Verilog-AMS in Gnucap

Felix Salfelder, Al Davis

FOSDEM '25

Gnucap & Verilog-AMS

GNU Circuit Analysis Package

- 1990. *ACS*, Al's Circuit Simulator
- 2001. Renamed to *Gnucap*, a GNU project, GPLv3+
- 2008-2010. Move to Plugins
- Run on hardware too small to run Spice (originally)
- C++, plugins (devices, algorithms, languages...)
- Beyond Spice – mixed techniques for fast analog
- Modelgen: suitable model compiler

Gnucap ongoing work

- Revamp data structures & algorithms
- Pull back things that drifted apart
- Verilog-A: ready for system level analog
- Verilog-AMS: build bridge to Verilog-95
- Verilog-S (schematic exchange)
- Verilog-L, -PC (layout, PCB)
- SPICE subsystem remains accessible from Verilog-A
  (.. down to native C implementation)

Why Verilog, IEEE1364 & friends

- "Open Hardware" needs a source language
- Analog, digital modelling and in between
- Standards define terminology
- ... and best practice

Widely used in industry

- IEEE1364: 590 pages, 40 contributors, 12 EDA companies
- Verilog-AMS: 442 pages, 74 contributurs from 16 companies
- System-Verilog: 586 pages ...

Simulator overview/comparison

	ngSpice	VACASK	Gnucap
source lang	C	C++	C++
license	BSD	AGPL3+	GPLv3+
architecture	modular*	monolith†	modular
input lang	Spice dialect	Spectre dialect	any (plugins)
devices	built-in	OSDI only?	plugin
paramset	no	no	yes
spice devices	"current"	distilled	wrapper, any
verilog models	ADMS/OpenVAF	OpenVAF	mg-verilog
solver	SPARSE, KLU	KLU	(C)BS (any)
simulation	..	+ HB	+ mixed-TRAN
co-simulation	XSPICE, IVL	?	plugin

* compiled to monolith
† OSDI interface

Modelgen: The model compiler

- Generate C++ from model description
- .. to build a device plugin
- `modelgen`: predates Verilog. was analog only

modelgen-verilog, status

- Most of Verilog-A covered
- growing Verilog-AMS support, e.g.
  user defined primitives ("lookup tables")
- connectmodule generation: pending
  stopgap: examples from LRM 9.22, hard wired
- Would generate code for other simulators if needed
  (preferred: support the interface in your simulator)

Model compiler overview/comparison

	ADMS	OpenVAF	modelgen-verilog
source code	C, XML	Rust	C++
license	LGPL2.1	GPLv3+	GPLv3+
input	CMC subset	CMC subset	Verilog-A(MS)
output	C (any)	binary	C++ (any)
target interface	SPICE (any)	OSDI/SPICE	Gnucap (any)
filters	few	few	most
paramset	no	no	yes
hierarchy	no	no	yes
analog events	no	no	yes
state variables	?	bug	yes
switch branch	?	bug	yes
discrete models	no	no	yes
disciplines	no	no	wip
always block	no	no	wip
generate	no	no	wip

QUCS – Quite Universal Circuit Simulator

- Qt GUI + RF centered simulator "Qucsator"
- Initiated ~ 2000, GPLv2+
- Various GUI forks, Qucsator now archived (?)
- Gnucsator (ongoing): alternative simulator
- Resolved Qt3/4 stalemate, December '24.

Towards "Verilog-S". Thanks David @ LibreSilicon*

- was: `<Resistor R1 1 123 345 "1" "0" "3" "7">`
- now: `(* x_p=120 [..]symbol="Resistor" *)`
  `resistor #(.R(1), .TC0(3) ..) R1(.p(a), .n(b));`
- Break lock-in, streamline simulation etc..

Next steps

- Read Verilog-S into Qucs
- Translate gEDA/Lepton to Verilog-S
- Bump device library format

*https://nlnet.nl/project/LibreSilicon/

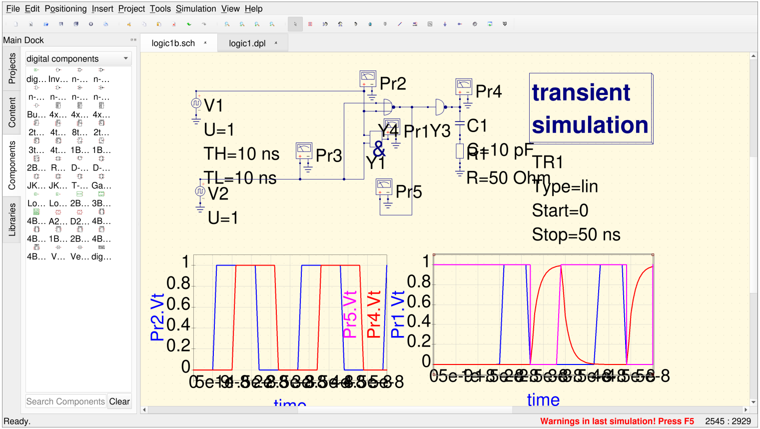
Gnucsator, Gnucap-based alternative for Qucs

- Provides Verilog-AMS "playground" with GUI
- portable, self-documenting device models underneath
- runs your own devices (still fiddly)
  restrictions by Qucsator "netlisting" & stuff.
- .. pending resolution: transition to Verilog-S
- (or just run Gnucap directly, if you can't wait)

Traditional Qucsator models now implemented in Verilog

- wrapped logic gate primitives (Verilog-95)
- wrapped Spice primitives (V-AMS, Annex E)
- `Vrect`, noise sources, `mux4to1`, `DCBlock` etc.

Qucs AMS smoke test



- Fixed disciplines (WIP), but true mixed-mode
- Have more complex circuits, but no drawings

Verilog circuit representation

```
module smoke_test ();
  ground gnd; // (not yet, really.)
  VProbe #() Pr1 (.p(_net0),.n(gnd));
  VProbe #() Pr2 (.p(_net1),.n(gnd));
  VProbe #() Pr3 (.p(_net2),.n(gnd));
  AND #(.V(1),.t(5)) Y1 (.y(_net0),.a(_net2),.b(_net1));
  NOR #(.t(19)) Y4 (.y(_net3),.a(_net1),.b(_net2));
  Vrect #(.TH(10n),.TL(10n),...) V1 (.p(_net1),.n(gnd));
  Vrect #(.TH(10n),.TL(10n),...) V2 (.p(_net2),.n(gnd));
  VProbe #() Pr4 (.p(_net4),.n(gnd));
  VProbe #() Pr5 (.p(_net3),.n(gnd));
  Inv #(.t(1)) Y3 (.y(_net4),.a(_net3));
  R #(.R(50),.Tc1(0.0),.Tc2(0.0),...) R1 (.p(gnd),.n(_net5));
  C #(.C(10p)) C1 (.p(_net5),.n(_net4));
endmodule // main
```

- Human readable schematic (here: netlist for brevity)
- Qucsator library maps to logic primitives

Help wanted

- Plugins
- Wrappers
- Device libraries
- Data exchange
- Test driving

Questions?