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 {
#ifndef PREDICTOR
         if (ckt->CKTmode & MODEINITPRED) {
```

Sensitivity analysis code

AC analysis code

Transient analysis related code

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

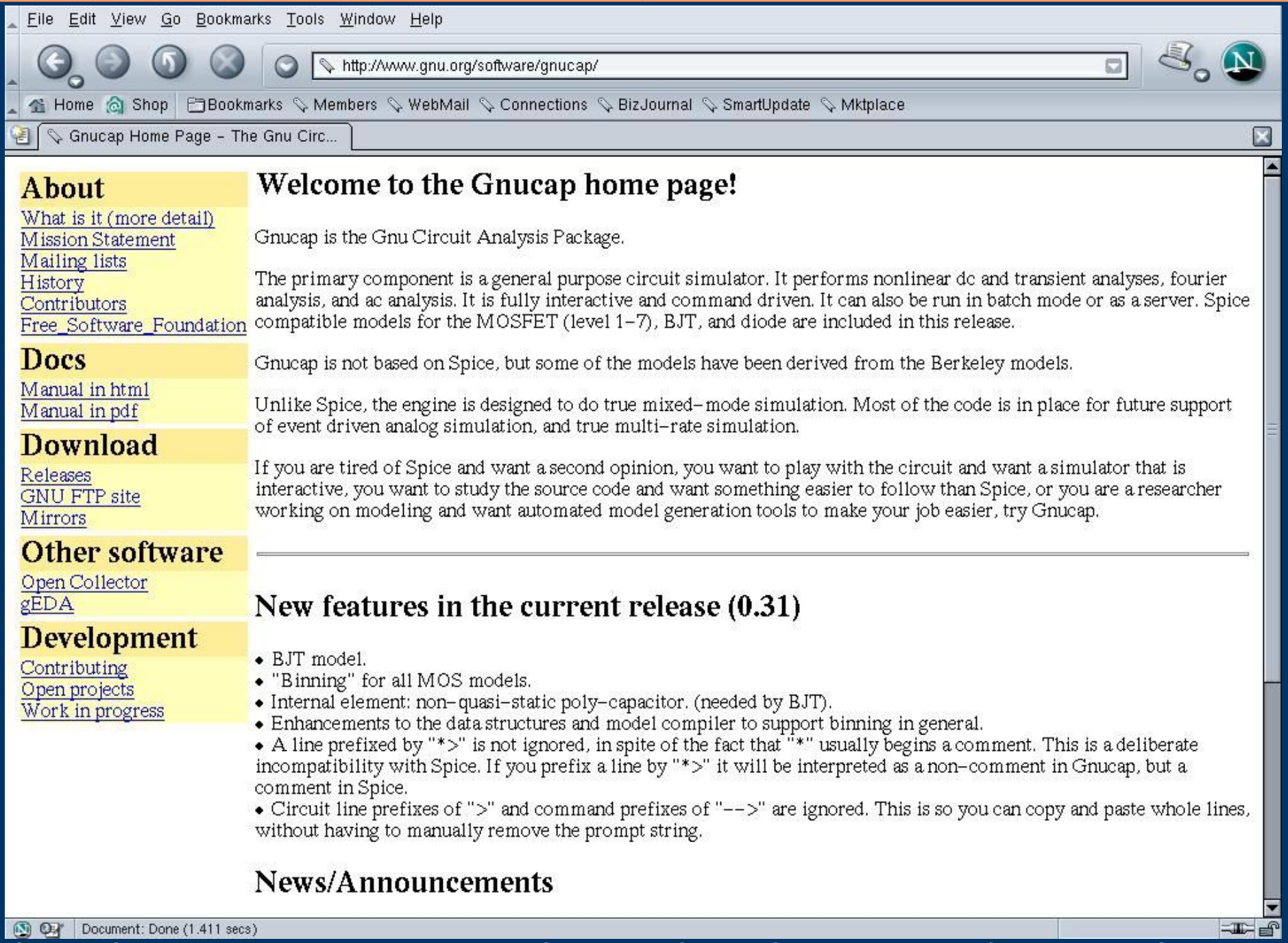
- Development without modularity is feasible if ...
  - one person understands it all (takes months if not years to get there)
  - simulator capabilities limited to relatively few
  - (entire PhD dedicated to maintaining simulator!)
- SPICE: series of one-person efforts
  - SPICE2G6 (== "SPICE"): Larry 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?

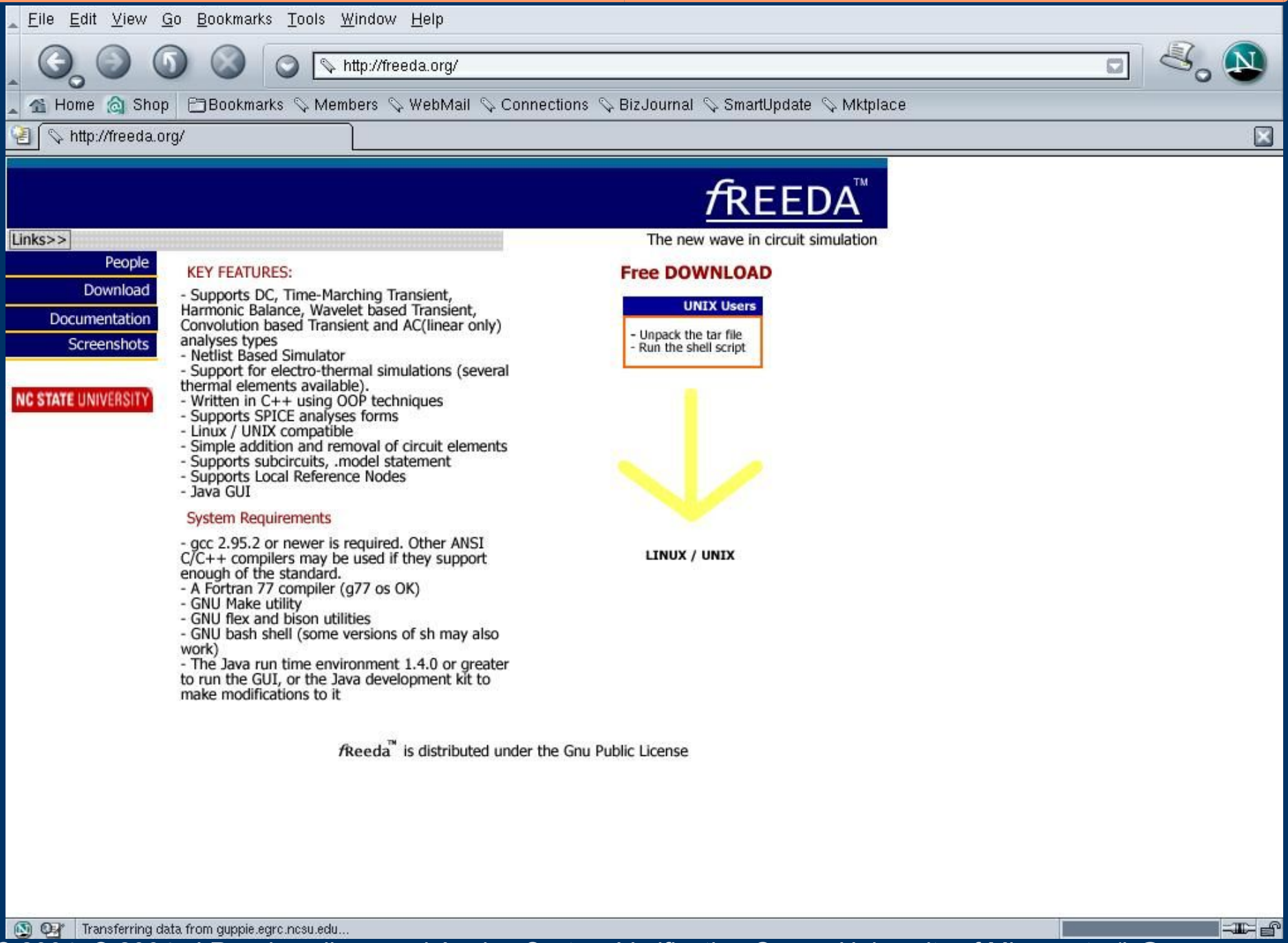
- Existence Theorem: **Linux**
  - eg, the linux kernel
    - 65MB of *modular, well-organized* source code!
  - Thousands of complex, disparate devices supported
  - 10s of millions of users
  - Totally distributed open-source development
- Analog simulation infrastructure: much easier!
  - several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

gEDA

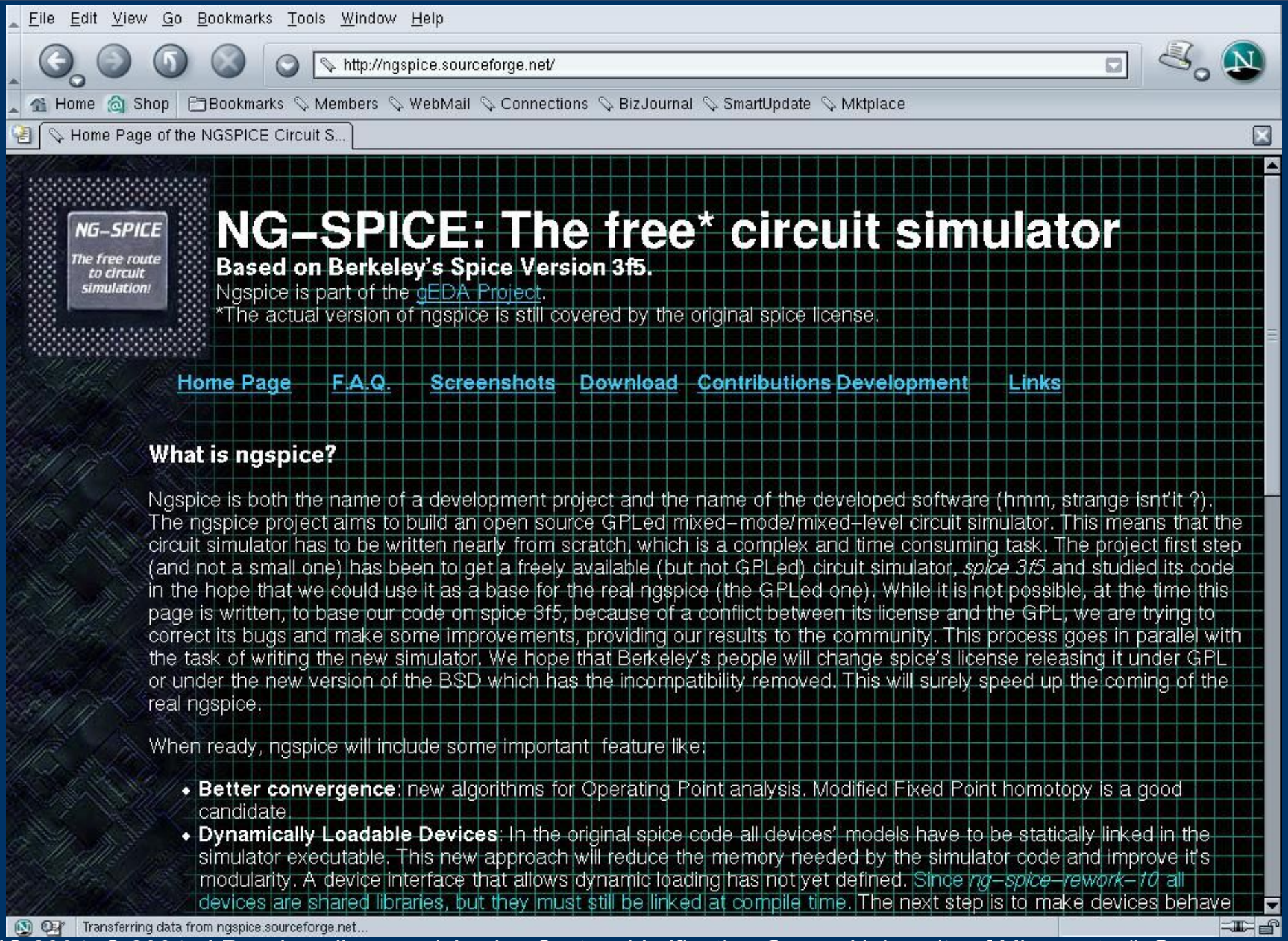
gnucap



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™
Links>>
People
Download
Documentation
Screenshots
NC STATE UNIVERSITY
The new wave in circuit simulation
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 os 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



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

