Update on Verilog-AMS in Gnucap

Felix Salfelder

FSIC 24

Content

- Gnucap, Verilog, what is it?
- Gnucap & Modelgen Update
- Verilog-A coverage in Modelgen
- Available models, update
- License amendments
- Revisit paramset
- Benchmaking and outlook
- Recent user contributions
- Conclusion

Gnucap & 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
- 2022 supported by NLnet (ongoing)
- 2023 initial Modelgen-Verilog

Gnucap Plug-In interface

- avoid monolith, allow user contributions
- no need to start from scratch
- endless customisation and experimentation
- present research without maintenance burden

Verilog-AMS (as of 2014)

- Based on Verilog, IEEE Std 1364-2005
- Analog, digital and in between
- Pull back things that drifted apart
- No known complete implementation

Intended: unification

- Verilog-A: ready for system level analog
- Verilog-AMS build brige to Verilog-95
- SPICE subsystem preserved. Accessible from Verilog-A

Verilog-AMS (as of 2014)

- Based on Verilog, IEEE Std 1364-2005
- Analog, digital and in between
- Pull back things that drifted apart
- No known complete implementation

Intended: unification

- Verilog-A: ready for system level analog
- Verilog-AMS build brige to Verilog-95
- SPICE subsystem preserved. Accessible from Verilog-A
  (akin to extern "C" in C++, or
  call Fortran function from C program)

... as opposed to

- Verilog-95: digital model description
- SPICE: here goes the analog
- Verilog-A: build compact models for use in SPICE

Modelgen-Verilog: overview

- Intent: A replacement for ADMS
- Standardisation: Verilog-AMS
- Goal: VLSI-ready simulation
- Inspired by modelgen architecture
- Status: Way ahead CMC subset of Verilog-A

Main Features

- Modular design, retargettable
- Code generated by program, not template
- Have Verilog-AMS and Gnucap plugin output
- .. programmed in suitable language

Roadmap

- Discrete subset of Verilog-AMS
- Catch up with simple optimisations
- Support SPICE input (.subckt), maybe SPICE output (C)

Verilog-A coverage in Modelgen-Verilog

- Flow/Potential contributions, switches, named branches
- Analog primitives and filters
  basic arithmetic, idt, ddt
  slew, absdelay laplace_*, zi_*
  ac_stim, white & flicker_noise
- Analog control structures
- hierarchy (aka. subdevices)
- Compiled paramset
  some data flow analysis
  topological collapse, dead code elimination
- Convergence checking (from legacy modelgen)

Current WIP

- transition, last_crossing, noise_tables
- discrete subset, connect modules
- generate

Gnucap, recent model updates

Spice-wrapper: Use SPICE3 models (C interface)

- automated tests
- new: also wrapping noise sources
- cover ngspice models until 42 etc.

Gnucap, recent model updates

Spice-wrapper: Use SPICE3 models (C interface)

- automated tests
- new: also wrapping noise sources
- cover ngspice models until 42 etc.

SPICE primitives (Table E.1)

- implemented in Verilog-A
- requirement for compliance, also: test piece
- have: R, L, C, D, trln, vsine. WIP: mosfet (harder)

Gnucap, recent model updates

Spice-wrapper: Use SPICE3 models (C interface)

- automated tests
- new: also wrapping noise sources
- cover ngspice models until 42 etc.

SPICE primitives (Table E.1)

- implemented in Verilog-A
- requirement for compliance, also: test piece
- have: R, L, C, D, trln, vsine. WIP: mosfet (harder)

"CMC models"

- Verilog-A subset agreed upon by council
- mostly work unmodified with Gnucap, need more testing

Gnucap, recent model updates

Spice-wrapper: Use SPICE3 models (C interface)

- automated tests
- new: also wrapping noise sources
- cover ngspice models until 42 etc.

SPICE primitives (Table E.1)

- implemented in Verilog-A
- requirement for compliance, also: test piece
- have: R, L, C, D, trln, vsine. WIP: mosfet (harder)

"CMC models"

- Verilog-A subset agreed upon by council
- mostly work unmodified with Gnucap, need more testing

Few more QUCS devices in Gnucap/Gnucsator

- "SPfile", "TLIN", "MLIN", "Inoise", "Vnoise"
- (Exploring Verilog-AMS integration, more later)

License update

- What license is the modelgen output subject to?
- Can I distribute binaries without source?
- “but it’s under GPL?!”.

License update

- What license is the modelgen output subject to?
- Can I distribute binaries without source?
- "but it's under GPL?!".

From now on:

- The modelgen output is subject to license of the input (Bug Accelera about their disciplines header, or use ours.)
- Same for binaries compiled/linked with modelgen+gcc. By means of a "linking exception"
- This applies to unmodified modelgen

Revisit paramset

- FSiC'23: paramset overloading explained
- Now: paramset in the model compiler

Revisit paramset

- FSiC'23: paramset overloading explained
- Now: paramset in the model compiler
- Use optimised binaries where applicable
  .. following overloading rules.

Revisit paramset

- FSiC'23: paramset overloading explained
- Now: paramset in the model compiler
- Use optimised binaries where applicable
  .. following overloading rules.
- 10-100x speedup (depends on model, not simulator or compiler)

Revisit paramset

- FSiC'23: paramset overloading explained
- Now: paramset in the model compiler
- Use optimised binaries where applicable
  .. following overloading rules.
- 10-100x speedup (depends on model, not simulator or compiler)

Standard needs amendment to streamline paramset use (RFC).

- All devices and models are just prototypes
- Regardless of their internal structure
- A device instance resolves to the suitable one
- Parameters do not need default values (to be implemented)

Making sense of $param_given

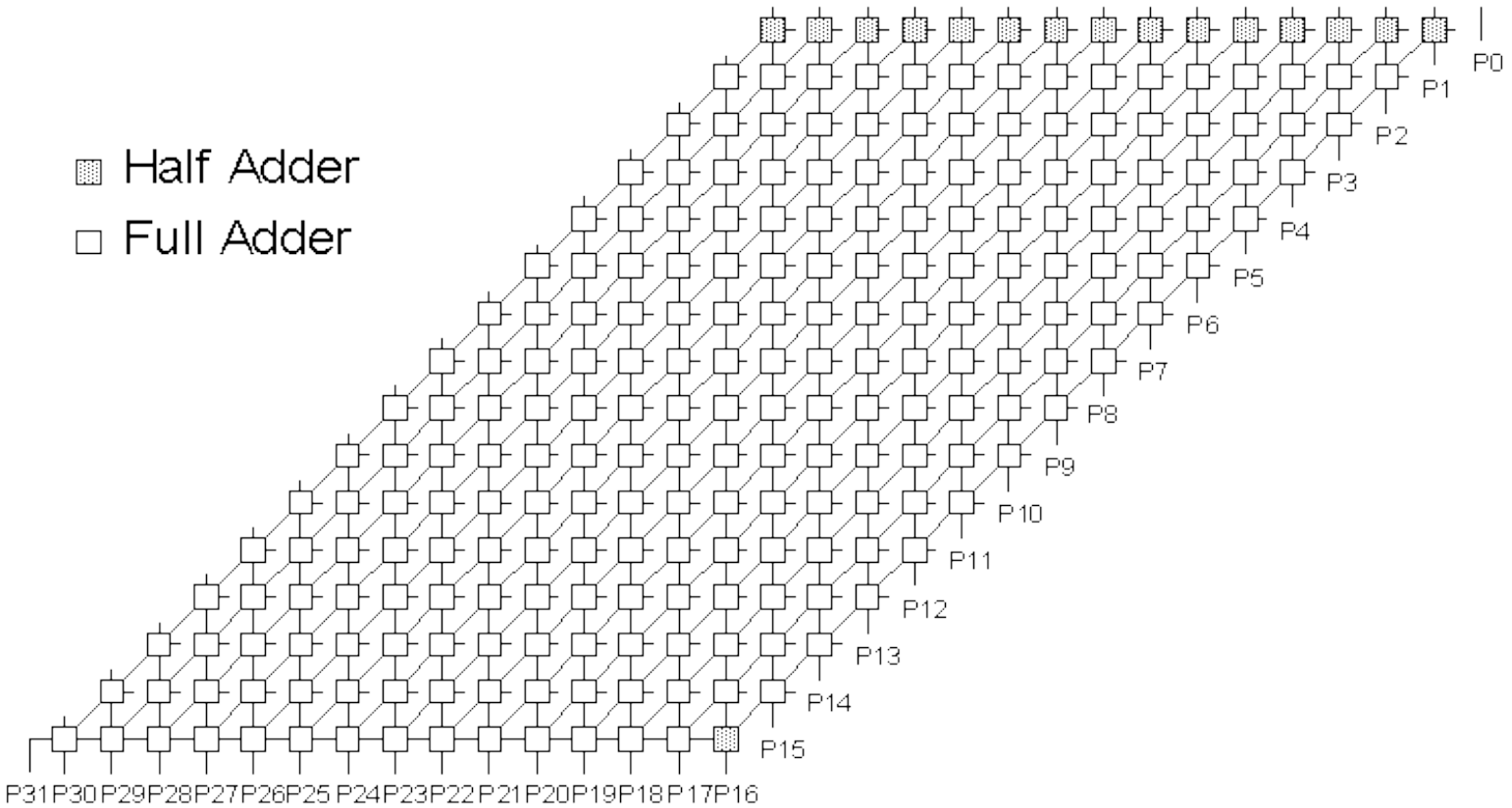
```
module M([...]);
  parameter real p=1;
  [..] pg = $param_given(p) [..]
endparamset
paramset M M1
  parameter real p=17; .p=p;
  [ pg = true (constant), p=17 ]
endparamset
paramset M M1
  // parameter real p=1; .p=p; // removed
  [ pg = false (constant), p=1 ]
endparamset
paramset M M // re-use name, just specialise, eg. for q.
  parameter real p; // no default.
  [ pg depends on input ] // hah. fixed.
  parameter real q=1 from [1:1]; // .. makes sense now
endparamset
```

Benchmark I: revisit ISCAS85

- ISCAS85 c6288: 16 bit multiplier netlist
- Originally targetting discrete simulators
- Analog version: add semiconductor models
- modelgen @FSiC'23: some dc analysis and traces

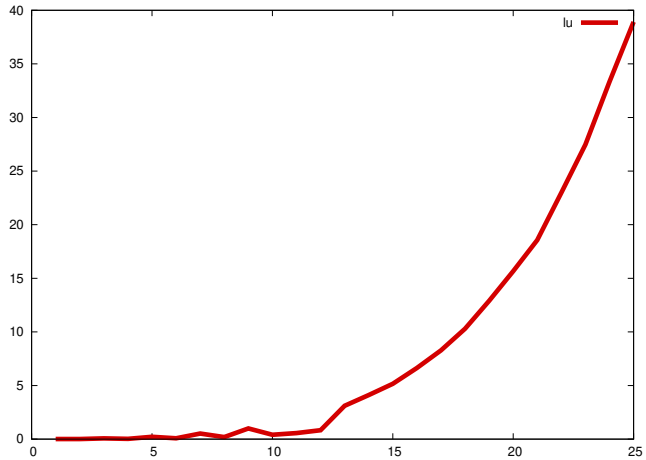
Benchmark I: revisit ISCAS85

- ISCAS85 c6288: 16 bit multiplier netlist
- Originally targetting discrete simulators
- Analog version: add semiconductor models
- modelgen @FSiC'23: some dc analysis and traces



Benchmark II: run time behaviour

- Quick check: restrict to dc/op
- Previous talk: 16x16 bit: 12 seconds with KLU & OpenVAF
- Now sweep multiplier width 1..25
- measure model evaluation time and LU time
- use proof-of-concept matrix interface
  (with experimental sparse solver to save a few cycles)



Benchmark III: roadmap

- dc time quadratic in number of nodes, (as expected)
- 150s wall time for 16 bit multiplier dc, 284 iterations
- Eval time improved since '22 (e.g. thanks to paramset)
- LU time (7 seconds), same ballpark as SPICE / VACASK
- (Pending optimisations in modelgen code.)

Next steps due '24: selective trace

- Aiming at linear time dcop (= "much faster")
- Expecting similar speedup in transient
- Finally outperform SPICE algorithm

User contributions

- QUCS Qt5 port (Matthias K.)
- Vector fitting algorithm (Seán H.)
- Convolution based analog filter (Seán H.)

Very useful. Big Thanks!

Contribution: Vector Fitting, Convolution

VF: An approximation scheme

- Input: map frequencies to complex values
  aka. Y, Z or S-parameters
- Output: Some kind of Padé approximation/interpolation

Contribution: Vector Fitting, Convolution

VF: An approximation scheme

- Input: map frequencies to complex values
  aka. Y, Z or S-parameters
- Output: Some kind of Padé approximation/interpolation

Why?

- Transmission line, RF, modelling
- Standardised in Verilog-AMS: laplace_*
  Takes rational function coefficients
- ... and supports transient simulation (e.g. for SI),
  superior to HB in some applications

Contribution: Vector Fitting, Convolution

VF: An approximation scheme

- Input: map frequencies to complex values
  aka. Y, Z or S-parameters
- Output: Some kind of Padé approximation/interpolation

Why?

- Transmission line, RF, modelling
- Standardised in Verilog-AMS: laplace_*
  Takes rational function coefficients
- ... and supports transient simulation (e.g. for SI),
  superior to HB in some applications

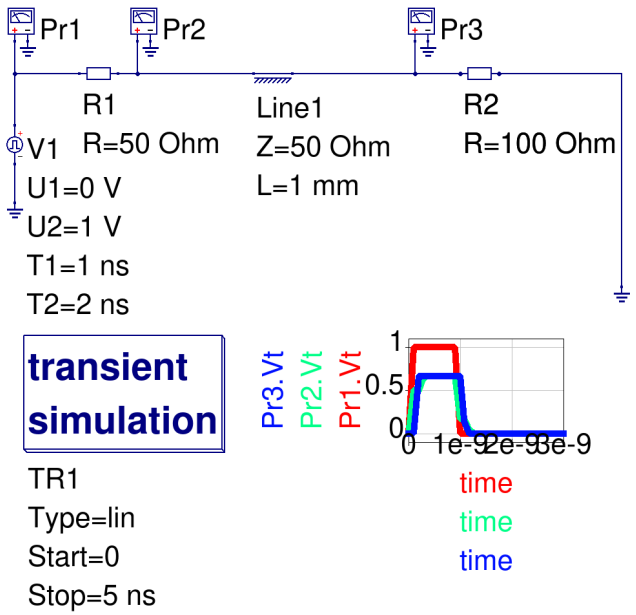
Convolution for filtering?

- Modelgen laplace_*: (ad-hoc?) subcircuit expansion.
- Convolution expected to be faster, where applicable.
- laplace_*d needs root finding first.
- They are all plugins anyway, use whatever works.

Contribution: QUCS Qt5 I

- Native Port, ditch Qt3/4 wrapper
- Compiles & runs on modern systems
- Still due for refactoring
- But some low hanging fruit available

Acts as a graphical UI for Gnucap (Gnucsator)



Contribution: QUCS Qt5 II

- More QUCS devices in Gnucsator
- Noise analysis now available
- TODO: Schematic file format update
- TODO: expose Verilog-AMS modelling

```
module tline0(out0, out1, in0, in1)
 [ see modelgen-verilog/examples for full source code ]
  parameter real delay = 1. from (0:inf);
  parameter real L = 1. from (0:inf);
  parameter real z0 = 50. from (0:inf);
  localparam real real_td = L*delay;
  analog begin
    vfwd = absdelay(2.*V(in) - V(ref), real_td);
    vref = absdelay(2.*V(out)- V(fwd), real_td);
    I(fwd) <+ V(fwd) - vfwd;
    I(ref) <+ V(ref) - vref;
    I(out) <+ V(out) - vfwd/z0;
    I(in) <+ V(in) - vref/z0;
  end
endmodule
```

Conclusion

- Verilog-AMS coverage growing
- Contributions are possible and trickling in
- Algorithms for VLSI now tangible

Possible projects (funding possible)

- Integration with other tools
- Advance related tools
- User extensions welcome (even osdi wrapper)

Research questions

- partial updates with KLU performance?
- Efficient PCB simulation / extraction?
- How about optimal device ordering?
- Verilog-AMS extensions: reliability?

Gnucap resources

- normally: one tag per month
  (to synchronise across repos)
- git repos on savannah + codeberg mirrors
- June '24: master release intended (let's see)
- The last one without the new solver(s)

Keep track:

- mailing list, gnucap-devel
- alternative (?): #gnucap:matrix.org
- recent wiki edits on www.gnucap.org