

Current and Future Directions in Simulator Development

Jaijeet Roychowdhury

University of Minnesota

Circuit Simulation

- Has become important (again!)
 - high speed, DSM: digital becoming more analog
 - analog/RF/mixed-signal design: integrated RF, SoCs
 - effects of packaging, interconnect: analog
 - system-level, eye-diagram, BER simulation
 - automated nonlinear macromodelling
 - I/O drivers, oscillators/PLLs, mixers, ...
- R&D in circuit simulation has been on the rise

Why a Common Shared Simulation Infrastructure is Needed

- Anyone working in simulation today needs ...
 - device models: BSIM, MOS1, MOS2, MOS3, MOS9, Gummel-Poon, ...
 - base algorithms: **robust nonlinear solution**, transient, HB/TD steady-state, Krylov-subspace implementations, ...
 - parsing, equation formulation, output, ..
- A good, openly-available infrastructure will avoid the **huge** (waste of) effort of re-development of these basic capabilities!

Why not use SPICE?

- SPICE: the original open-source simulator
 - de-facto standard
- To be useful: modular, well-structured, flexible
 - separated numerics, algorithms, models, I/O
 - simple, clean interfaces
 - short, easy to read, easy to modify
- *i.e.*, not SPICE!

excerpt from dioload.c (SPICE3)

```
#ifdef SENSDEBUG
    printf("vd = %.7e \n", vd);
#endif /* SENSDEBUG */
    goto next1;
}
if(ckt->CKTmode & MODEINITSMSIG) {
    vd= *(ckt->CKTstate0 + here->DIOvoltage);
} else if (ckt->CKTmode & MODEINITTRAN) {
    vd= *(ckt->CKTstate1 + here->DIOvoltage);
} else if ( (ckt->CKTmode & MODEINITJCT) &
            (ckt->CKTmode & MODETRANOP)
            && (ckt->CKTmode & MODEUIC) ) {
    vd=here->DIOinitCond;
} else if ( (ckt->CKTmode & MODEINITJCT) && here->DIOoff)
{
    vd=0;
} else if ( ckt->CKTmode & MODEINITJCT) {
    vd=here->DIOtVcrit;
} else if ( ckt->CKTmode & MODEINITFIX && here->DIOoff) {
    vd=0;
} else {
#ifdef PREDICTOR
    if (ckt->CKTmode & MODEINITPRED) {
```

Why Modularity is Necessary

- Key to managing complexity
 - limits what each developer needs to know about the system
 - the “API” encapsulates rest of the simulator
 - Eg, to implement BSIM6.4, you don't need to know all about:
 - trapezoidal, Gear, etc. algorithms
 - shooting, harmonic balance, sensitivity, ...
 - Newton-Raphson, homotopy, ...
- Errors and mistakes reduced

How SPICE Got To Be The Way It Is

- Development without modularity is feasible if ...
 - one person understands it all (takes months if not years to get there)
 - simulator capabilities limited to relatively few
 - (entire PhD dedicated to maintaining simulator!)
- SPICE: series of one-person efforts
 - SPICE2G6 (== “SPICE”): Larry Nagel, Ellis Cohen
 - SPICE3: Tom Quarles
- All other SPICE development: piecemeal efforts
 - every new device model (eg, BSIM, lossy transmission lines)
 - every new analysis (eg, sensitivity, Volterra series)

Modularity: Is It Really Achievable?

- Existence Theorem: Linux
 - eg, the linux kernel
 - 65MB of *modular, well-organized* source code!
 - Thousands of complex, disparate devices supported
 - 10s of millions of users
 - Totally distributed open-source development
- Analog simulation infrastructure: much easier!
 - several open source efforts already on:
 - gnuicap, ngSPICE, fREEDA, gEDA, ...

gEDA

File Edit View Go Bookmarks Tools Window Help

http://geda.seul.org/

Home Shop Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktplace

gEDA Homepage

General

- [News](#)
- [Screenshots](#)
- [General FAQ](#)
- [Tools](#)
- [Tools FAQ](#)
- [Symbol Library](#)
- [Symbol Upload](#)
- [Documentation](#)
- [Mailing lists](#)
- [Historical](#)

Software

- [Source](#)
- [Binary](#)
- [Anon CVS](#)
- [Web CVS](#)
- [Bugs](#)

Project

- [People](#)
- [Task list](#)
- [Press](#)

Website

- [Links](#)
- [OC](#)
- [Mirrors](#)
- [Search](#)
- [SEUL](#)

What is gEDA?

The gEDA project is working on producing a full GPL'd suite of Electronic Design Automation tools. These tools are used for electrical circuit design, schematic capture, simulation, prototyping, and production. The gEDA project was started because of the lack of free EDA tools for UNIX. The tools are being developed mainly on GNU/Linux machines, but considerable effort is being made to make sure that gEDA runs on other UNIX variants. For a complete list of freely available tools please be sure to visit [Open Collector](#).

Quick links to the tools:

- [gerbv](#) – Gerber/Excellon file viewer
- [gnetlist](#) – Netlist generation (part of [gaf](#))
- [gnucap](#) – GPLed mixed-mode/mixed-level circuit simulator
- [gschem](#) – Schematic capture (part of [gaf](#))
- [GTKWave](#) – Electronic waveform viewer
- [gwave](#) – Analog waveform viewer
- [Icarus](#) – Verilog simulation and synthesis tool
- [ng-spice](#) – An improved SPICE

For the latest news regarding gEDA and associated software please see the [News](#) page.

Please see the [mailing list](#) archives for current status and information.

Website Mirrors

gnucap

File Edit View Go Bookmarks Tools Window Help

http://www.gnu.org/software/gnuCap/

Home Shop Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktplace

GnuCap Home Page - The Gnu Circ...

About

[What is it \(more detail\)](#)
[Mission Statement](#)
[Mailing lists](#)
[History](#)
[Contributors](#)
[Free Software Foundation](#)

Docs

[Manual in html](#)
[Manual in pdf](#)

Download

[Releases](#)
[GNU FTP site](#)
[Mirrors](#)

Other software

[Open Collector](#)
[gEDA](#)

Development

[Contributing](#)
[Open projects](#)
[Work in progress](#)

Welcome to the GnuCap home page!

GnuCap is the Gnu Circuit Analysis Package.

The primary component is a general purpose circuit simulator. It performs nonlinear dc and transient analyses, fourier analysis, and ac analysis. It is fully interactive and command driven. It can also be run in batch mode or as a server. Spice compatible models for the MOSFET (level 1-7), BJT, and diode are included in this release.

GnuCap is not based on Spice, but some of the models have been derived from the Berkeley models.

Unlike Spice, the engine is designed to do true mixed-mode simulation. Most of the code is in place for future support of event driven analog simulation, and true multi-rate simulation.

If you are tired of Spice and want a second opinion, you want to play with the circuit and want a simulator that is interactive, you want to study the source code and want something easier to follow than Spice, or you are a researcher working on modeling and want automated model generation tools to make your job easier, try GnuCap.

New features in the current release (0.31)

- BJT model.
- "Binning" for all MOS models.
- Internal element: non-quasi-static poly-capacitor. (needed by BJT).
- Enhancements to the data structures and model compiler to support binning in general.
- A line prefixed by "*>" is not ignored, in spite of the fact that "*" usually begins a comment. This is a deliberate incompatibility with Spice. If you prefix a line by "*>" it will be interpreted as a non-comment in GnuCap, but a comment in Spice.
- Circuit line prefixes of ">" and command prefixes of "-->" are ignored. This is so you can copy and paste whole lines, without having to manually remove the prompt string.

News/Announcements

Document: Done (1.411 secs)

fREEDA

File Edit View Go Bookmarks Tools Window Help

http://freeda.org/

Home Shop Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktplace

http://freeda.org/

fREEDA™

The new wave in circuit simulation

Links>>

- People
- Download
- Documentation
- Screenshots

NC STATE UNIVERSITY

KEY FEATURES:

- Supports DC, Time-Marching Transient, Harmonic Balance, Wavelet based Transient, Convolution based Transient and AC(linear only) analyses types
- Netlist Based Simulator
- Support for electro-thermal simulations (several thermal elements available).
- Written in C++ using OOP techniques
- Supports SPICE analyses forms
- Linux / UNIX compatible
- Simple addition and removal of circuit elements
- Supports subcircuits, .model statement
- Supports Local Reference Nodes
- Java GUI


System Requirements

- gcc 2.95.2 or newer is required. Other ANSI C/C++ compilers may be used if they support enough of the standard.
- A Fortran 77 compiler (g77 or OK)
- GNU Make utility
- GNU flex and bison utilities
- GNU bash shell (some versions of sh may also work)
- The Java run time environment 1.4.0 or greater to run the GUI, or the Java development kit to make modifications to it

Free DOWNLOAD

UNIX Users

- Unpack the tar file
- Run the shell script

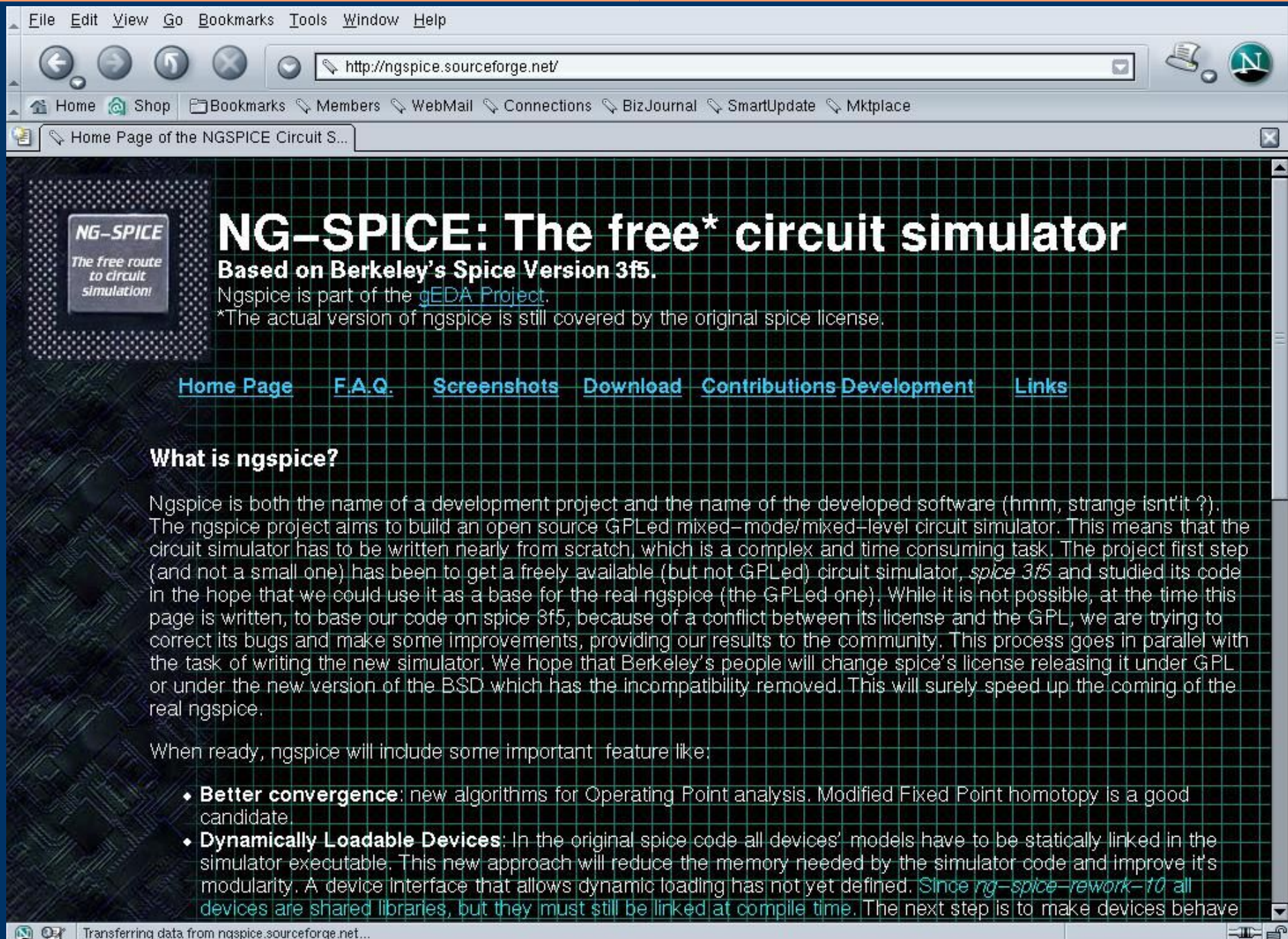


LINUX / UNIX

fReeda™ is distributed under the Gnu Public License

Transferring data from guppie.egrc.ncsu.edu...

ngSPICE



File Edit View Go Bookmarks Tools Window Help

http://ngspice.sourceforge.net/

Home Shop Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktplace

Home Page of the NGSPICE Circuit S...

NG-SPICE
The free route to circuit simulation!

NG-SPICE: The free* circuit simulator

Based on Berkeley's Spice Version 3f5.
Ngspice is part of the [gEDA Project](#).
*The actual version of ngspice is still covered by the original spice license.

[Home Page](#) [F.A.Q.](#) [Screenshots](#) [Download](#) [Contributions](#) [Development](#) [Links](#)

What is ngspice?

Ngspice is both the name of a development project and the name of the developed software (hmm, strange isn't it?). The ngspice project aims to build an open source GPLed mixed-mode/mixed-level circuit simulator. This means that the circuit simulator has to be written nearly from scratch, which is a complex and time consuming task. The project first step (and not a small one) has been to get a freely available (but not GPLed) circuit simulator, *spice 3f5* and studied its code in the hope that we could use it as a base for the real ngspice (the GPLed one). While it is not possible, at the time this page is written, to base our code on spice 3f5, because of a conflict between its license and the GPL, we are trying to correct its bugs and make some improvements, providing our results to the community. This process goes in parallel with the task of writing the new simulator. We hope that Berkeley's people will change spice's license releasing it under GPL or under the new version of the BSD which has the incompatibility removed. This will surely speed up the coming of the real ngspice.

When ready, ngspice will include some important feature like:

- **Better convergence:** new algorithms for Operating Point analysis. Modified Fixed Point homotopy is a good candidate.
- **Dynamically Loadable Devices:** In the original spice code all devices' models have to be statically linked in the simulator executable. This new approach will reduce the memory needed by the simulator code and improve its modularity. A device interface that allows dynamic loading has not yet defined. *Since ng-spice-rework-10 all devices are shared libraries, but they must still be linked at compile time.* The next step is to make devices behave

Transferring data from ngspice.sourceforge.net...

Benefits of Open Source

- Coalesces scattered resources
- Quick deployment of prototypes to users
- Reproducibility of published results
- Code development becomes motivating, fun
 - empowers users and developers
 - high quality: the “many-eyes effect”
- Helps benchmarking
- Suggested reading
 - Eric Raymond: “The Cathedral and the Bazaar”
 - Steven Webber: “The Success of Open Source”

Benefits for Academia

- Research

- infrastructure currently major barrier to research
- shared, open platform: critical enabler

- Teaching

- 4 months: need easy to learn, easy to use infrastructure
- interesting and useful class projects
- “Hands-on” demos: effective teaching tool

Benefits for Industry and Users

- Companies developing CAD tools
 - access to tried-and-tested algorithms, reference implementations
 - improved pool of qualified developers
- Designers and users
 - shorter “time-to-market” for new ideas/methods
 - direct involvement (feedback) in simulator development
 - try (open-source) before you buy (commercial)

Existing Open-Source Simulators: Not As Modular As We'd Like!

- Encumbered by SPICE's structuring
 - centered around device models
 - clean separation of models, analyses, etc lacking
 - main difficulty: clean, intuitive coding of new analyses such as:
 - homotopy, automated macromodelling, multi-time, envelope, ...
- So: redesign simulator from scratch

Our MATLAB centered approach

- For learning/quick prototyping
- Dramatically reduces development time/pain
 - built-in numerical methods
 - sparse matrices, LU, iterative linear methods, ODE solution, FFTs, ...
 - Short, simple, intuitive
 - Interfaces to C/C++/Fortran
 - Push-button C-code generation
 - Built-in system-level functionalities (eg, Simulink)
- Cleanly separated devices, numerics, algorithms, I/O
 - HB implemented and debugged in 15 person-hours

Other MATLAB benefits

- Excellent output support
 - powerful, intuitive graphics
- Effective documentation
 - simple, intuitive help system
 - self-paced learning, interactive tutorials
- Designers familiar with MATLAB
 - heavily used in circuit/system exploration and design

Open Source Simulators: What Next?

- Open device models needed
 - currently: reverse-engineer SPICE3 code
 - promising: Verilog-A compact models
- Separate co-operative efforts: a good idea
 - cross-fertilization, friendly competition
 - *critical* for progress, sustainability
 - bad ideas rapidly exposed, killed
- Careful licensing policies needed
 - want to promote openness, contributions from all
 - want to help industry *benefit* from open-source