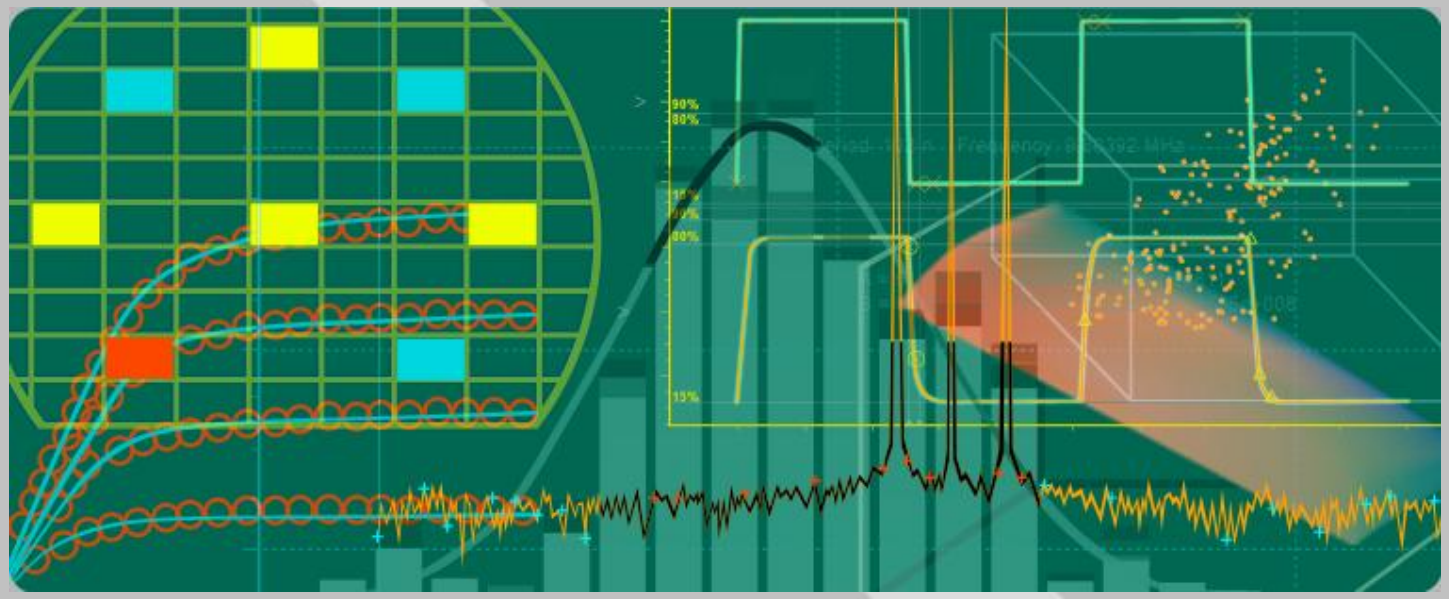
Convergence Aid Techniques for Compact Modelling with Verilog-A

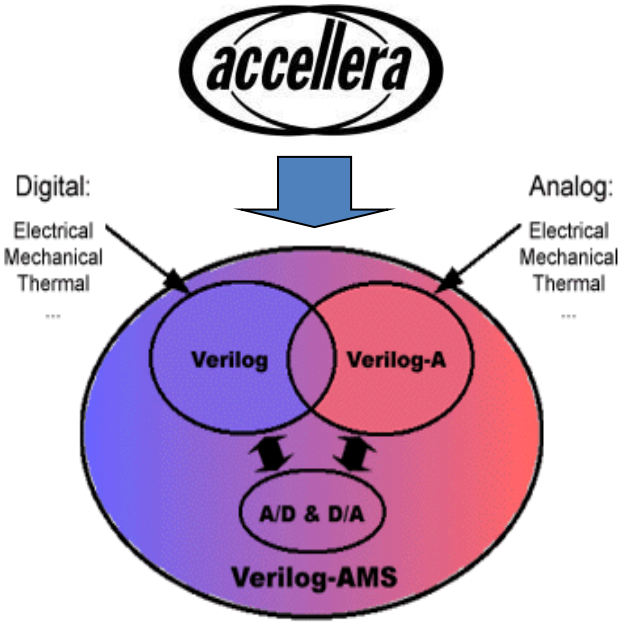


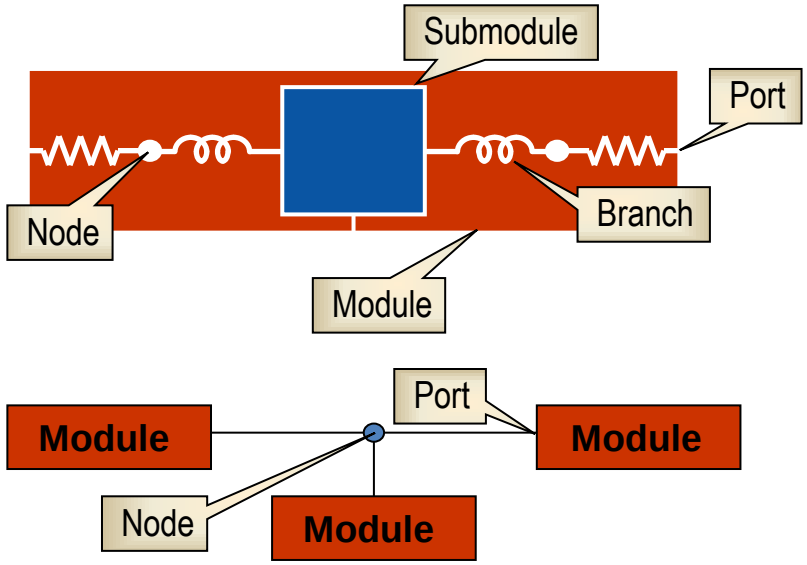
Slobodan Mijalković

Silvaco Europe, St Ives, Cambridge, UK

SILVACO

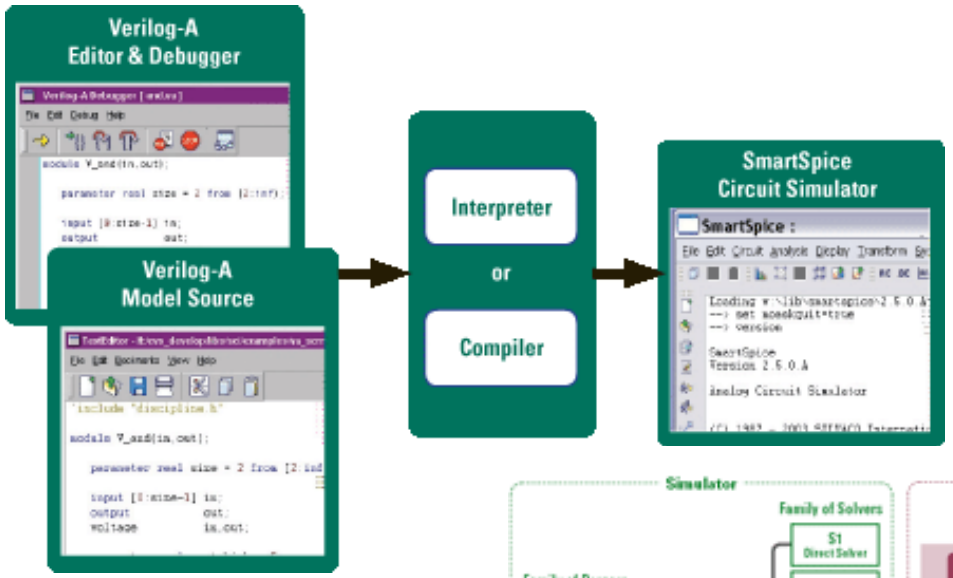
Verilog-A Language



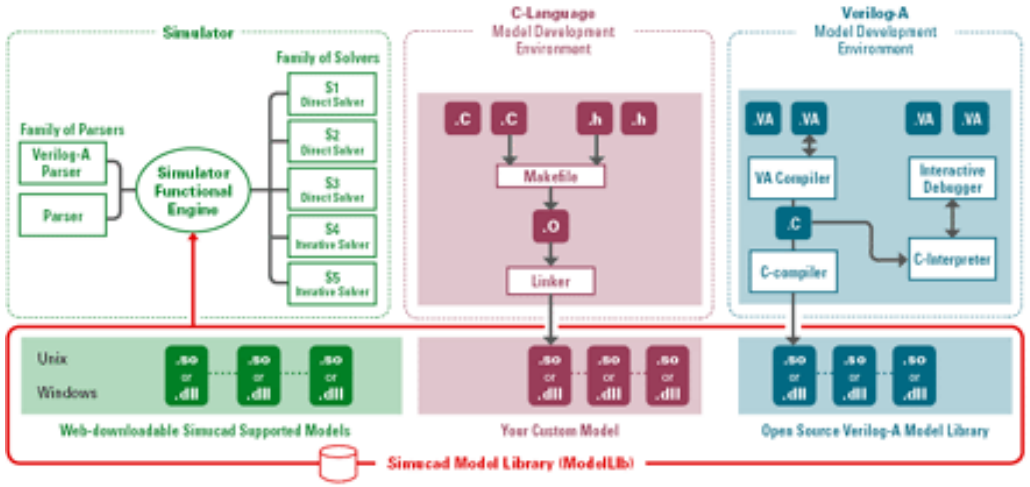


- Standard language
- Independent language entity ("highjacked" by Spice-like circuit simulators as device modeling language).
- Supports DC, AC, transient and noise analysis (and corresponding initialization steps)
- SPICE-like structural and hierarchical concepts
- Use of generalized Kirchhoff's KPL and KFL laws (converted to standard SPICE KVL and KCL laws)
- Enhanced for compact modeling (from LRM 2.2)

SmartSpice Verilog-A Compiler



- Integrated development and debugging environment accelerates compact model development
- Support for (partially or fully) encrypting of the Verilog-A source allows distribution of proprietary models without disclosure. Provides secure, transportable method for analog IP distribution and evaluation
- Compatible with all analog features of the Verilog-AMS 2.3 language specification
- SmartSpice Verilog-A executable models are currently within 2x runtime performance of the corresponding optimized C-compiled models



Compact Models (in Verilog-A source, binary or encrypted formats) are available at **`https://dynamic.silvaco.com/dynamicweb/silen/`**

SILVACO

Compact Modeling with Verilog-A

Benefits	Risks
Verilog-A provides concise and structured description of a compact model interface and architecture.	The compact model implementations are often incomplete (e.g. output variables, descriptions, units of parameters, parameter range, etc.)
Highly simplified model implementation (automatic code differentiation and load of branch matrix Jacobians).	Easy to neglect **numerical model requirements** and **implementation of the convergence aid techniques**.
Instant implementation of new compact models in circuit simulators.	The programming practices and performance of executable models varies with Verilog-A compilers.

Numerical Model Requirements

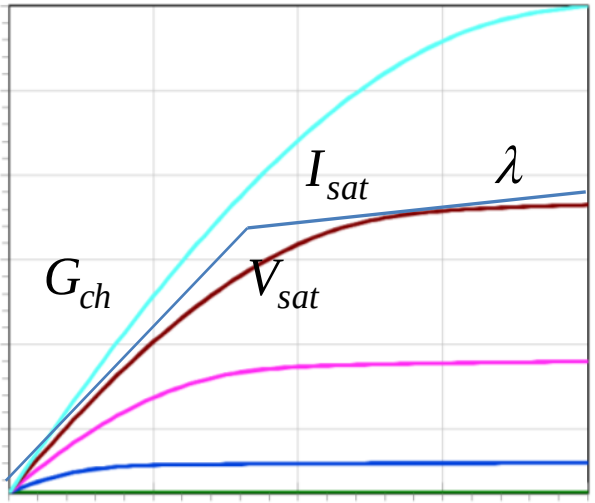
- **Asymptotic correctness**. The model evaluation code should be defined (free from floating point exceptions) for arbitrary bias and temperature conditions and for all allowed values of the model parameters
- **Smooth model behavior**. The branch nodal voltages and branch currents should be a C_∞ continuous function of input variables
- **Existing and unique solution of implicit model equations** (e.g thermal node temperature).

Eliminating Model Discontinuity

Nested Transformations (Silvaco UOTFT Model)

$$I_{ds} = G_{ch} V_{dse}(V_{ds})$$

$$V_{dse}(V_{ds}) = \frac{V_{ds}}{\left\{1 + \left[\frac{G_{ch} V_{ds}}{I_{sat}\left(1 + \lambda V_{ds}\right)}\right]^{m}\right\}^{1/m}}$$



$$I_{sat} = G_{ch} V_{sat}$$

$$V_{sat} = \frac{Q'_C}{C'_i(\gamma + 2)} + \frac{2\eta V_{to}}{\gamma + 1}$$

drift diffusion

SILVACO

Eliminating Model Discontinuity

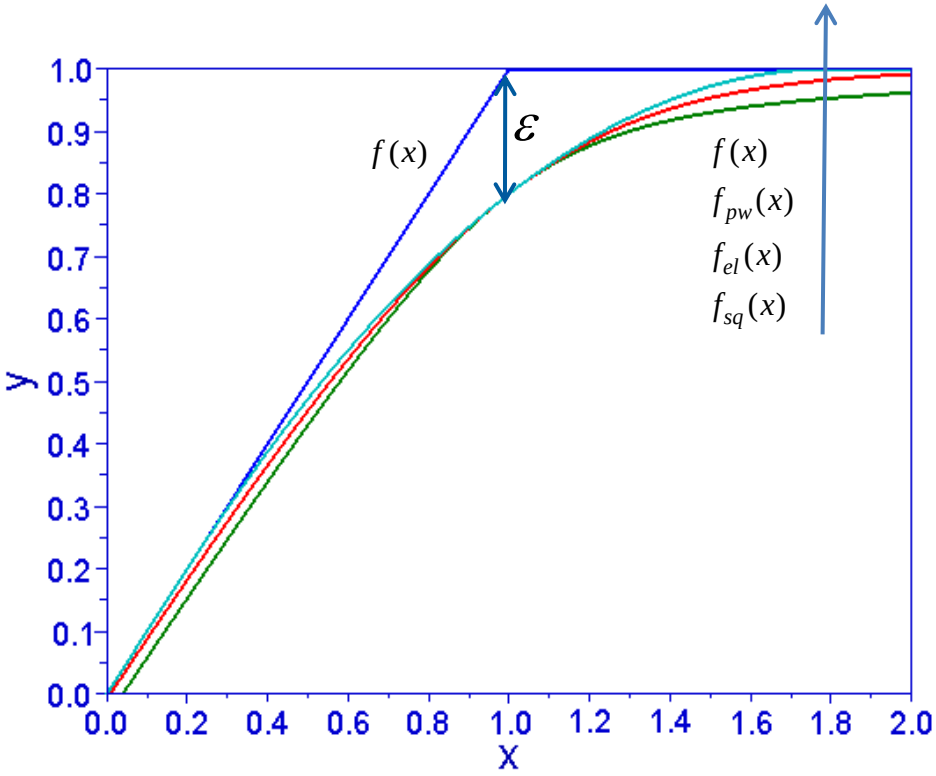
Limiting Functions

$$f(x)=\begin{cases} x \text{ for } x<1 \\ 1 \text{ for } x>1 \end{cases}$$

$$f_{sq}(x)=\frac{1}{2}\left(x+1-\sqrt{(x-1)^2+4\varepsilon^2}\right)$$

$$f_{el}(x)=x-\frac{\varepsilon}{\ln(2)}\ln\left(1+\exp\left(-\frac{(1-x)\ln(2)}{\varepsilon}\right)\right)$$

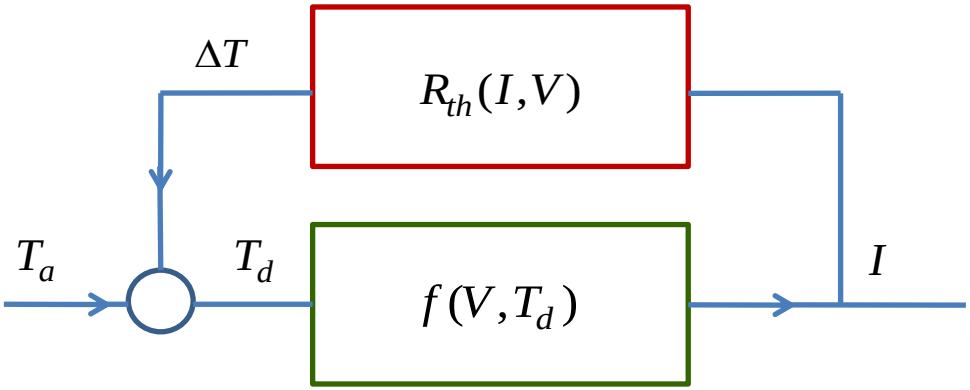
$$f_{pw}(x)=\begin{cases} x \text{ for } x\le 1-4\varepsilon \\ y-\dfrac{(x-1-4\varepsilon)^2}{16\varepsilon} \text{ for } 1\text{-}4\varepsilon < x < 1+4\varepsilon \\ 1 \text{ for } x\ge 1-4\varepsilon \end{cases}$$

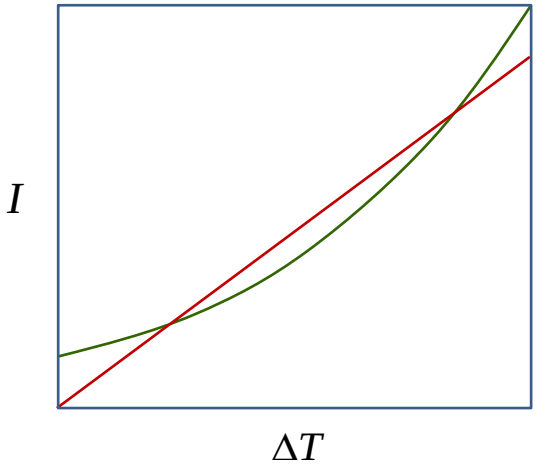


SILVACO

Existing and Unique Solution

Self-Heating Models





- Critical for SOI, TFT and any other MOSFET model with self-heating
- Requires an appropriate application of the limiting functions or modifications in the model formulation to prevent non-existing and close multiple solutions.

Non-Convergence and Possible Remedies

Reason for Non-Convergence	Convergence Aids	
	Solver Level	Model Level
Non-existing solution	Improving circuit design topology, model/instance parameters and numerical range solver parameters	Clipping of the model parameters Limiting variations of nodal voltages and branch currents.
Initial solution outside the convergence region	Setting initial solutions at the netlist level	**Setting model state initial solutions**
Multiple solutions	Homotopy	**Using solver homotopy parameters**
	Dumped iteration steps (modified Newton methods)	**Limiting internal model state iteration steps**
Floating-point exceptions		Limiting the arguments of root and pole functions
Floating-point precision problems (overflow, underflow, ill conditioning)		**Bounding branch conductivity** **The method of alternating bases**
Non-continuous model evaluation expressions		Eliminate model discontinuity

Setting Initial Solutions

Global initialization

```
electrical g = 1.0;
```

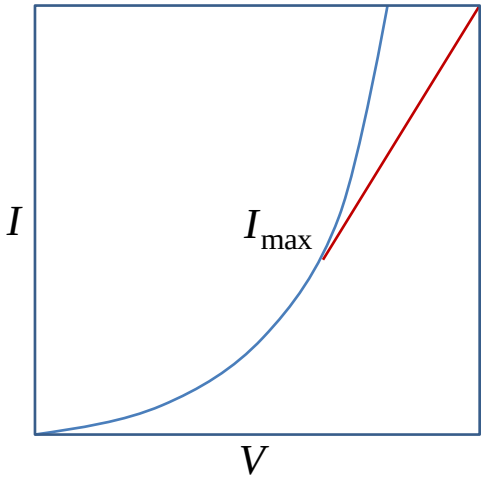
Analysis dependent initialization

```
if (analysis("ic"))
  V(cap) <+ initial_value;
else
  I(cap) <+ ddt(C*V(cap));
```

Analysis	Argument	DC	Sweep d1 d2 dN	TRAN op Tran	AC op AC	NOISE op AC
First part of "static" analysis	"nodeset"	1	1 0 0	1 0	1 0	1 0
Initial condition	"ic"	0	0 0 0	1 1	0 0	0 0

Bounding Branch Conductivity

- Adding small series resistance to nonlinear branch
- Adding small conductance in parallel with the nonlinear branch
- Linearized nonlinear branch above a critical current threshold




```
gmin = $simparam("gmin", 1e-13);
```

String	Unit	Description
gmin	1/Ω	Minimum conductance placed in parallel with nonlinear branches.
imax	A	Branch current threshold above which the constitutive relation of a nonlinear branch should be linearized.

Homotopy Methods

$$\mathbf{h}(\mathbf{x},\lambda) = \lambda \mathbf{f}(\mathbf{x}) + (1-\lambda)\mathbf{f_0}(\mathbf{x}) = 0$$

```
gdev = $simparam("gdev");
```

String	Unit	Description
gdev	1/Ω	Additional conductance to be added to nonlinear branches for *conductance homotopy convergence algorithm.*
sourceScaleFactor		Multiplicative factor for independent sources for *source stepping homotopy convergence algorithm.*

Limiting Newton Iteration Steps

```
analog begin
von = TYPE * VT0;
vcrit = `CONSTvt0 * ln(`CONSTvt0 / (`CONSTroot2*1.0e-14));

vgs = TYPE * $limit (V(gate,sourcep), fetlim, von);
vbs = TYPE * $limit (V(bulk,sourcep), pnjlim, `CONSTvt0, vcrit );
...
```

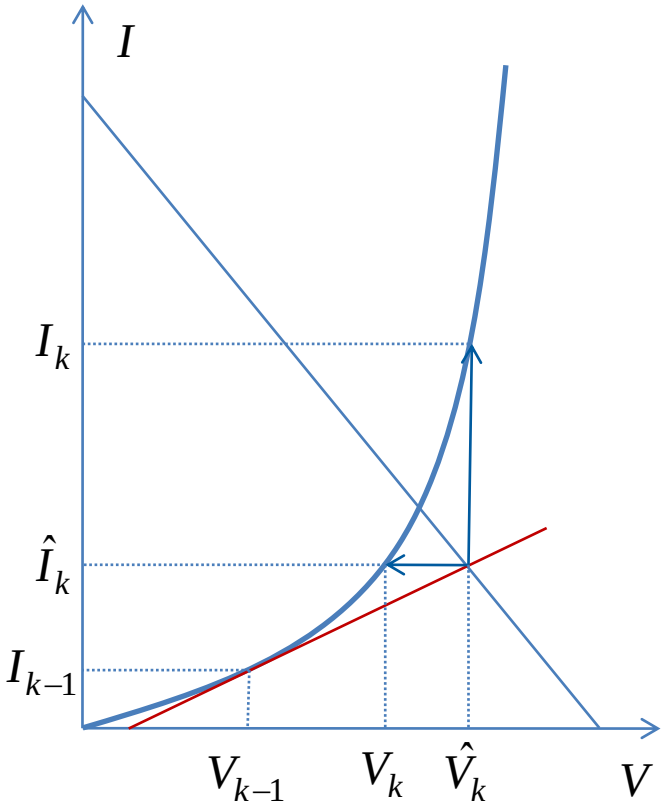
Function name	Arguments	Meant for limiting
fetlim	vth	gate-to-source voltage of field-effect transistors
pnjlim	vte, vcrit	voltage across diodes and p-n junctions in other devices
vdslim	(none)	drain-to-source voltage of field-effect transistors

Limiting Newton Iteration Steps

```
analog function real DEVlimvds;
  input VdsNew, VdsOld;
  real VdsNew, VdsOld;
  if (VdsOld >= 3.5)
    begin
      if (VdsNew > VdsOld)
        begin
          limitedValue = MIN(VdsNew,(3*VdsOld)+2);
  ...
  if (limitedValue != VdsNew) $discontinuity(-1);
  DEVlimvds = limitedValue;
endfunction

analog begin
vds = TYPE * $limit(V(d, s), DEVlimvds);
...
```

The Method of Alternating Bases




```
branch (p,n) dio
…
if (V(dio) < Vcrit ? 1 : 0)
  I(dio) <+ IS*(exp(V(p,n)/Vth)-1);
else
  V(p,n) <+ Vth*ln(I(p,n)/IS+1);
```


SILVACO

Summary

- The best set of convergence aid techniques is the numerically correct (asymptotically correct and C_∞ continuous) implementation of the physics based compact models
- Even numerically correct model implementation could require the setting of the initial solutions and bounding of the branch conductivities to improve the convergence properties in particular circuit designs.
- The homotopy methods of the solver should be employed also on the model side.
- Non-convergence is often caused by the incorrect circuit design as well as inappropriate solver and model parameters (it is not always a model fault)