NextGen SPICE – Electrical Circuit Simulation Library for .NET

author: Radek Zikmund | supervisor: Mgr. Pavel Ježek, Ph.D | 2018

Introduction

Electrical circuit simulators were initially developed as a design aid for electrical engineers who designed integrated circuits. Since then, the computational capacity available in personal computers has increased significantly, and it is possible to use circuit simulators as part of other applications such as educational programs for high school students or programs using approaches of evolutionary programming to evolve new circuits. The goal of this thesis is to provide a circuit simulator library for .NET which could be used to create such applications.

Goals

1. Implement Time-domain (transient) simulation of electrical circuits
2. Support following devices: voltage source, current source, resistor, inductor, capacitor, diode, BJT transistor
3. Make the library portable and extensible
4. Implement parser for a subset of standard SPICE netlist format of circuit description
5. Evaluate use of double-double technique for representing real numbers with greater precision inside the simulator

Double-double Precision Technique

In our implementation, we used the C++ library QD by Hida et al., which implements the double-double arithmetic. This technique is able to represent real numbers with at least 2*53 = 106 bit significand, which is only slightly less than the 113 bit significand of the IEEE binary128 format. Operations on the double-double format are implemented using standard 64-bit double operations and therefore do not require special HW support.

Berkeley SPICE Simulators

Our simulator is inspired by the family of SPICE programs which were originally developed at University of California, Berkeley. These programs were very influential in the development of modern circuit simulator programs. The syntax used for describing the circuits for the SPICE2 and SPICE3 programs became an industry standard during the 1980s and is still used today. We have therefore implemented support for importing circuits from standard SPICE netlist files. Figure 1 shows an example of a SPICE netlist.

```
BRIDGE-T CIRCUIT
*
VBIAS 1 0 12
R1 1 2 10
R2 2 0 10
R3 2 1 5
R4 1 3 5
*
.OP
.END
```

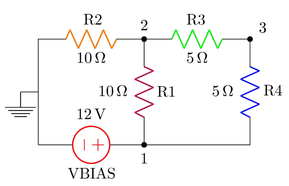

Fig. 1. Example of a supported SPICE netlist and corresponding circuit

Library Architecture

Our library was designed with portability in mind and targets .NET Standard 2.0 to support the widest possible range of platforms.

Transient analysis is only one of many types of circuit analyses commonly supported by SPICE-like simulators. We have designed our library in a way that allows more analysis types to be added in the future. Since each type of analysis models different characteristics of circuit devices, we have split our library into two parts (fig. 2). The core part of the library contains the analysis independent logic like creating and representing electrical circuits, the other part implements the transient analysis logic. This way, new circuit analysis types (like AC frequency sweep analysis) can be added without breaking backwards compatibility.

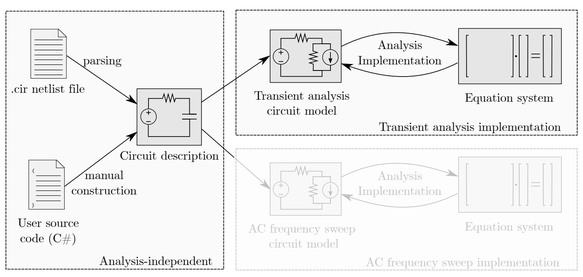

Fig. 2. Splitting the library to individual parts

Transient Analysis Implementation

For the implementation of transient analysis itself, we used standard *Modified Nodal Analysis* (MNA) method, which assigns each device type a fixed stamp which describes how the device affects the circuit equation matrix coefficients. Figure 2 shows the equation system matrix for the circuit shown earlier in figure 1.

$$\begin{bmatrix} \frac{1}{10} & 0 & -\frac{1}{10} & 0 & 0 & 0 & -1 \\ 0 & \frac{1}{10}+\frac{1}{5} & -\frac{1}{10} & -\frac{1}{5} & 0 & 1 & \\ -\frac{1}{10} & -\frac{1}{10} & \frac{1}{10}+\frac{1}{10}+\frac{1}{5} & -\frac{1}{5} & 0 & 0 & \\ 0 & -\frac{1}{5} & -\frac{1}{5} & \frac{1}{5}+\frac{1}{5} & 0 & 0 & \\ -1 & 1 & 0 & 0 & 0 & 0 & \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ I_V \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 12 \end{bmatrix}$$

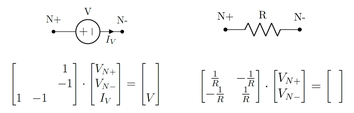

Fig. 2. Example of an equation system matrix created by MNA and stamps for voltage sources and resistors

For simplicity, we store the whole matrix as a two-dimensional array and use *Gaussian Elimination* for solving the equation system. However, abstraction used in our implementation (fig. 4) allows replacing the solver backend and even the equation system representation.

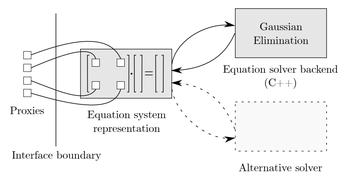

Fig. 4. Equation system abstraction

Simulating Complex Devices

Simulating circuits with nonlinear devices (such as diodes) requires solving nonlinear equation systems. To solve such systems, we have used the *Newton-Raphson method*.

For simulating the transient effects of energy storing devices (inductors, capacitors), we implemented several methods of numerical integration (*GEAR*, *Trapezoidal Rule*, *Backward Euler*, *Adams-Moulton*). Users of our library can choose the one that is most appropriate for the given circuit.

Lastly, the whole library is designed such that new devices can be easily added, the necessary steps are demonstrated on a simple diode device in the thesis text.

Console SPICE-Like Interface

In addition to the library, we have implemented a simple .NET Core 2.0 console application which enables features of our library to be used via command line interface like the original SPICE program.

Results

Our NextGen SPICE simulator was able to simulate several circuits with results visually indistinguishable from the results from other existing simulators, such as ngspice. Example of these circuits include saturating complementary flip-flop (cfflop), A stable multivibrator (astable) and cascaded RTL inverters (rtlinv). The cfflop circuit (fig. 7) with plots from both ngspice (fig. 5) and our NextGen SPICE simulator (fig. 6) are shown below.

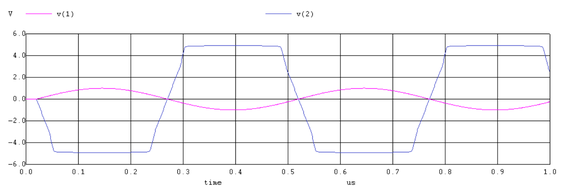

Fig. 5. ngspice output on cfflop circuit

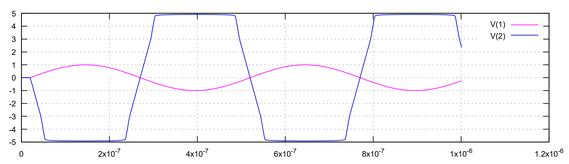

Fig. 6. NextGen SPICE output on cfflop circuit

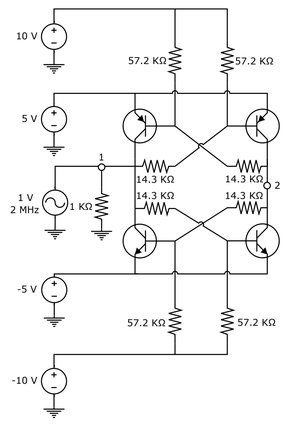

Fig. 7. Saturating complementary flip-flop circuit (cfflop)

There are also some circuits, which are difficult to simulate because their circuit equation matrix is ill-conditioned. On most today used simulators, the truncation errors accumulated during the circuit equation solution then lead to numerical noise. One such circuit is the backtoback circuit (fig. 8), on which both ngspice (fig. 9) and NextGen SPICE simulator (fig. 10) with double precision produce noisy output.

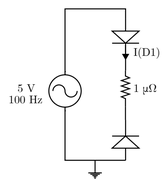

Fig. 8. backtoback circuit

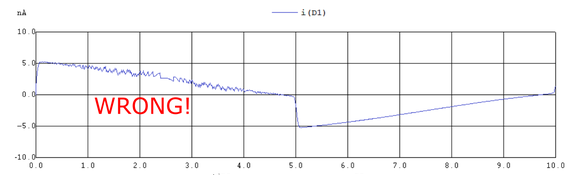

Fig. 9. ngspice output on backtoback circuit

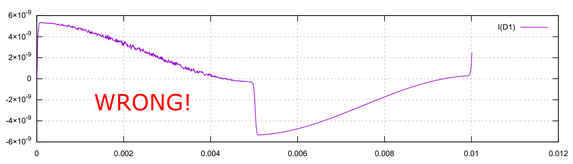

Fig. 10. NextGen SPICE (double) output on backtoback circuit

When the same simulation is run on our NextGen SPICE simulator with double-double precision, the noise dissapears and the simulator produces correct results according to the physics interpretation of the circuit, as shown in figure 11.

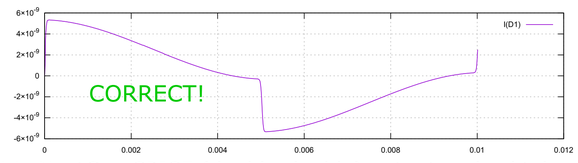

Fig. 11. NextGen SPICE (double-double) output on backtoback circuit

Furthermore, use of double-double precision significantly increased the range of circuits our simulator was able to simulate. Among these circuits were e.g. circuit simulating a four-bit adder (adder), Schottky-barrier TTL inverter (sdbgate) or RCA3040 wideband amplifier (rca3040).

However, the use of double-double precision comes at a cost. Our measurements on sample circuits showed that the simulation using double-double was on average approximately 5x slower then standard double precision was used.

	Double	Double-double	Slowdown
backtoback	2.640 ms	6.696 ms	2.54
cfflop	16.493 ms	57.694 ms	3.50
astable	14.78 ms	178.91 ms	12.11
rtlinv	549.5 µs	1 632.0 µs	2.97

There is therefore a trade-off between precision and speed and the decision which precision type should be used depends on the simulated circuit and the user's needs.

We have also compared the simulation times of our simulator with that of ngspice. Even though our NextGen SPICE simulator gives the same or even more precise results, it is many times slower than ngspice. The observed slowdown seems to be proportional to the number of variables in the equation system of the simulated circuits.

	ngspice	NextGen SPICE (double-double)	Slowdown
cfflop	11 ms	57 ms	5.18
rca3040	8 ms	150 ms	22.50
sbdgate	6 ms	130 ms	21.67
adder	67 ms	6147 ms	91.75

This result is not surprising because we used the simplest possible implementation of equation system – storing the matrix as a two-dimensional array. This representation is very inefficient for the sparse matrices of circuit equations. Performance profiling showed that during simulation of the adder circuit, our simulator spends more than 95% of the time solving the circuit equation system.

However, because we have anticipated this issue during the library's development, we have designed the library in a way that allows the equation system representation to be easily replaced in future versions.

Example of Library API

```
var builder = new CircuitBuilder();
builder
    .AddVoltageSource(1, 0, 12)
    .AddResistor(0, 2, 10)
    .AddResistor(1, 2, 10)
    .AddResistor(2, 3, 5)
    .AddResistor(1, 3, 5);
var circuit = builder.BuildCircuit();
var model = circuit.GetLargeSignalModel();
model.EstablishDcBias();
Console.WriteLine(model.NodeVoltages[1]);
Console.WriteLine(model.NodeVoltages[2]);
Console.WriteLine(model.NodeVoltages[3]);
```

This source code prints the values of node voltages from the circuit shown on fig. 1.

Output:

```
12
8
10
```

Conclusion

We managed to achieve all of the goals we set for our thesis. We implemented an extensible circuit simulator library capable of producing results comparable to other SPICE-like simulators and for some circuits even more precise. Even though our library seems slow in comparison with other simulators, it is usable for simulating small circuits. Also, it is possible to improve simulation speed in next version of the library by implementing methods more appropriate for sparse matrices.

Contact

r.zikmund.rz@gmail.com
https://github.com/rzikm/NextGenSpice

CHARLES UNIVERSITY
Faculty of mathematics and physics