

DBCS-0.0.29V: An open-source circuit simulator

Y. Naveen Kumar¹ and K. Vijaya Lakshmi²

¹*Assistant Professor, GATES Institute of Technology, Gooty*

²*Senior Research Fellow, Sri Krishnadevaraya University, Anantapuramu*

E-mail: mail2naveeny@gmail.com , mail2vijjiy@gmail.com

Abstract—Circuit Simulation is widely used for the design of circuits. Modelling of a device is a key aspect of circuit simulation, since it is the link between the physical device and the simulated device. Currently available circuit simulation programs provide a variety of built in models. This article introduces a new version circuit simulator called DBCS-0.029V. It is a simulation tool which supports multiple circuit simulators. The Simulator is equipped with GUI (Graphical User Interface) components, models of common circuit elements, and active as well as passive elements and built in packages. This simulator allows user to perform various analyses of electrical and electronic circuits. It is a versatile program and can be used both in academic and industries. This paper presents a Java package implementation details for finding the response of an element in a circuit.

Index Terms— DBCS, Spice, GUI, 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 "Digi Brains circuit simulator" (DBCS) [1], [2] is one of a new breed of circuit simulator. DBCS was started by Y. Naveen Kumar and K. Vijaya Lakshmi in 2016. The initial intention was that DBCS should be primarily a circuit analysis package which offered features not found in SPICE. Recently a new team took over responsible for DBCS development. DBCS-0.0.29V is a freely available package with versions for Windows XP, 7, 8 and 10©. In order to run this simulator the user has to install Java on his computer. It includes Java Virtual Machine. JVM is an abstract computing machine that enables a computer to run a Java program. Although JVM has acceptable performance it is not fully compatible with SPICE 2g6 or 3f5 [3], [4]. DBCS 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 and simulation types. These have excellent performance, but usually lack a graphical user interface (GUI) for schematic capture and external simulator launch control.

II. AN OVERVIEW OF DBCS-0.0.29V COMPONENT MODELS

Let's consider available analysis types in DBCS. They are: DC, AC, TRAN analysis, parameter sweep, frequency domain scattering matrix analysis, and harmonic balance. The following components are available:

- 1) Lumped RLC-components;
- 2) Independent (AC, DC, and pulse) and linear-dependent sources (CCCS, CCVS, VCCS, VCVS);
- 3) Switches and relays;
- 4) Digital components: logic gates, flip-flops, counters, adders, coders, decoders, VHDL and Verilog modules;
- 5) Library components for the storage and recall of previously defined component and device models.

DBCS has powerful data visualization system. The following simulation data representation methods are available:

- 1) 2D and 3D Cartesian plots;
- 2) Polar plots;
- 3) Plot in complex plain;
- 4) Tabular data representation;

The key feature of simulator is transforming of two port network parameters from impedance parameters to admittance, transmission and hybrid parameters and similarly we can find out the remaining parameters just by knowing any one parameter. It is not presented in SPICE-based simulators. We can find the response of an element in the circuit and we can also trace the response at any instant. There is provision for creating our own package called user defined packages. With this provision, the user can create his/her own package and it can be accessed in any program wherever required.

III. DBCS usage for education

Open architecture and user defined packages make DBCS an ideal candidate for academic usage. Some new features were added for an optimization of DBCS usage for educational purposes. They are:

1) Export of schematic graphics to raster or vector graphics files (PNG, JPEG, and PDF). These graphics could be used in LaTeX or popular Office suites. This feature is important for articles, publishing and reports preparation. This makes publishing process more straight forward. 2) DC analysis can be used for circuits with time invariant sources for eg., steady state DC sources.

3) DBCS can open schematic documents from its future versions. This feature is useful for simulation results exchange.

4) Simulation results could be exported to Matlab/Octave [5] numeric analysis software.

A practical training course on computer simulation of electrical circuits was developed at Digi Brains Academy at Anantapuramu. This course includes the following practical works:

- 1) DBCS circuit simulator introduction. Basic simulation types (DC, AC, Transient, Parameter sweep).
- 2) Creating User defined packages.
- 3) Creating and using Java Archive files.
- 4) Transforming of two port network parameters from one parameter to another.

Students learn the bases of writing up a program and analysing the behaviour of the elements of the given circuit. SPICE-based proprietary simulator is unable to perform such simulation types. So DBCS is optimal choice for two port network simulation with academic purposes.

IV. THE OPERATION PRINCIPLES OF MULTI-SIMULATOR SUPPORT IN DBCS-0.0.29V

Algorithm 1 outlines the DBCS netlist building method. A DBCS schematic is represented as a Java 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. 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. A DBCS schematic consists of a group of components where every item has a properties list.

Algorithm 1

Data: DBCS Schematic
Data: DBCS netlist filename
Result: DBCS netlist

```
begin
    foreach (Component in Schematic) do
        | Netlist
Component.getDBCSNetlist()
    end
end
```

Algorithm 2

Data: DBCS 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

For example, consider the RC-network schematic given in Figure 1. DBCS simulation icons and equations are represented as a special form of component. The DBCS netlist has a declarative format. During scanning JVM automatically separates components, equations, and simulator directives. The order has no effect on the final result.

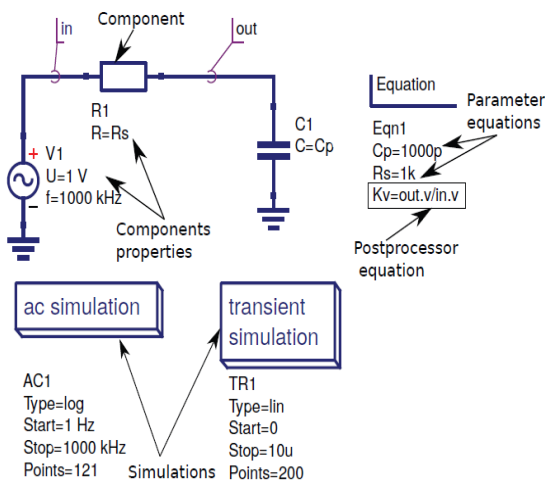


Figure 1: A DBCS RC circuit schematic with netlist sections labelled.

DBCS output data are translated into a byte code dataset when simulation finishes. The Ngspice netlist format is very close to an imperative programming language, with .PARAM directives listed in proper order for error free evaluation. Placed at the end of a Ngspice netlist is a .controlendc group. During scanning, simulation and post-processor directives are placed between the control words .controlendc. Ngspice datasets are written in the SPICE-3f5 raw-ASCII format which in turn are converted and saved by DBCS-0.0.29V as part of a DBCS byte code dataset. The Xyce simulator is different in that it does not support multiple simulations.

End

V. CONCLUSION

DBCS-0.0.29V is the first step in the development of an open-source circuit simulator that combines, and extends, the best features available in the circuit simulators. It can simulate a wide range of different size circuits, including those designed using manufacturer's device models. DBCS-0.0.29V allows switching of simulation engines. The package is open-source and free to individual and industrial users. It allows users to easily modify sources and propose new features with the help of built in packages.

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] J. W. Eaton, D. Bateman, and S. Hauberg, *GNU Octave Manual*. Network Theory Ltd. Bristol, UK, 2008. [Online]. Available: <http://www.networktheory.co.uk/docs/octave3/index.html>