Verilog-AMS in Gnucap – Project Report

Felix Salfelder

FSIC 23

Content

- Gnucap, what is it?
- Verilog-AMS, why?
- Verilog-AMS, how?
- Examples
- Further roadmap
- Call for collaboration

Gnucap, overview

- 1990. *ACS*, Al's Circuit Simulator
- 1992. GPL'd
- 2001. Renamed to *Gnucap*, a GNU project
- Modular Modern Mixed architecture
- The original "fast Spice"
- Still, the only (free) one
- Plug-in interface

Today (thanks to NLnet):
Verilog-AMS model generator

What are plug-ins again?

- Dynamically loaded (`dlopen`) extensions
- Turing complete
- Linux `insmod`, Python `import`
- Decentralised development, no interference
- Stable public interface

What are plug-ins again?

- Dynamically loaded (`dlopen`) extensions
- Turing complete
- Linux `insmod`, Python `import`
- Decentralised development, no interference
- Stable public interface

Some existing examples

- Input languages: Verilog, SPICE, Spectre, gEDA, Qucsator
- Component models: primitives, modelgen, SPICE, Qucsator
- Commands: ac, dc, tran, fourier, pz, sparam, postprocessing..

What are plug-ins again?

- Dynamically loaded (`dlopen`) extensions
- Turing complete
- Linux `insmod`, Python `import`
- Decentralised development, no interference
- Stable public interface

Some existing examples

- Input languages: Verilog, SPICE, Spectre, gEDA, Qucsator
- Component models: primitives, modelgen, SPICE, Qucsator
- Commands: ac, dc, tran, fourier, pz, sparam, postprocessing..

Your project here.

Verilog-AMS

- Modelling Language
  - Based on Verilog, IEEE Std 1364-2005
  - Standard in semiconductor industry
  - Authored by former SPICE devs
  - Multidomain, Mixed Signal
- Distinct subsets
  - Behavioural, implement components
  - Structural, build networks
  - Compatibility with SPICE (pick one).

Why Verilog-AMS?

Pull back things that drifted apart

- Share structure with existing Verilog
- Portable compact modelling

Why Verilog-AMS?

Pull back things that drifted apart

- Share structure with existing Verilog
- Portable compact modelling
- Input language for AMS simulator
- Supersede SPICE & its variants
- Replace XSPICE and similar extensions
- Cover IBIS, Touchstone etc.

Why Verilog-AMS?

Pull back things that drifted apart

- Share structure with existing Verilog
- Portable compact modelling
- Input language for AMS simulator
- Supersede SPICE & its variants
- Replace XSPICE and similar extensions
- Cover IBIS, Touchstone etc.
- Unify Schematic & Layout
- Foster data exchange

Why Verilog-AMS?

Pull back things that drifted apart

- Share structure with existing Verilog
- Portable compact modelling
- Input language for AMS simulator
- Supersede SPICE & its variants
- Replace XSPICE and similar extensions
- Cover IBIS, Touchstone etc.
- Unify Schematic & Layout
- Foster data exchange
- The opposite of xkcd #927

About Analog and Mixed Signal

Modelling paradigms

- Continuous disciplines, conservative
- Discrete disciplines, signal flow
- Conversion between the two, `connectmodule`

About Analog and Mixed Signal

Modelling paradigms

- Continuous disciplines, conservative
- Discrete disciplines, signal flow
- Conversion between the two, `connectmodule`

Simulation requirements and challenges

- Analog "SPICE accurate" solver
- speed vs. accuracy tradeoff
- Real event queue, selective trace

Verilog-AMS in Gnucap

- Structural subset: Language plugin for Gnucap
  - Build & instanciate subcircuit modules
  - Parameters with ranges, names ports
  - Module overloading: consider all candidates
- Behavioural models: gnucap-modelgen-verilog
  - Handle procedural blocks. `analog`, `always`, `initial`
  - Emit component models (C++).
- SPICE compatibility
  - Plugin defined, whichever dialect
  - Top level circuit, macros
  - SPICEisms accessible through `.subckt`
  - Careful: SPICE may be case insensitive

Verilog-AMS module overloading

```
paramset new_component existing_component
 parameter type value = default [range];
[..]
.protoparm(value_expression);
[..]
endparamset
```

Verilog-AMS module overloading

```
paramset new_component existing_component
 parameter type value = default [range];
[..]
.protoparm(value_expression);
[..]
endparamset
```

- `paramset` replaces SPICE `.MODEL`
- Build `new_component` from `existing_component`
- Essentially singleton subcircuit

Verilog-AMS module overloading

```
paramset new_component existing_component
 parameter type value = default [range];
[..]
.protoparm(value_expression);
[..]
endparamset
```

- `paramset` replaces SPICE `.MODEL`
- Build `new_component` from `existing_component`
- Essentially singleton subcircuit
- User defined parameters with ranges
- Model selection and binning
- Recursion...

Replacing model cards

Current practice

```
.load ./bsim480.so ; unmodified spice model
.model nch nmos level=54 [..]
M1 d g s b nch w=1u l=1u
```

Replacing model cards

Current practice

```
.load ./bsim480.so ; unmodified spice model
.model nch nmos level=54 [..]
M1 d g s b nch w=1u l=1u
```

- What is `nmos`? What is `M`?
- Is `level` hardwired in `nch`?
- Model parameters vs. instance parameters

It depends (There is no standard.)

Replacing model cards

Current practice

```
.load ./bsim480.so ; unmodified spice model
.model nch nmos level=54 [..]
M1 d g s b nch w=1u l=1u
```

In Verilog-AMS

```
load_vams ./bsim480.va // ported from C
bsim480 #(.w(1u) .l(1u) ..  )  m1(d g s b);
```

Replacing model cards

Current practice

```
.load ./bsim480.so ; unmodified spice model
.model nch nmos level=54 [..]
M1 d g s b nch w=1u l=1u
```

In Verilog-AMS

```
load_vams ./bsim480.va // ported from C
bsim480 #(.w(1u) .l(1u) ..  )  m1(d g s b);
 paramset nch bsim4va
  parameter real l=0.1u, w=1u;
  parameter int level = 54 from [54:54];
  [..]                          // possible instance:
 .TYPE = 1 .L = l .W = w [..]
 endparamset

nch #(.w(1u) .l(1u) .level(54)) m2(d g s b);
```

Device selection

```
load_vams ./bsim480.va // ported from C
load_vams ./psp103.va // native PSP103VA module
```

Device selection

```
load_vams ./bsim480.va // ported from C
load_vams ./psp103.va // native PSP103VA module
 paramset nch bsim4va
  parameter real l=0.1u, w=1u;
  parameter int level = 45 from [54:54];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset
 paramset nch PSP103VA
  parameter real l=0.1u, w=1u;
  parameter level = 69 from [69:69];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset
```

Device selection

```
load_vams ./bsim480.va // ported from C
load_vams ./psp103.va // native PSP103VA module
 paramset nch bsim4va
  parameter real l=0.1u, w=1u;
  parameter int level = 45 from [54:54];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset
 paramset nch PSP103VA
  parameter real l=0.1u, w=1u;
  parameter level = 69 from [69:69];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset

nch #(.w(1u) .l(1u) .level(54)) m1(d g s b);
```

Device selection

```
load_vams ./bsim480.va // ported from C
load_vams ./psp103.va // native PSP103VA module
 paramset nch bsim4va
  parameter real l=0.1u, w=1u;
  parameter int level = 45 from [54:54];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset
 paramset nch PSP103VA
  parameter real l=0.1u, w=1u;
  parameter level = 69 from [69:69];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset

nch #(.w(1u) .l(1u) .level(69)) m1(d g s b);
```

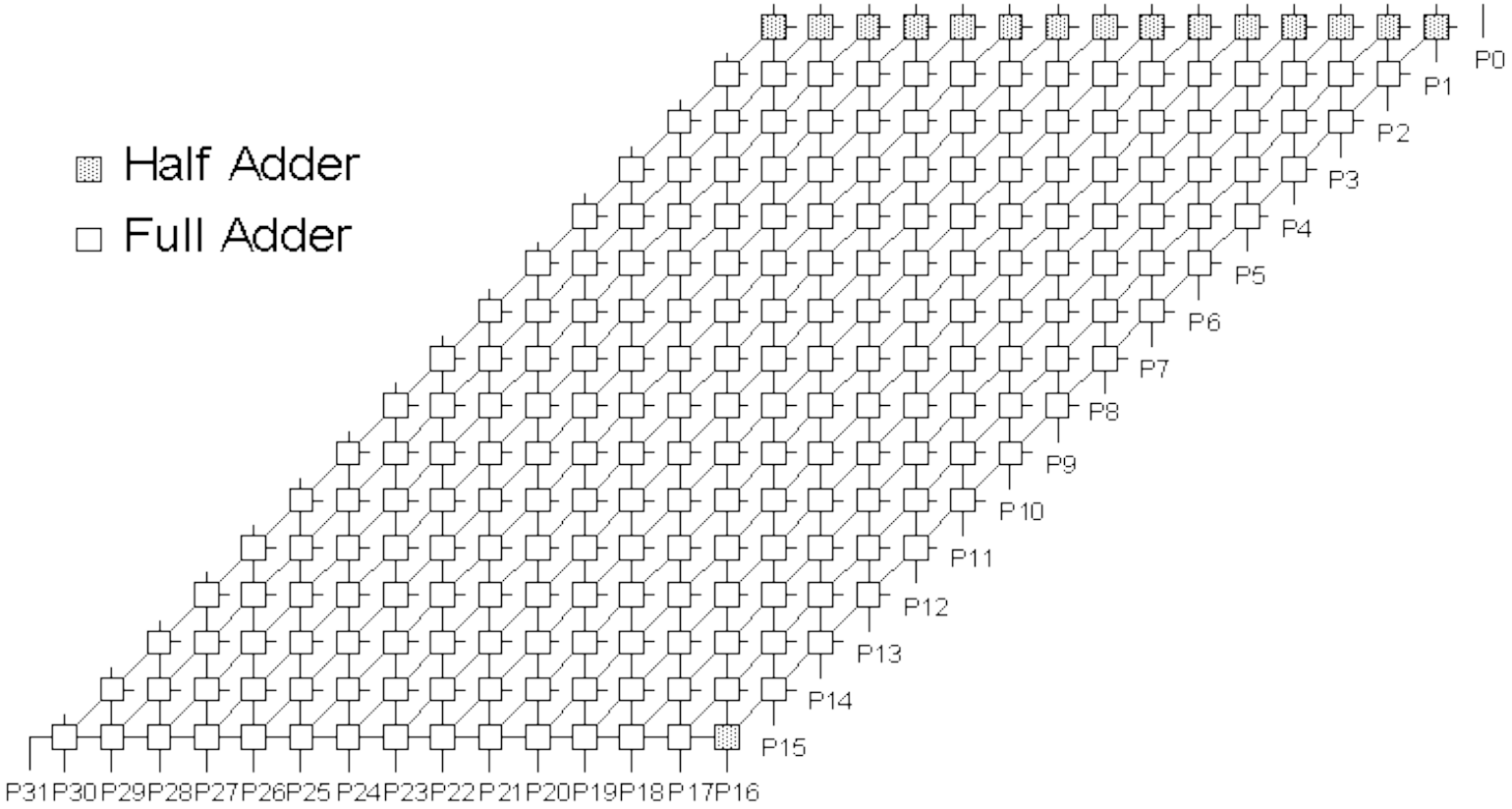
Device selection

```
load_vams ./bsim480.va // ported from C
load_vams ./psp103.va // native PSP103VA module
 paramset nch bsim4va
  parameter real l=0.1u, w=1u;
  parameter int level = 45 from [54:54];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset
 paramset nch PSP103VA
  parameter real l=0.1u, w=1u;
  parameter level = 69 from [69:69];
  [..]
 .TYPE = 1 .L = l .W = w [..]
 endparamset

nch #(.w(1u) .l(1u) .level(level)) m1(d g s b);
```

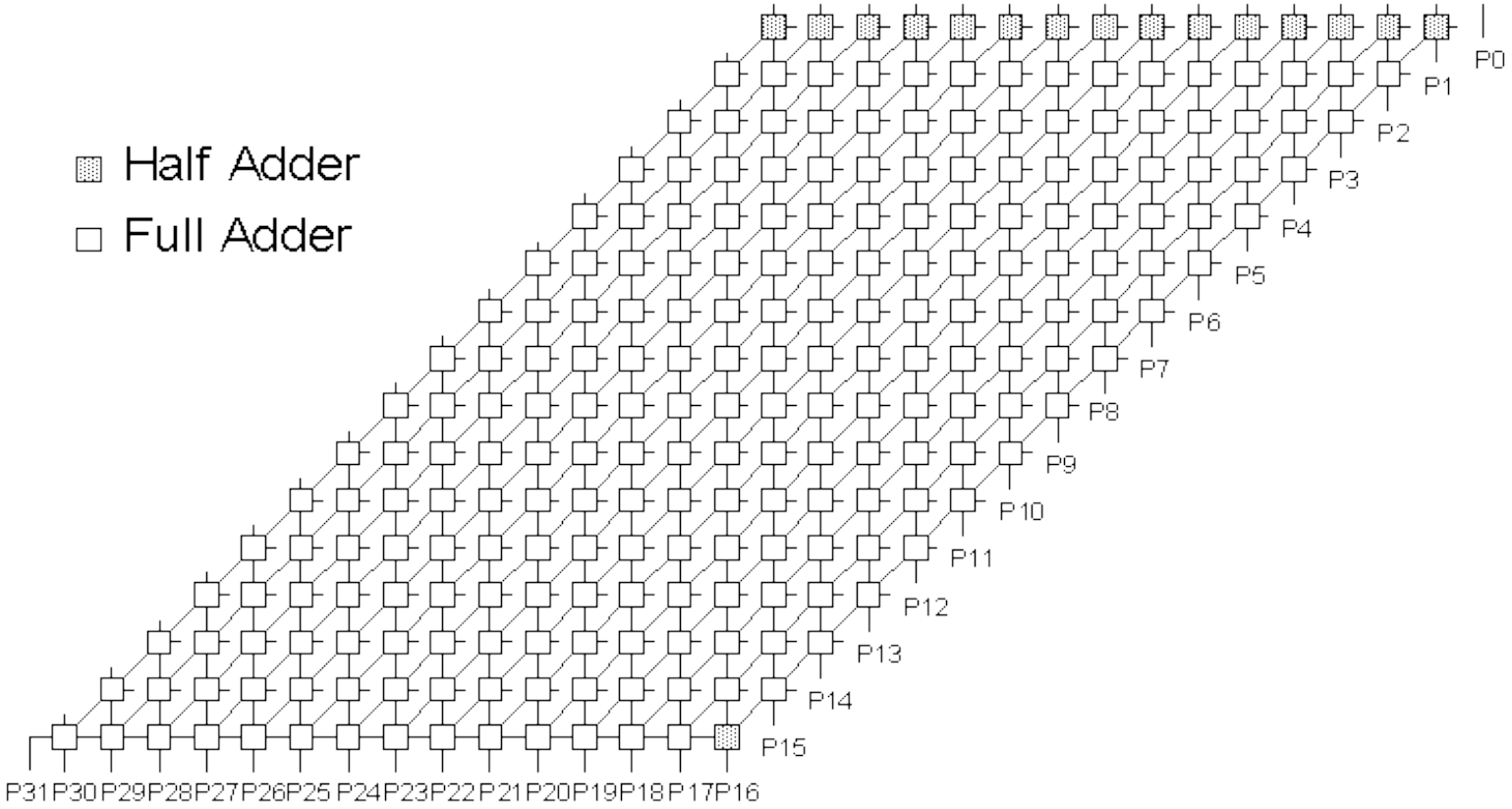
Example circuit

```
module nor2(z a b vdd vss)
 pch  #(.l(0.12u) .w(1.54u) .level(69)) m1(vdd a 01  vdd);
 nch  #(.l(0.12u) .w(0.44u) .level(69)) m2(vss a z   vss);
 pch  #(.l(0.12u) .w(1.54u) .level(69)) m3(01  b z   vdd);
 nch  #(.l(0.12u) .w(0.44u) .level(69)) m4(z   b vss vss);
endmodule

module FA(s_out, c_out, a, b, c, vdd, vss);
 nor2 g1(n1, a, b, 0, 0);
 nor2 g2(n2, n1, b, 0, 0);
 nor2 g3(n3, a, n1, 0, 0);
 nor2 g4(n4, n2, n3, 0, 0);
 nor2 g5(n5, c, n4, 0, 0);
 nor2 g6(n6, c, n5, 0, 0);
 nor2 g7(n7, n4, n5, 0, 0);
 nor2 g8(s_out, n6, n7, 0, 0);
 nor2 g9(c_out, n1, n5, 0, 0);
endmodule
```

Example circuit

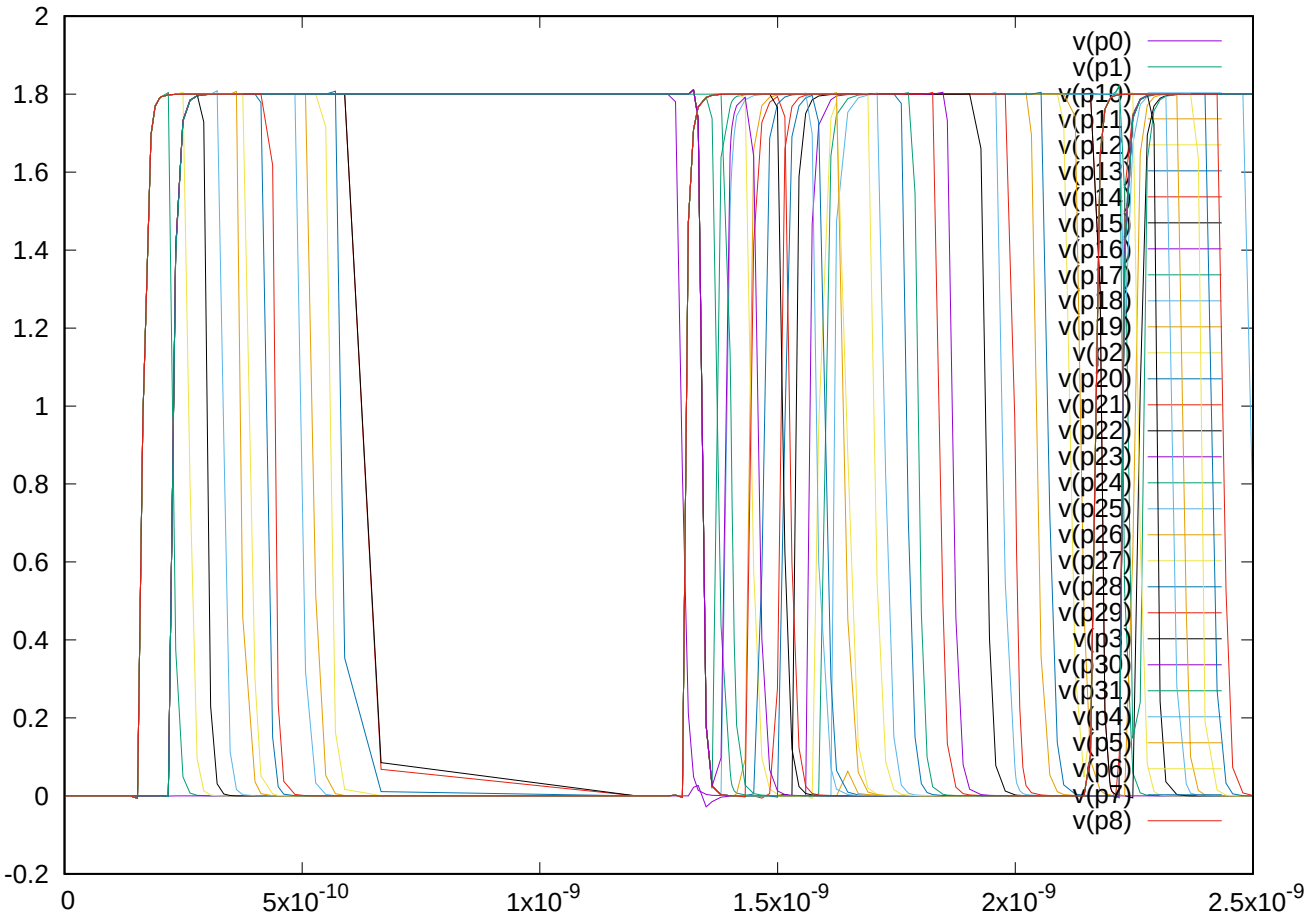


Example circuit

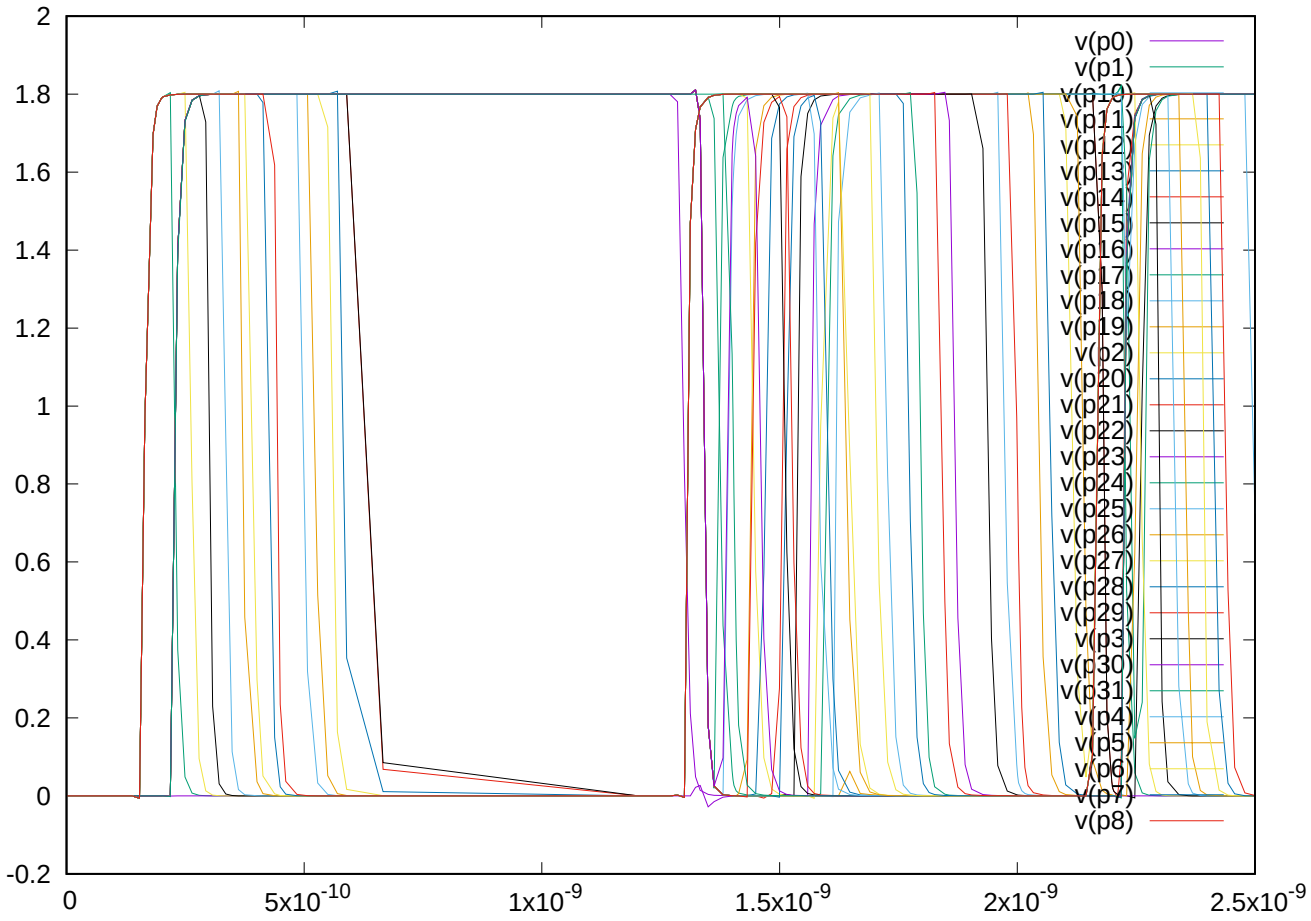


Essentially c6288 from ISCAS85, but analog.

Example Waveforms



Example Waveforms



- A bit slow still, 30 internal nodes per FET...
- Yes, DC analysis also working

Gnucap-modelgen-verilog usage and structure

Stages are

- Data structures. Defined in `mg_*.h`.
- A preprocessor, invoked with `-E` or `--pp`.
- Parse and dump, `mg_in*.cc`. Invoked with `--dump`.
- Gnucap plugin output, in `*_out*.cc`. Invoke with `--cc`.
- (compile plugins with `g++ -fPIC -shared`)

Gnucap-modelgen-verilog usage and structure

Stages are

- Data structures. Defined in `mg_*.h`.
- A preprocessor, invoked with `-E` or `--pp`.
- Parse and dump, `mg_in*.cc`. Invoked with `--dump`.
- Gnucap plugin output, in `*_out*.cc`. Invoke with `--cc`.
- (compile plugins with `g++ -fPIC -shared`)

The `*_out*.cc` are replaceable

- Target other simulators
- Emit byte code, not C++
- Endless options.

Gnucap-modelgen-verilog Status

Done (pending snapshot 202307**)

- Preprocessor
- Conservative disciplines, modules
- Branches, sources, analog blocks, derivatives
- Ready for bsim, psp etc.

Remaining goals '23:

- Modelgen optimisations
  constant propagation, node collapse, conditional sources
- simulator level improvements
  refactoring, plugin loading, build system
- Release

Further roadmap

Gnucap ground works

- Full event queue, selective trace, node ordering
- Harvest fast-Spice capabilities
- True AMS, multirate etc.

Language features in modelgen, planned for '24

- Signal-flow
- Cross events
- Connectmodule

Working towards new possibilities

- Post layout simulation
- timing analysis, signal integrity

Summary

- Gnucap is back thanks to NLnet
- Free/libre Verilog-AMS is on its way
- The Verilog-A functionality is close to release
- Next round: digital and mixed

Summary

- Gnucap is back thanks to NLnet
- Free/libre Verilog-AMS is on its way
- The Verilog-A functionality is close to release
- Next round: digital and mixed

Help needed: plugins

- Schematic/Layout interchange format
- Compatibility plugins, device wrappers
- Simulation commands, algorithms
- Your favourite language. VHDL-AMS?
- Revisit modular Qucs (or any GUI).

Thank You.