ngspice - current status and developments

Holger Vogt

Fraunhofer IMS, Duisburg, Germany
Universität Duisburg-Essen, Germany
holger.vogt@ims.fraunhofer.de

September 14-18, 2020
Contents

Introduction to ngspice
Core elements of the simulator
Some results
Current developments
CMOS inverter

.include ./bsim4soi/nmos4p0.mod
.include ./bsim4soi/pmos4p0.mod
.option TEMP=27C

Vpower VD 0 1.5
Vgnd VS 0 0

Vgate Ein VS PULSE(0 1.5 100p 50p 50p 200p 500p)

MN0 Aus Ein VS VS N1 W=10u L=0.18u
MP0 Aus Ein VD VS P1 W=20u L=0.18u

.tran 3p 600ps
.control
   run
.plot Ein Aus
.endc
.END

the circuit  the input  the output
ngspice introduction

- **ngspice – what is it**
  - Circuit simulator that numerically solves equations describing (electronic) circuits
  - Circuits are made of passive and active devices
  - The equations are solved for (time varying) currents and voltages

- **ngspice – what does it do**
  - Read and parse the circuit netlist, check for compatibility (PSPICE, HSPICE, …)
  - Set up the circuit matrixes
  - Solve the matrix, solve the device equations (for every time step)
  - Output the data (plotting, saving to disc, returning to calling program)
The two major application areas

**PCB design support**
- Circuits are made with a mix of ICs and discrete components

*Requirements:*
- Comfortable user interface (offered by third parties)
- PSPICE and LTSPICE model compatibility

KiCad, Altium, Eagle, PartSim, CoolCAD, PSIM, WeSpice ...

**IC design support**
- Circuits are made of (MOS) transistors and (parasitic) passive components

*Requirements:*
- BSIM 3, 4, HICUM models etc.
- Large circuit capability, speed
- HSPICE PDK compatibility

gEDA, Yosys, efabless, Isotel, XSCHEM, PySpice ...
Some impressions, ngspice-32

KiCad
Inputs

Postscript

KiCad
Outputs

XSchem

UNICODE axis labels

gnuplot
Three flavors of ngspice

Standard executable
- Command line input
- File and graphics output
- Control language

Shared library with tcl/tk interface
- Tcl command input
- Controlled by tcl scripts
- Blt library for graphics output

C shared library (so, dll)
- Input and output via exported functions and callbacks
- Caller has full control over (nearly) all internal variables
- Simulation may run in its own thread
- No graphics interface

OSs supported: Linux, Windows, macOS, (Solaris, BSD, Android, Wasm ...)
## Scripting with control language

### Controls
- ~100 commands
- 62 math functions
- 91 internally defined user controllable variables
- Loops etc.

### Applications
- To control simulation sequences, including plotting, measuring etc.
- Monte Carlo simulation

### Example script
```
*ng_script
* Script to run transient sim of adder-digital.control
source adder-digital.cir
tran 500p 64000n
rusage
display
edisplay
* save data to input directory
cd $inputdir
eprvcd 1 2 3 4 5 6 7 8 s0 s1 s2 s3 c3 > adder_x.vcd
* plotting the vcd file (e.g. with GTKWave)
shell start gtkwave adder_x.vcd --script nggtk.tcl.endc
```
KiCad/ngspice: Class A power amp example

Archetype: F6
by Nelson Pass

KiCad/EEschema input
Some results for the class A amp

**Procedure**
- Draw the circuit
- Set up a control script
- Run the simulation

**FFT of 1 kHz at 46 W**
- Mostly 3rd harmonics
- Ultra low noise floor

*True class A*
# Mixed signal simulation with XSPICE

<table>
<thead>
<tr>
<th>digital</th>
<th>analog</th>
</tr>
</thead>
<tbody>
<tr>
<td>event simulation</td>
<td>C coded models</td>
</tr>
<tr>
<td>fast</td>
<td>versatile</td>
</tr>
<tr>
<td>no analog values, but signal strength and delays</td>
<td>analog (and frequency domain)</td>
</tr>
<tr>
<td>23 predefined devices (gates like: nand, flip flops, latches, state machine, LUT)</td>
<td>29 predefined devices (e.g. limiter, multiplier, file source, integrator, table model ...)</td>
</tr>
<tr>
<td>third party interfaces to µC</td>
<td>7 hybrid (interface) devices</td>
</tr>
</tbody>
</table>
Compare the venerable 4-bit adder

<table>
<thead>
<tr>
<th>analog</th>
<th>hybrid</th>
<th>event-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>65 s BSIM3 CMOS, 2-input NAND, OpenMP, 4 threads</td>
<td>16 s XSPICE nand gates with analog interfaces</td>
<td>1.4 s XSPICE nand gates, digital interconnects</td>
</tr>
</tbody>
</table>

Analog, viewed with ngspice

VCD file, viewed with gtkwave
Current developments

Thermal modeling
HICUM bjt model
Verilog A interface
KLU matrix solver
SIMD evaluation of model equations
Thermal modeling

Things get hot when we consume electrical energy.

What about the devices and their parameters?

Let's model them with ngspice!

<table>
<thead>
<tr>
<th></th>
<th>Thermal term</th>
<th>Electrical term</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacitance</strong></td>
<td>$c$ [J/K]</td>
<td>$C$ [A/V]</td>
</tr>
<tr>
<td><strong>Conductance</strong></td>
<td>$g$ [W/K]</td>
<td>$1/R$ [1/Ω]</td>
</tr>
<tr>
<td><strong>Temperature/Voltage</strong></td>
<td>$T(t)$ [K]</td>
<td>$V(t)$ [V]</td>
</tr>
<tr>
<td><strong>Heat/Current</strong></td>
<td>$Q$ [W]</td>
<td>$I(t)$ [A]</td>
</tr>
<tr>
<td><strong>Time constant</strong></td>
<td>$c/g$ [s]</td>
<td>$RC$ [s]</td>
</tr>
</tbody>
</table>

Make use of the equivalence of electronic and thermal properties (circuits).
Translate thermal to electrical circuits, run both circuit parts in ngspice
Feedback loop: Electrical power dissipation generates heat, restricted heat flow raises temperature, temperature changes power dissipation.
Thermal modeling example

Excerpt from the previously shown power amp

Thermal circuit for heat sink: thermal resistances, capacitances, connected to transistor case

Power transistor with 5 terminals
- Electrical: D, G, S
- Thermal: Junction temp, case temp.

VDMOS model with self-heating [5]

The transistor temperature rises over time
Thermal runaway modeling

Resistor with negative temperature coefficient
1 Ω, 1 V, -0.03 Ω/K

Transient simulation: After some time the temperature rises beyond bounds
=> thermal runaway
device destruction
HICUM

Ultra-high speed silicon circuits for RADAR etc. apply SiGe bipolar transistors.
HICUM: Proven high speed bipolar model [1].

Project: Add HICUM 2.4 to ngspice [9]
Implementation directly into the C-code ngspice model interface.
Use advanced math (Dual Numbers) for efficient calculation of derivatives.
HICUM bipolar model overview

- Physics based, geometry scalable and large signal BJT/HBT model [1].
- Includes most to all relevant physical effects present in today's integrated BJTs:
  - Separate peripheral and internal elements for correct scaling and bias dependency
  - Self-Heating
  - Avalanche current model until BVCB0
  - Current dependent internal base resistance
  - Substrate transistor
  - Noise
- Model is CMC standard for SiGe BJT [2] and is used in industry PDKs.
- Applied and verified until 600 GHz.
HICUM implementation uses Dual Numbers

- Dual numbers are numbers with their own algebra similar to complex numbers.
- A dual number has a real part and a dual part. The dual part is indicated by \( \epsilon \).
- Dual numbers are available in C++ [4].
- The dual numbers can be used to calculate derivatives of complicated functions without having to write them out [3]

\[
\frac{df(x)}{dx} = \text{Dual}(f(x + \epsilon)).
\]

- The accuracy is equal to the analytical derivative, this is an advantage compared to the already known finite difference quotients.

For example:

\[
f(x) = 4x^2
\]

\[
f(2 + \epsilon) = 16 + \sqrt{16} \epsilon
\]

\[
\frac{df(2)}{dx}
\]
**Verilog A**

**Requirement:** make modern device models written in Verilog A efficiently available to ngspice

**Status:** currently available ADMS interface is incomplete, buggy and not very well documented

**Activity:** Create a compiler reading Verilog A device models and making them available to ngspice [10]
Simulator speed enhancements

Analysis of CPU load for digital circuit with back annotation [6]

**CPU time usage:**
- ~ 50% for device equations
- ~ 50% for solving matrix

**Developments already done:**
- OpenMP for device equations
- KLU for solving the matrix

**Research:**
SIMD for device equations
KLU

- Alternative matrix solver [7]
- KLU or Sparse 1.3 selected by option statement
- Faster device setup with KLU
- Faster matrix solver for large circuits

Result on circuit with 15k transistors (34k resistors, 48k capacitors)
i9 9600 8 cores, OpenMP enabled
Speed-up KLU versus Sparse by factor of 4 (transistors only)
Speed-up by factor of 3 (incl. back annotated RC parasitics)

Will be available with ngspice-33
SIMD

Exploit the capability of modern computer hardware and compilers for single instruction, multiple data (SSE, AVX2 ...).

Goal for ngspice: Speed-up the device equation evaluation by parallel processing on a single core.
SIMD

Procedure to establish SIMD:
- Replace doubles by vectors of doubles.
- Replace all math functions by their vector equivalents (using e.g. AVX2 intrinsics).
- Gather data to evaluating them in parallel.
- Scatter them back to the individual devices.

Preliminary results [8]:
- Speed-up may be a factor of 3.
- Depends on environment (OS, compiler, machine).
- Tools may still be buggy.
- HW and SW portability issues.
ngspice is a well established, proven simulator for discrete and integrated electronics.

It has been adopted by several PCB design tool makers.

Active development is going on towards models (HICUM), speed (KLU, SIMD), interfacing (Verilog A), and usability (thermal simulation, wave audio, etc.).

ngspice-33 is scheduled for October/November 2020
References

[6] F. Lannutti, P. Nenzi, M. Olivieri, KLU Sparse Direct Linear Solver Implementation into NGSPICE,
   MIXDES 2012, 19th International Conference "Mixed Design of Integrated Circuits and Systems",
   May 24-26, 2012, Warsaw, Poland.
[7] F. Lannutti, private communication, 2020,
   https://sourceforge.net/p/ngspice/ngspice/ci/klu2rebase/tree/
[8] F. Ballenegger, 2020,
   https://anamosic.com/images/SimdModEvalNgspice.pdf,
   https://sourceforge.net/p/ngspice/ngspice/ci/simd/tree/
[9] D. Warning, M. Müller and M. Krattenmacher, private communication,
   https://sourceforge.net/p/ngspice/ngspice/ci/hicum_vae/tree/,
   https://sourceforge.net/p/ngspice/ngspice/ci/hicum2-mario/tree/