Convergence Aid Techniques for Compact Modelling with Verilog-A

Slobodan Mijalković
Silvaco Europe, St Ives, Cambridge, UK
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/
## Compact Modeling with Verilog-A

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

\[ I_{ds} = G_{ch} V_{dse} (V_{ds}) \]

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

\[ I_{sat} = G_{ch} V_{sat} \]

\[ V_{sat} = \frac{Q'_C}{C'_i (\gamma + 2)} + \frac{2 \eta V_{io}}{\gamma + 1} \]

drift diffusion

MOS-AK 2010 Rome
Eliminating Model Discontinuity
Limiting Functions

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

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

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

\[ f_{pw}(x) = \begin{cases} 
  x & \text{for } x \leq 1 - 4\varepsilon \\
  y - \frac{(x - 1 - 4\varepsilon)^2}{16\varepsilon} & \text{for } 1 - 4\varepsilon < x < 1 + 4\varepsilon \\
  1 & \text{for } x \geq 1 - 4\varepsilon 
\end{cases} \]
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

<table>
<thead>
<tr>
<th>Reason for Non-Convergence</th>
<th>Convergence Aids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Solver Level</strong></td>
</tr>
<tr>
<td>Non-existing solution</td>
<td>Improving circuit design topology, model/instance parameters and numerical range solver parameters</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial solution outside the convergence region</td>
<td>Setting initial solutions at the netlist level</td>
</tr>
<tr>
<td>Multiple solutions</td>
<td>Homotopy</td>
</tr>
<tr>
<td></td>
<td>Dumped iteration steps (modified Newton methods)</td>
</tr>
<tr>
<td>Floating-point exceptions</td>
<td></td>
</tr>
<tr>
<td>Floating-point precision problems (overflow, underflow, ill conditioning)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-continuous model evaluation expressions</td>
<td></td>
</tr>
</tbody>
</table>
Global initialization

electrical g = 1.0;

Analysis dependent initialization

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

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Argument</th>
<th>DC</th>
<th>Sweep</th>
<th>TRAN</th>
<th>AC</th>
<th>NOISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>First part of &quot;static&quot; analysis</td>
<td>&quot;nodeset&quot;</td>
<td>1</td>
<td>1 0 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>Initial condition</td>
<td>&quot;ic&quot;</td>
<td>0</td>
<td>0 0 0</td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>
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

\[ g_{\text{min}} = \text{simparam}(\text{“gmin”}, 1e^{-13}); \]

<table>
<thead>
<tr>
<th>String</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gmin</td>
<td>(1/\Omega)</td>
<td>Minimum conductance placed in parallel with nonlinear branches.</td>
</tr>
<tr>
<td>imax</td>
<td>A</td>
<td>Branch current threshold above which the constitutive relation of a nonlinear branch should be linearized.</td>
</tr>
</tbody>
</table>
Homotopy Methods

\[ h(x, \lambda) = \lambda f(x) + (1 - \lambda)f_0(x) = 0 \]

\[ gdev = \texttt{simparam}(\text{"gdev"}); \]

<table>
<thead>
<tr>
<th>String</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdev</td>
<td>1/Ω</td>
<td>Additional conductance to be added to nonlinear branches for conductance homotopy convergence algorithm.</td>
</tr>
<tr>
<td>sourceScaleFactor</td>
<td></td>
<td>Multiplicative factor for independent sources for source stepping homotopy convergence algorithm.</td>
</tr>
</tbody>
</table>
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 );
...

<table>
<thead>
<tr>
<th>Function name</th>
<th>Arguments</th>
<th>Meant for limiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetlim</td>
<td>vth</td>
<td>gate-to-source voltage of field-effect transistors</td>
</tr>
<tr>
<td>pnjlim</td>
<td>vte, vcrit</td>
<td>voltage across diodes and p-n junctions in other devices</td>
</tr>
<tr>
<td>vdslim</td>
<td>(none)</td>
<td>drain-to-source voltage of field-effect transistors</td>
</tr>
</tbody>
</table>
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;
        end
    end
endfunction

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

branch \((p,n)\) dio

\[
\text{if (V(dio) < V_{crit} ? 1 : 0)}
\]

\[
I(dio) \leftarrow IS \times (\exp(V(p,n)/V_{th}) - 1);
\]

\[
\text{else}
\]

\[
V(p,n) \leftarrow V_{th} \times \ln(I(p,n)/IS + 1);
\]
Summary

- The best set of convergence aid techniques is the numerically correct (asymptotically correct and $C_\infty$ 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).