Verilog-AMS in Gnucap

Felix Salfelder

FSIC 26

Content

- Gnucap history and now
- Verilog-AMS, role of standards
- New features in Gnucap and Modelgen-Verilog
- Device models and PDK support
- Summary, Roadmap and Opportunities

Gnucap & Modelgen-Verilog

The GNU Circuit Analysis Package

- Small C++ library with (user) extensions
- "Everything is a plugin"
- Provide free/libre **mixed signal simulator**

Gnucap & Modelgen-Verilog

The GNU Circuit Analysis Package

- Small C++ library with (user) extensions
- "Everything is a plugin"
- Provide free/libre **mixed signal simulator**

Modelgen-Verilog

- "Model compiler" developped with Gnucap
- Process **behavioural models**
- Build device plugins..

Gnucap & Modelgen-Verilog

The GNU Circuit Analysis Package

- Small C++ library with (user) extensions
- "Everything is a plugin"
- Provide free/libre **mixed signal simulator**

Modelgen-Verilog

- "Model compiler" developped with Gnucap
- Process **behavioural models**
- Build device plugins..

Goals & Vision

- Enable collaborative development
- Embrace (open) standards

Gnucap past and present

Past

- 1973-1989. SPICE 1-3: final: '93 (for context)
- 1990. *ACS*, Al's Circuit Simulator
- 2000 Verilog-AMS, early traces
- 2001... ACS Renamed to *Gnucap*, a GNU project

Gnucap past and present

Past

- 1973-1989. SPICE 1-3: final: '93 (for context)
- 1990. *ACS*, Al's Circuit Simulator
- 2000 Verilog-AMS, early traces
- 2001... ACS Renamed to *Gnucap*, a GNU project

Present

- 2022 Gnucap support by NLnet (ongoing)
- 2023 Initial Modelgen-Verilog
- 2024 .. most of the analog covered
- 2025 Growing team, four active
- 2026 Initial IHP PDK in Verilog-AMS
  → don't miss Krzystofs talk.

Verilog-AMS purpose and meaning

- Extend Verilog-95, popular in digital domain
- Add analog and mixed behavioural modelling
- But keep a venerable Spice subsystem
- Focus on language features, not device models

Verilog-AMS purpose and meaning

- Extend Verilog-95, popular in digital domain
- Add analog and mixed behavioural modelling
- But keep a venerable Spice subsystem
- Focus on language features, not device models

Relevant high level aspects

- Based on expertly written standard
- I Data interchange
- II Advanced processing
- III Portable models

Verilog-AMS I: Data Interchange

- "Verilog-A" compact modelling
  enhance portable models including popularity
- Vendor independent netlists
  Get your circuit into another tool
  (without losing half of it).
- "Verilog-S": Schematics are circuits, not drawings.
  → my presentation at FSiC'25
- Combine compiler capabilities by outputting Verilog-AMS.
  → Lukas lighning talk, 11:20

Verilog-AMS II: Advanced processing

Move the complexities back to where they belong

- `paramset`: map instances to models
- Metadata annotations: `attributes`
- Avoid syntax creep and “dialects”
- Preprocessing

Coherent solution for

- topology, hierarchy
- net types, interconnect models
- (buses and arrays)
- (circuit generation)
- (system level modelling)

(Parenthesized: planned and worked on)

Verilog-AMS III: Portable models

Facts

- Verilog-95: Understood by a zoo of free tools
- Recent “Spice” compact models developped in Verilog
- Mixed VAMS models exist (commercial simulators)

Verilog-AMS III: Portable models

Facts

- Verilog-95: Understood by a zoo of free tools
- Recent “Spice” compact models developped in Verilog
- Mixed VAMS models exist (commercial simulators)
- Gnucap is catching up
- vendor-independent netlists made feasible

Verilog-AMS III: Portable models

Facts

- Verilog-95: Understood by a zoo of free tools
- Recent "Spice" compact models developped in Verilog
- Mixed VAMS models exist (commercial simulators)
- Gnucap is catching up
- vendor-independent netlists made feasible

Observations

- Lock-in plays against users
- **Portability** breaks lock-in and

Verilog-AMS III: Portable models

Facts

- Verilog-95: Understood by a zoo of free tools
- Recent "Spice" compact models developped in Verilog
- Mixed VAMS models exist (commercial simulators)
- Gnucap is catching up
- vendor-independent netlists made feasible

Observations

- Lock-in plays against users
- **Portability** breaks lock-in and
- free software benefits more than commercial
- Looks like worth the effort..

Verilog-AMS in Gnucap, feature overview

Existing, tested, stabilised

- Basic analog modelling ("Spice")
- Advanced analog models (events, state)
- Frequency domain models, filters
- Some digital: logic primitives, UDP..
- Hierarchy, paramset, binning

Verilog-AMS in Gnucap, feature overview

Existing, tested, stabilised

- Basic analog modelling ("Spice")
- Advanced analog models (events, state)
- Frequency domain models, filters
- Some digital: logic primitives, UDP..
- Hierarchy, paramset, binning

New features since FSiC'25

- Storage type and efficiency (→ FOSDEM'26)
- I Distribution functions and sampling
- II Continouous assignments
- III Discrete event handling
- IV Connect modules, connect rules

Gnucap I: Distribution Functions (LRM 9.13.2)

```
module proc
  localaram glob = $rdist_normal(.., "global");
  localaram inst = $rdist_normal(.., "instance");
endmodule
paramset resistor rr;
  [..]
  parameter string which="" from '{"mismatch"};
  .r = r * (proc.glob + proc.inst);
endparamset
```

Gnucap I: Distribution Functions (LRM 9.13.2)

```
module proc
  localaram glob = $rdist_normal(.., "global");
  localaram inst = $rdist_normal(.., "instance");
endmodule
paramset resistor rr;
  [..]
  parameter string which="" from '{"mismatch"};
  .r = r * (proc.glob + proc.inst);
endparamset
```

- `instance` and `global` variation
- Process parameters (hierarchical references)
- (efficient) Monte Carlo runs

Gnucap II: Continuous assignments

Combinatiorial circuits in terms of expressions

```
module latch(q, r, s)
  logic q, r, s;
  input r, s;
  output q;
  assign #17 q = r & ( ( q & r & s ) | !s );
endmodule
```

Gnucap II: Continuous assignments

Combinatiorial circuits in terms of expressions

```
module latch(q, r, s)
  logic q, r, s;
  input r, s;
  output q;
  assign #17 q = r & ( ( q & r & s ) | !s );
endmodule
```

Status

- Complement user defined primitives (tables)
- Still WIP on some rhs restrictions
- `reg` implementation pending

Gnucap III: Discrete Events

```
module counta(clk);
  electrical input clk;
  (*desc="edge count"*) integer count;
  analog @cross(V(clk), 1) count = count+1;
endmodule;

module countd(clk);
  logic input clk; // discrete domain
  (*desc="edge count"*) integer count;
  always @(posedge(clk)) count = count+1;
endmodule;
```

Gnucap III: Discrete Events

```
module counta(clk);
  electrical input clk;
  (*desc="edge count"*) integer count;
  analog @cross(V(clk), 1) count = count+1;
endmodule;

module countd(clk);
  logic input clk; // discrete domain
  (*desc="edge count"*) integer count;
  always @(posedge(clk)) count = count+1;
endmodule;
```

Why bother?

Gnucap III: Discrete Events

```
module counta(clk);
  electrical input clk;
  (*desc="edge count"*) integer count;
  analog @cross(V(clk), 1) count = count+1;
endmodule;

module countd(clk);
  logic input clk; // discrete domain
  (*desc="edge count"*) integer count;
  always @(posedge(clk)) count = count+1;
endmodule;
```

Why bother?

- Model behaviour and timing separately
- Accuracy/speed tradeoffs in (nearly) every device
- Scalable system level models

Gnucap IV: Connect Modules

Connect discrete and continous nets

```
module countd(clk)[..]  // discrete counter, above.
module tb();
  electrical clk;
  electrical ground gnd;
  vpulse #(args [..]) v1(aclk), gnd);
  my_ad_cm c1(.out(clk), .in(aclk));
  countd t(clk);
endmodule;
```

Gnucap IV: Connect Modules

Connect discrete and continous nets

```
module countd(clk)[..]  // discrete counter, above.
module tb();
  electrical clk;
  electrical ground gnd;
  vpulse #(args [..]) v1(aclk), gnd);
  my_ad_cm c1(.out(clk), .in(aclk));
  countd t(clk);
endmodule;
```

- Interconnect model goes here
- Accurate timing across domains
- Soon: user defined `connectmodule`
- Prospect: supplement for `IBIS` & al.

IV almost there: Automatic CM Placement

```
connectrules my_chip;
  connect my_d2a #(args..) input electrical output logic;
  [..]
endconnectrules;
module tb();
  electrical clk;
  electrical ground gnd;
  vpulse #(args [..]) v1(clk), gnd);
  // my_ad_cm .. (clk); // automatic, implicit
  countd t(clk);
endmodule;
```

- Clean separation circuit $\leftrightarrow$ device models
- Relies on types only, hence re-usable schematics
- Most convient in complex circuits

Gnucap work with device models

Realities

- Traditional libraries missing out on important concepts
- Spice is pretty good at digesting Spice
  each implementation in its own way
- Collaborators fancy new stuff

Gnucap work with device models

Realities

- Traditional libraries missing out on important concepts
- Spice is pretty good at digesting Spice
  each implementation in its own way
- Collaborators fancy new stuff

Recent Progress

- I "Spice subsystem" constantly improving
- II IHP OpenPDK in Gnucap Status & Teaser
- III Promise of device libraries in Verilog-AMS
- IV "Designers Guide" examples and Roadmap

Models I: Spice Subsystem in Gnucap

Quoting from Verilog-AMS LRM, Annex E

- ".. provide an appropriate degree of SPICE compatibility"
- "a great deal of incompatibility even among the SPICE languages themselves."

Models I: Spice Subsystem in Gnucap

Quoting from Verilog-AMS LRM, Annex E

- ".. provide an appropriate degree of SPICE compatibility"
- "a great deal of incompatibility even among the SPICE languages themselves."

Improvements in digesting Spice

- Parse performance bottleneck fixed
- Instanciation of `.model` declarations including binning
- Nested prototypes now feasible (need testing)
  Motivation actually from System-Verilog...

```
load bsim483.so // from models repo
.model nfet nmos  level = 14 wmin=17 ..

module ex1();
  nfet #(.w(1)) n1( ...);
endmodule;
```

Models II: IHP OpenPDK in Verilog-AMS

- Originally initiated by Pepijn, converted from Spice
- Lukas took over, de-obfscated Verilog-AMS port
- Rigourous numerical validation against legacy

Models II: IHP OpenPDK in Verilog-AMS

- Originally initiated by Pepijn, converted from Spice
- Lukas took over, de-obfscated Verilog-AMS port
- Rigourous numerical validation against legacy

Work in progress

- Analog models under review (→ Krzystofs talk)
- Supplementary mixed library planned
- Setting priorities in simulator development

Models III: PDKs in Verilog-AMS

Idea

- Use Verilog-AMS as primary source
- sub-select, extract or convert as needed
- Supply standardised test benches, characterisers

Models III: PDKs in Verilog-AMS

Idea

- Use Verilog-AMS as primary source
- sub-select, extract or convert as needed
- Supply standardised test benches, characterisers

But why Verilog-AMS?

Models III: PDKs in Verilog-AMS

Idea

- Use Verilog-AMS as primary source
- sub-select, extract or convert as needed
- Supply standardised test benches, characterisers

But why Verilog-AMS?

- Simulator independent, no other known standard
- Save maintenance time, spend on quality
- It runs faster than legacy ($\rightarrow$ Lukas talk)

Models III: PDKs in Verilog-AMS

Idea

- Use Verilog-AMS as primary source
- sub-select, extract or convert as needed
- Supply standardised test benches, characterisers

But why Verilog-AMS?

- Simulator independent, no other known standard
- Save maintenance time, spend on quality
- It runs faster than legacy ($\rightarrow$ Lukas talk)
- Proprietary simulators are there to stay

Models IV: Designers Guide Examples

What is it?

- Mixed signal modules in Verilog-AMS
- Showcase language expressivity
- Corner stone of VLSI chip design
- De-facto standard approach in advanced courses

Models IV: Designers Guide Examples

What is it?

- Mixed signal modules in Verilog-AMS
- Showcase language expressivity
- Corner stone of VLSI chip design
- De-facto standard approach in advanced courses

Status and coverage in Gnucap

- All examples technically supported (with quirks)
- Making progress on discrete domain
- Ongoing validation (→ Krzystofs Talk)

Summary, Roadmap and Opportunities

- Completed features according to plan
- New members boosting progress
- Slowly reducing the bus factor

Summary, Roadmap and Opportunities

- Completed features according to plan
- New members boosting progress
- Slowly reducing the bus factor

NLnet grants awarded .. until mid '27

- More of Verilog-AMS in the pipeline
- Support PDKs and standardisation
- Performance improvements

Summary, Roadmap and Opportunities

- Completed features according to plan
- New members boosting progress
- Slowly reducing the bus factor

NLnet grants awarded .. until mid '27

- More of Verilog-AMS in the pipeline
- Support PDKs and standardisation
- Performance improvements

Extensible funding for (new) contributors

- Verilog-AMS support in related tools
- Especially a schematic editor, Verilog-S
- Devices and wrappers. OpenVAF? SystemC?
- Anything really

Thanks!

Questions?