A. Zonca, B. Roucaries, B. Williams, I. Rubin, O. D'Arcangelo, P. Meinhold, P. Lubin, C. Franceschet, S. Jahn, A. Mennella, and M. Bersanelli, "Modeling the frequency response of microwave radiometers with QUCS," *Journal Of Instrumentation*, no. 5(12):T12001, November 2010.

[10] V. Kuznetsov and L. Kechiev, "Charged Board Model ESD Simulation for PCB Mounted MOS Transistors," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 57, no. 5, pp. 947–954, 2015.

[11] M. Brinson and V. Kuznetsov. Spice4qucs-help documentation. User Manual and Reference Material. [Online]. Available: https://qucs-help.readthedocs.org/en/spice4qucs

[12] S. Jahn, M. Margraf, V. Habchi, and R. Jacob, *Qucs. Technical papers.* [Online]. Available: http://qucs.sourceforge.net/docs/technical/technical.pdf

[13] F. Cox III, W. Kuhn, J. Murray, 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 vol.2.

[14] M. Brinson and V. Kuznetsov, "Qucs equation-defined and Verilog-A RF device models for harmonic balance circuit simulation," in *Mixed Design of Integrated Circuits Systems (MIXDES), 2015 22nd International Conference*, June 2015, pp. 192–197.