Standardization of Compact Device Modeling in High Level Description Language

L. Lemaitre*, C. McAndrew** and W. Grabinski*

*Motorola – Geneva – Switzerland
**Motorola, Tempe, AZ, USA
Contents

• Goals of the presentation
• What is Verilog-AMS?
• Verilog-AMS by Example
• Why a need for device modeling language?
• Extending Verilog-AMS for device modeling
• Model Compiler based on Verilog-AMS
• Performances
• Current Work and Contacts
Goal of the presentation

• Convince you that:
  – compact device modeling can be done based on the use of a model builder and based on the use of Verilog-AMS, a high-level description language
  – the benefits of having a model compiler (robustness, time to market, fast re-design) are higher than the loss in performance (speed)
What is Verilog-AMS?

- Verilog-AMS is a Hardware Description Language
- Verilog-AMS is mainly used as a behavioral language for analog circuit simulators
- Verilog-AMS gives analog designers a means to encapsulate behavioral description of analog systems into modules
- The language has been derived from a IEEE Verilog HDL specification.
- More info at http://www.accellera.org/
Verilog-AMS by Example

- show that syntax of Verilog-AMS is easy to understand
- show that only simple constructs of the language are necessary to start compact device modeling

1. `include disciplines.h
2. module MYRESISTOR (pnode, nnode);
3.   electrical pnode, nnode;
4.   parameter real W=1u, L=1u, GSH=1;
5.   real g;
6.   analog
7.     begin
8.       g = (W/L) * GSH;
9.       I (pnode, nnode) <+ g*V(pnode, nnode);
10.   end
11. endmodule
Need for Device Modeling Language?

- Why do we need a compact device modeling language?
- What kind of language do we need?

1. Write first draft of a model: build physics based constitutive equations
   if coding done in behavioral language then
   the model can be simulated as it were built-in into the simulator.
   Verilog-AMS is a good candidate. Most simulators support Verilog-AMS.
   Make easier the sharing of code between models.

2. Encode constitutive equations in computer language
   if the code of the model is done in high level description language then
   we can think of building a standard model compiler.
   The model compiler will compute the symbolic partial derivatives of the model.

3. Implement the code into electrical simulators
   Different flavor of c code has to be created from one simulator to another one.
   This can easily be handled by a model compiler.
   Coding will be perfectly synchronized between simulators.

4. Validate compact device model implementation
   Test-benches that validates the model can be specified in Verilog-AMS.
Extensions to Verilog-AMS Syntax

• We need to extend Verilog-AMS syntax in order to support compact device modeling
• Proposal: use the following construct
  (* property1 = value1, namespace:property2=value2 ... *)
• Example:
  parameter real g=1 (* info="conductance" spectre:type="inout" unit="S"); Parameter g is a conductance and it has specific property for Spectre.
• Advantages:
  minor impact on the Language Reference Manual
  offer large flexibility – each simulator can add custom information
  easy to implement into Verilog-AMS parsers
  partly integrated into Verilog-AMS
Model Compiler – I

ADMS – Automatic Device Model Synthesizer

Verilog-AMS Source Code

ADMS-XML Simulator-specific Interfaces

Testing prior implementation

C code Mica, Spectre, ADS, …

Documentation Circuit Test benches

ADMS Parser

XML Internal data

Code Generator

Other applications

ADMS Data Base

Other ADMS-XML Interfaces
Model Compiler - II

Run admsSpectre

BIPOLAR TRANSISTOR in VERILOG-AMS

SPECTREinterface.h
BIPdefs.h
BIPinitParameter.c
BIPloadJacobian.c
BIPevaluateStatic.c
BIPevaluateDynamic.c

ready-to-compile C code
Model Compiler – III

• CAD vendors can specify the way c code will be generated:
  – admsXml mymodule.va –e ADSinterface.xml
    • ready-to-compile c code for ADS
  – admsXml mymodule.va –e SPECTREinterface.xml
    • ready-to-compile c code for Spectre
  – admsXml mymodule.va –e SPICE3interface.xml
  – admsXml mymodule.va –e StandAloneinterface.xml
  – admsXml mymodule.va –e TestBenchinterface.xml

• Note that core c code of a device model is the same between simulators. Integrity of the core of the model is preserved.
• if one bug is found in one model then a fix will apply to all models created so far.
Performance – Speed

- We compared the performances between a built-in MOS model and the same model implemented into MICA by model compiler ADMS.

- **Penalty:**
  - after a first pass, the compiled MOS model was 2 time slower than the built-in model
  - after some iterations the model compiler was 20% slower than the built-in model:
    - optimization in partial derivatives
    - optimization of coding of the model at the Verilog-AMS level
  - ADMS-SSIM = 1.2X slower than BUILTIN-SSIM
Performance – Across Simulators

Spectre (ver. 4.4.3).
DC Analysis `opPoint'
Operating at T = 27 C.
V(Bint) = 650.428 mV
V(Cint) = 921.346 mV
V(Eint) = 79.0034 mV
I(vb:p) = -349.572 uA
I(vc:p) = -78.6538 mA
Power Dissipation = 79.0 mW

Ads(ver. "170")
© Agilent Technologies,
DC Operating Point:
V(Bint) = 650.428 mV
V(Cint) = 921.346 mV
V(Eint) = 79.0034 mV
vb.i = -349.572 uA
vc.i = -78.6538 mA
**Comparisons done by Cadence**

<table>
<thead>
<tr>
<th></th>
<th>spectre vbic</th>
<th>1.2.0 ADMS v0</th>
<th>1.2.0 ADMS v1</th>
<th>1.2.0 Modified</th>
<th>1.1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg vbicEval (cpu)</td>
<td>17,880</td>
<td>51,613</td>
<td>40,646</td>
<td>35,700</td>
<td>34,188</td>
</tr>
<tr>
<td>#calls of pow()</td>
<td>4,341,353</td>
<td>15,840,066</td>
<td>10,320,043</td>
<td>6,000,025</td>
<td>5,520,023</td>
</tr>
<tr>
<td>#calls of exp()</td>
<td>4,139,663</td>
<td>12,240,051</td>
<td>7,440,031</td>
<td>6,000,025</td>
<td>5,040,021</td>
</tr>
<tr>
<td>#calls of log()</td>
<td>3,409,682</td>
<td>5,760,024</td>
<td>4,320,018</td>
<td>4,320,018</td>
<td>4,080,017</td>
</tr>
<tr>
<td>#calls of sqrt()</td>
<td>1,767,640</td>
<td>8,640,036</td>
<td>1,920,008</td>
<td>1,920,008</td>
<td>1,440,006</td>
</tr>
<tr>
<td>#Tran steps</td>
<td>240,000</td>
<td>240,000</td>
<td>240,000</td>
<td>240,000</td>
<td>240,000</td>
</tr>
<tr>
<td>#Tran Iters</td>
<td>480,001</td>
<td>560,001</td>
<td>560,001</td>
<td>560,001</td>
<td>560,001</td>
</tr>
<tr>
<td>Tran CPU (sec)</td>
<td>100.4</td>
<td>123.9</td>
<td>123.6</td>
<td>118.3</td>
<td>112.2</td>
</tr>
</tbody>
</table>
Models implemented at Motorola

- Models available in Verilog-AMS:
  - SSIM (MOS)
  - EKV 2.6 (MOS)
  - SP (MOS, Prof. Gildenblatt)
  - MOSCAP (MOS varactors)
    - equations derived from SP core model
  - R3 (Three Terminal Resistor)
  - VBIC with self-heating (BJT)
  - GaAs based HBT
    - equations derived from VBIC core model
  - SOI MOS – Motorola China – J. Fossum and al. – U. of Florida

- All models are available in ADS (Agilent), Spectre (Cadence) and Mica (internal Motorola simulator)
module test_dc_instance_parameters();
  real wgiven, lgiven;
  // netlist
  myC #( .w (wgiven), .l (lgiven) ) m1 (1, 0);
  VoltageSource #( .dc (1.0) ) v1 (1, 0);
  for (wgiven=1u;wgiven<10u;wgiven=wgiven+1u) begin
    for (lgiven=1u;lgiven<10u;lgiven=lgiven+1u) begin
      @{
        "dc"
      } begin
        $assert(m1#w==wgiven);
        $assert(m1#l==lgiven);
      end
    end
  end
endmodule
**Device Model on Motorola Intranet**

**SP model for evaluation**

This repository contains basic data related to the implementation of SP (surface potential model) into SPICE simulators. SP is being developed by Prof. Gennady Gildeblat and Ten-Chon Chen. Contact: laurent.lemaitre@motorola.com


**NOTE**

- SP is under development
- Current MICA implementation NOT supported by MICA team

---

**Binaries**

*all simulators synchronized in one shot*

<table>
<thead>
<tr>
<th>Platform</th>
<th>MICA</th>
<th>SPECTRE</th>
<th>ADS</th>
</tr>
</thead>
<tbody>
<tr>
<td>hppl.1-hpplx10.20</td>
<td>SP.s1</td>
<td>on request</td>
<td>on request</td>
</tr>
<tr>
<td>hppl.2-hpplx10.20</td>
<td>SP.s1</td>
<td>on request</td>
<td>on request</td>
</tr>
<tr>
<td>hppp.0-hpplx11.00</td>
<td>SP.s1</td>
<td>SP.s0</td>
<td>on request</td>
</tr>
<tr>
<td>1666-pc-linux-gnu</td>
<td>SP.s0</td>
<td>on request</td>
<td>SP.s0</td>
</tr>
<tr>
<td>sparc-sun-solaris2.6</td>
<td>SP.s1</td>
<td>on request</td>
<td>SP.s0</td>
</tr>
<tr>
<td>sparc-sun-solaris2.8</td>
<td>SP.s0</td>
<td>on request</td>
<td>SP.s0</td>
</tr>
<tr>
<td>sparc-sun-solaris2.9</td>
<td>SP.s0</td>
<td>on request</td>
<td>SP.s0</td>
</tr>
</tbody>
</table>

**Source**

- Add directory MYDEV to ADS variable EESOF_MODEL_PATH. In one of:
  - /custom/config/hpessofilm.cfg
  - /hpessof/config/hpessofilm.cfg
- Note: EESOF_MODEL_PATH is *NOT* a shell variable
- Run hpessof your netlist as usual
- Note: examples of netlists can be found below
- More info: http://eesof.tm.agilent.com/docs

**Netlists**

*netlists ready for use for each simulator*

<table>
<thead>
<tr>
<th>Netlist</th>
<th>Description</th>
<th>MICA</th>
<th>SPECTRE</th>
<th>ADS</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP_dc</td>
<td>basic dc test</td>
<td>SP_dc.bst</td>
<td>SP_dc.cst</td>
<td>SP_dc.ckt</td>
<td><a href="mailto:laurent.lemaitre@motorola.com">laurent.lemaitre@motorola.com</a></td>
</tr>
<tr>
<td>r2rladder</td>
<td>R2R ladder</td>
<td>r2rladder.cst</td>
<td>on request</td>
<td>on request</td>
<td><a href="mailto:laurent.lemaitre@motorola.com">laurent.lemaitre@motorola.com</a></td>
</tr>
</tbody>
</table>

**Parameter Sets**

These data are made available only for testing the SP implementation. No update, no quality check have been performed.

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
<th>MICA</th>
<th>SPECTRE</th>
<th>ADS</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>hipe6wrf</td>
<td>extraction on progress</td>
<td>hipe6</td>
<td>hipe6</td>
<td>hipe6</td>
<td><a href="mailto:wgrabinski@motorola.com">wgrabinski@motorola.com</a></td>
</tr>
</tbody>
</table>
Contacts

Very Interested (want to have ADMS-XML-interfaces)
- Motorola - Mica
- Cadence - Spectre
- Agilent - ads
- Nassda - hsim

Interested (want to see)
- Xpedion - GoldenGate
- Mentor - Eldo
- Helsinki University of Technology - Aplac
- University of Washington: C. J. Richard Shi – spice3 interface

Compact Device Modeling
- EPFL – S3: EKV v3 model
- Penn State University: SP model
- Motorola-China - University of Florida: SOI MOS models
Conclusions

• Verilog-AMS language has been introduced
• Extensions to Verilog-AMS has been proposed to support compact device modeling
• A model compiler based on Verilog-AMS has been presented
• Advantages/Results of the technology have been stressed
demo

• installing adms
• running adms
• what is adms-xml?
• how works adms-xml?
• how is it easy to change adms-xml?
• a simple example
roadmap (next 3 months)

- promote adms
  - CAD vendors / device model community
  - get feedbacks from CAD vendors
- Most of the efforts on:
  - documenting the tool (how to install, how to run)
  - formalizing the different formats used by ADMS
- implement following simulator interfaces:
  - spice3
  - Spectre
  - Ads
  - Mica
- define DTD of adms-xml language
  - rules used by admsXml
- define DTD of internal adms data
  - internal data dumped out by running admsVerigola
- increase syntax supported by adms Verilog-AMS parser
- push Verilog-AMS committee to accept "compact device modeling" extensions

DTD: document type definition