Verilog-AMS in Gnucap

Felix Salfelder

FSIC 25

Content

Gnucap & historical context

- The Gnu Circuit Analysis Package
- Modular design, C++ codebase
- Small library with (user) extensions

Gnucap & historical context

- The Gnu Circuit Analysis Package
- Modular design, C++ codebase
- Small library with (user) extensions

History

- 1973-1989. SPICE 1-3: final: '93
- 1990. *ACS*, Al's Circuit Simulator
- 1992... GPL'd, The original "fast spice"
- 1995 Verilog (as we know it)
- 2000 Verilog-AMS, early traces
- 2001... ACS Renamed to *Gnucap*, a GNU project
- 2004 Larrys talk "Is it time for SPICE4"
- 2022 Gnucap supported by NLnet (ongoing)
- 2023 initial Modelgen-Verilog

Verilog-AMS (as of 2025)

- Open Standard, based on IEEE 1364-2005
- Analog, digital and in between
- Popular Hardware description language

Verilog-AMS (as of 2025)

- Open Standard, based on IEEE 1364-2005
- Analog, digital and in between
- Popular Hardware description language

Different sections for different applications

- Structural Verilog: Circuits and Netlists
- Verilog-95: Very popular in digital domain
- Verilog-A: from “SPICE” models up to system level
- Verilog-AMS: bridge -A to -95
- System-Verilog-AMS: more to come

Modelgen-Verilog: Overview

- Model compiler of the Gnucap project
- Inspired by modelgen which inspired Verilog-AMS..
- Goal: models for VLSI-ready simulation
- Approach: Turn models into plugin code

Modelgen-Verilog: Overview

- Model compiler of the Gnucap project
- Inspired by modelgen which inspired Verilog-AMS..
- Goal: models for VLSI-ready simulation
- Approach: Turn models into plugin code

Key Features

- Modular design, retargettable
- Code generated by program
  (not Bytecode, not filling templates)
- Add Verilog-AMS behavioural modelling to Gnucap
  (and what Verilog-AMS entails)

Verilog-AMS coverage in Modelgen-Verilog

Basic analog modelling

- ports, parameters
- control blocks
- integrate, differentiate
- contributions
- Small signal AC & noise

Verilog-AMS coverage in Modelgen-Verilog

Basic analog modelling

- ports, parameters
- control blocks
- integrate, differentiate
- contributions
- Small signal AC & noise

Beyond SPICE/analog

- hierarchy (aka. subdevices)
- (compiled) paramset, binning
- analog filters, freq. domain modelling
- events, switching branches etc.
- Some digital: logic primitives, UDP

Achievements and progress since FSiC '24

New deep stuff, impossible without funding (NLnet!)

- Pluggable matrix solver
- Quantized time in event driven simulation
- Compliant logic primitives, UDP
- Predictor, prepare for multi-rate
- Ground statement, same-port devices etc.
- Per device integration method
- Hierarchical temperature
- Numerous improvements (see resp. NEWS files)

Achievements and progress since FSiC '24

New deep stuff, impossible without funding (NLnet!)

- Pluggable matrix solver
- Quantized time in event driven simulation
- Compliant logic primitives, UDP
- Predictor, prepare for multi-rate
- Ground statement, same-port devices etc.
- Per device integration method
- Hierarchical temperature
- Numerous improvements (see resp. NEWS files)

Under review/testing

- pluggable node ordering
- selective trace algorithm
- connect module placement

Achievements and progress since FSiC '24

New deep stuff, impossible without funding (NLnet!)

- Pluggable matrix solver
- Quantized time in event driven simulation
- Compliant logic primitives, UDP
- Predictor, prepare for multi-rate
- Ground statement, same-port devices etc.
- Per device integration method
- Hierarchical temperature
- Numerous improvements (see resp. NEWS files)

Under review/testing

- pluggable node ordering
- selective trace algorithm
- connect module placement

Side tracks

- Qucs: revival & Verilog-S dump
- Verilog-AMS QA (in preparation)

Data exchange I: circuits

How To: Vendor independent circuit storage

- named device prototypes
- named port connections
- named instance parameters
- optional: metadata aka. attributes
- syntax: human readable

Data exchange I: circuits

How To: Vendor independent circuit storage

- named device prototypes
- named port connections
- named instance parameters
- optional: metadata aka. attributes
- syntax: human readable

Existing ideas

- ~~SPICE netlist & friends~~
- ~~gEDA/lepton, Qucs, xschem etc. drawings~~
- ~~XML, Json conglomerates~~
- Structural Verilog

Data exchange I: circuits

How To: Vendor independent circuit storage

- named device prototypes
- named port connections
- named instance parameters
- optional: metadata aka. attributes
- syntax: human readable

Existing ideas

- ~~SPICE netlist & friends~~
- ~~gEDA/lepton, Qucs, xschem etc. drawings~~
- ~~XML, Json conglomerates~~
- Structural Verilog

```
(* spice=R *) resistor #(.r(1k)) r1(.p(a), .p(b));
```

Data exchange II: schematics

Well established graphical representation

- Edit circuits without text editor
- Introduce beginners, 1llustrate concepts
- Often: Lock-in ("can I get my circuit?" – "no!")

Data exchange II: schematics

Well established graphical representation

- Edit circuits without text editor
- Introduce beginners, 1llustrate concepts
- Often: Lock-in ("can I get my circuit?" – "no!")

We also want schematics, but done right

- Schematics are circuits, not drawings
- Screen positions is markup, the circuit is primary
- Store schematics as circuits (see above)
- Pick a syntax, here: Verilog
- add few markup conventions: Verilog-S

Data exchange II: schematics

Well established graphical representation

- Edit circuits without text editor
- Introduce beginners, 1llustrate concepts
- Often: Lock-in (“can I get my circuit?” – “no!”)

We also want schematics, but done right

- Schematics are circuits, not drawings
- Screen positions is markup, the circuit is primary
- Store schematics as circuits (see above)
- Pick a syntax, here: Verilog
- add few markup conventions: Verilog-S

```
(* S0_x_p=42, S0_y_p=100 *)
resistor #(.r(1k)) r1(.p(a), .p(b));
```

Data exchange III: implementation status

gnucap-geda: use gEDA/lepton schematics

- Well organised format, but missing circuit
- Reconstruct connections (using device library)
- Dump into Verilog-S for use without gEDA/lepton
- Roundtrip: now implemented. Lossless is possible

Data exchange III: implementation status

gnucap-geda: use gEDA/lepton schematics

- Well organised format, but missing circuit
- Reconstruct connections (using device library)
- Dump into Verilog-S for use without gEDA/lepton
- Roundtrip: now implemented. Lossless is possible

Qucs: towards a native Verilog-AMS GUI

- Native files are too messy to use, hence
- Add code to dump Verilog-S, then parse.
- No more need for messy files.
- Replace messy file format (nearly there)

Data exchange III: implementation status

gnucap-geda: use gEDA/lepton schematics

- Well organised format, but missing circuit
- Reconstruct connections (using device library)
- Dump into Verilog-S for use without gEDA/lepton
- Roundtrip: now implemented. Lossless is possible

Qucs: towards a native Verilog-AMS GUI

- Native files are too messy to use, hence
- Add code to dump Verilog-S, then parse.
- No more need for messy files.
- Replace messy file format (nearly there)

Plans (Volunteers?)

- Lossless transfer between the two (including symbols)
- Import from others. KiCAD? XSchem?
- Upcycle data from commercial tools.

Conclusion & Outlook

- New algorithms taking shape
- Towards free/libre Verilog-AMS implementation
- Demonstrated standardisation and interoperability

Conclusion & Outlook

- New algorithms taking shape
- Towards free/libre Verilog-AMS implementation
- Demonstrated standardisation and interoperability

New Grant awarded for 2025/2026

- More of Verilog-AMS in the pipeline
- Revamp of the SPICE subsystem, B sources etc.
- Will follow user requests: e.g. VCD output
- Tackling speed issues

Conclusion & Outlook

- New algorithms taking shape
- Towards free/libre Verilog-AMS implementation
- Demonstrated standardisation and interoperability

New Grant awarded for 2025/2026

- More of Verilog-AMS in the pipeline
- Revamp of the SPICE subsystem, B sources etc.
- Will follow user requests: e.g. VCD output
- Tackling speed issues

Internship funding available, too

- OpenVAF wrapper? KLU wrapper?
- Revisit convolution based filters?
- Anything really.

Thanks!

Questions?