ADMS – A Fully Customizable Compact Model Compiler

Laurent Lemaitre and Ben Gu
laurent.lemaitre@freescale.com
Freescale Semiconductor

INTODUCTION: Compact device models have been an important element of circuit simulations since the SPICE simulators were developed. Accuracy and performance of models to a large extent determines quality of simulations. Conventionally, compact models were plugged into a simulator through the application programming interface (API) of the simulator. Using this approach, model developers have to code their model equations using the data structure defined in the API and write several interface routines to communicate with the simulator. This approach has proved to be very inefficient. Coding the model equations and the calculate partial derivatives of charge/currents in the model into low level language can be very tedious and error prone, which makes implementing a model into a simulator often take months long. In addition, the approach is very in-flexible, the code developed for simulator A can't be used in simulator B without extensive changes, and any modification to device nodes, model parameters, and equations will have to result in a substantial changes to the code. Last a few years have witnessed a quick rise of Verilog-A language as a new standard for compact model development. Developing and delivering models in verilog-A format liberate model developers from the tedious labor of coding model equations in low level computer language and computing analytical partial derivatives, which significantly accelerates development process and improves quality of compact models. More importantly, models developed in Verilog-A format are ready to run on any circuit simulator as long as the simulator supports Verilog-A language. Because of these advantages, Verilog-A language has been adopted by several leading compact model developers, such as the Berkeley BSIM group, the EKV team, the HiSiM group and others.

Architecture of ADMS

First of all, ADMS is a fully customizable compact model compiler, a software which users can customize according to their own specifications. The software strategy employed by ADMS is somewhat similar to the compiler techniques used in the Bison program. As illustrated in Fig.1, The ADMS software has two major components: a Verilog-A parser and an ADMST interpreter.

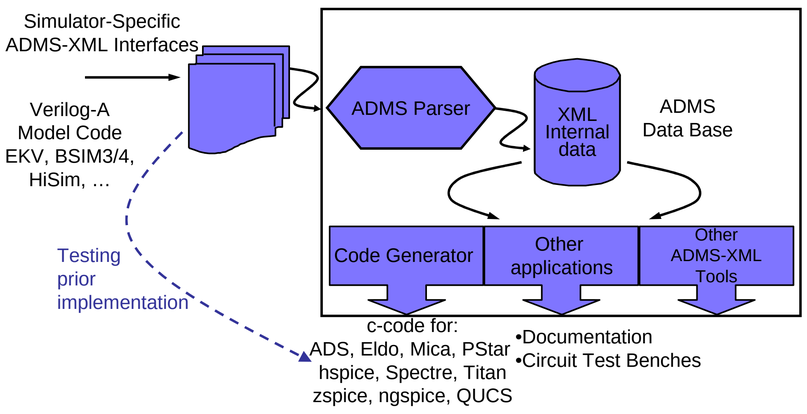

Fig. 1: ADMS data flow diagram

ADMS Data Tree

After the ADMS Verilog-A parser parses a Verilog-A source file it builds an ADMS internal data tree in computer memory. The XML tree is a different representation of model data stored in the Verilog-A file. Figure 2 gives a visual representation of the ADMS data tree that has been created for a simple module. For simplicity the picture has been simplified. The structure of the actual XML tree has a lot more branches and nodes.

```
//Verilog-AMS code
module myR (p,n);
   parameter real G=1;
   analog begin
     I(p,n)<+G*V(p,n);
   end
endmodule
```

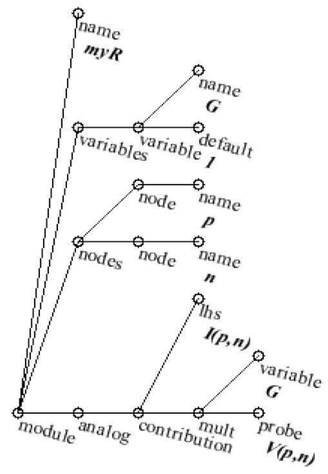

Fig. 2. ADMS Data Tree (Simplified View)

ADMST and ADMSTPATH

The language used to “describe” the behavior of the compiler is very similar to the XSLT and XPATH languages used in XML technologies. Missing features in the XSLT language for this particular application and the lack of available high-speed open-source XPATH interpreters required us to abandon a fully-XML based approach. We therefore defined an ADMS specific system. However, a significant level of backward compatibility has been maintained. At some time it will be still possible to move back to the XML system with minimal effort when it is extended to handle capabilities necessary for ADMS.

```
1. <admst:open file=”myfile”>
2. <admst:for-each select=”module”>
3. <admst:for-each select=”node”>
4. <admst:value-of select=”name”>
5. <admst:text format=”node is: %s\n”>
6. </admst:for-each>
7. </admst:for-each>
8. <admst:open>
```

Fig. 3. Simple script illustrating the ADMST syntax.

Optimizing Performance of Generated Code

The ADMST language provides a framework on which users can build their own model compilers. One of the most essential requirements for a model compiler is the capability of generating computationally efficient computer code to evaluate the model described in Verilog-A format. In this section, we review a few techniques users of ADMS can employ to enhance the computation efficiency of the generated code. These techniques has been implemented in ADMST language in the compact model compiler developed for the Freescale in-house SPICE simulator, Mica, and have been used to significantly improve the computational efficiency of a MOS model in Mica. Following a strategy judiciously promoted by the MOS-AK organization [1] the model has been developed and distributed in the Verilog-AMS format. Since speed is the main concern during the simulator implementation process special techniques have been developed in order to improve the model computational efficiency. The same techniques may be applied to well-accepted compact models like the BSIM family [5] or the EKV model [6].

Below is an equation in the section evaluating mobility reduction in the Verilog-A code of the MOS model:

```
Mutmp = pow(Eeffm * MUE_i, THEMU_i) + CS_i
      * (Pm / (Pm + Dm + 1.0e-14));
```

For this equation, an unoptimized compiler will generate C code as follows:

```
Mutmp_Vs_bp
= pow(Eeffm * MUE_i, THEMU_i) *
   MUE_i * Eeffm_Vs_bp / (Eeffm * MUE_i) +
   CS_i * (Pm_Vs_bp * (Pm + Dm + 1.0e-14) -
   Pm * (Pm_Vs_bp + Dm_Vs_bp)) /
   ((Pm+Dm + 1.0e-14) * (Pm + Dm + 1.0e-14));
Mutmp_Vd_s
= pow(Eeffm * MUE_i, THEMU_i) *
   MUE_i * Eeffm_Vd_s / (Eeffm * MUE_i) +
   CS_i * (Pm_Vd_s * (Pm + Dm + 1.0e-14) -
   Pm * (Pm_Vd_s + Dm_Vd_s)) /
   ((Pm + Dm + 1.0e-14) * (Pm + Dm + 1.0e-14));
Mutmp_Vgp_s
= pow(Eeffm * MUE_i, THEMU_i) *
   MUE_i * Eeffm_Vgp_s / Eeffm * MUE_i +
   CS_i * (Pm_Vgp_s * (Pm + Dm + 1.0e-14) -
   Pm * (Pm_Vgp_s + Dm_Vgp_s)) /
   ((Pm + Dm + 1.0e-14) * (Pm + Dm + 1.0e-14));
Mutmp = pow(Eeffm * MUE_i, THEMU_i)
           + CS_i * Pm / (Pm + Dm + 1.0e-14);
```

Having this technique implemented, an optimized compiler should generate code
as follows:

```
__pow_0 = pow(Eeffm * MUE_i, THEMU_i - 1.0);
__dF1_pow_0 = (THEMU_i) * __pow_0;
__pow_0 = (Eeffm * MUE_i) * __pow_0;
__dF1_div_1 = 1.0 / (Pm + Dm + 1.0e-14);
__div_1 = (Pm) * __dF1_div_1;
__dF2_div_1 = -__div_1 * __dF1_div_1;
Mutmp_Vs_bp
= (__dF1_pow_0 * MUE_i * Eeffm_Vs_bp) +
   CS_i * (__dF1_div_1 * Pm_Vs_bp
   + __dF2_div_1 * (Pm_Vs_bp + Dm_Vs_bp));
Mutmp_Vd_s
= (__dF1_pow_0 * MUE_i * Eeffm_Vd_s) +
   CS_i * (__dF1_div_1 * Pm_Vd_s
   + __dF2_div_1 * (Pm_Vd_s + Dm_Vd_s));
Mutmp_Vgp_s
= (__dF1_pow_0 * MUE_i * Eeffm_Vgp_s) +
   CS_i * (__dF1_div_1 * Pm_Vgp_s
   + __dF2_div_1 * (Pm_Vgp_s + Dm_Vgp_
```

The MOSFET Model Implementation

Figure 5 summarizes performance improvements of the MOS model achieved by cumulatively using the techniques we have reviewed in this section. The result presented is an average of 1e6 evaluations of the code generated by the compiler we have built for Freescale's in-house SPICE simulator, Mica, using ADMS. The model card used in the experiment is from one of bulk technologies developed by Freescale, and node voltages at each evaluation are randomly generated from 0 to 3V. As shown in Fig, 4, using these techniques collectively can lead to a spectacular result, speeding up the computational performance of the MOS model by nearly 2x.

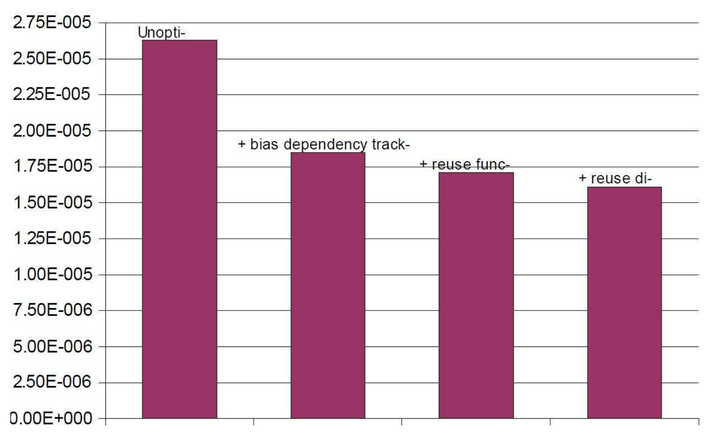

Fig. 4 Performance improvements of the MOS model under test

The Qucs XML interface

- The ADMS software translates Verilog-A device models into a structured XML tree
  Qucs/ADMS uses the XML tree to generate ready_to_compile C/C++ code
- The generated C/C++ code is specific to the Qucs API
- The process of transforming Verilog-A model code into C/C++ code is performed
  by the command line script:

$ admsXml <device.va> -e <interface-1.xml> -e <interface-2.xml>

Qucs release 0.0.11 is distributed with the following ADMS XML interface transformation scripts :

- qucsMODULEcore.xml : this creates the simulator C/C++ code for a Verilog-A device model
- qucsMODULEdefs.xml : this creates device parameter descriptions
- qucsMODULEgui.xml : this generates a model GUI interface
- qucsVersion.xml : this is a basic library
- analoguefunction.xml : this is used to create analogue function code

A detailed description of the steps needed to translate a Verilog-A model into C/C++ code and link it to the Qucs API are documented.

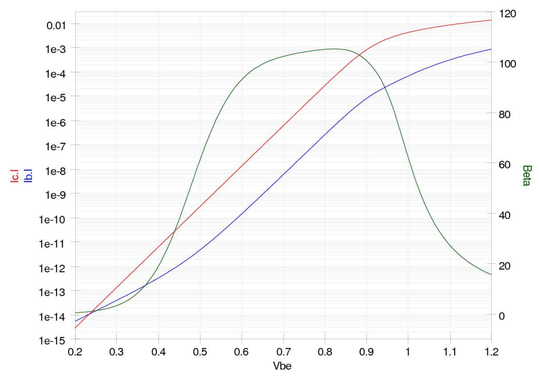

Fig. 5: Output plots for the Qucs implementation of the HiCUM/L2 V2.11 device model using ADSM

The NGSpice XML interface

- ADMS is not (yet) included into NGSpice and must be downloaded separately.
- Verilog-AMS devices are compiled statically into the simulator and code must be present at *configure* time.
- Adms templates and codemodels devices are grouped under a common directory.
- Adms templates are used to translate Verilog-AMS code and fill with the appropriate code NGSpice model structure.
- There exist a template file for each file to be created.
- Spice noise analysis is not (yet) supported.
- There exist a “special” template file needed to generate the Makefile.am needed by NGSpice to build C code from XML and this file is processed first.

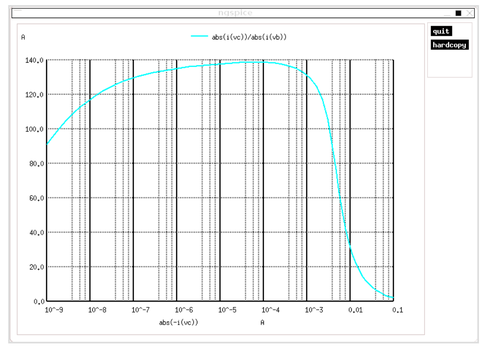

Fig. 6: Output plots for the NGSpice implementation of the HiCUM/L2 V2.22 device model using ADMS

Conclusions

This paper reviews architecture of ADMS, a fully customizable a compact compiler and syntax of ADMST language with which users define their specifications of device compilers. Several common optimization techniques to improve the computational efficiency of compilers built by ADMS are introduced by using a MOS model as an example.

Acknowledgments

Authors would like to thank every member of Mica group in Freescale for being wonderful to work with.

References

[1] http://www.mos-ak.org
[2] Verilog-AMS language reference manual http://www.eda-stds.org/verilog-ams/.
[3] L. Lemaitre et. al. IEEE CICC Proc. pp. 27-30, 2002
[4] http://sourceforge.ne t/projects/mot-adms/
[5] http://www-device.eecs.berkeley.edu/
[6] http://legwww.epfl.ch/ekv/
[7] S. Jahn, M. Brinson, M. Margraf, H. Parruitte, B. Ardouin, P. Nenzi and L. Lemaitre GNU Simulators Supporting Verilog-A Compact Model Standardization http://www.mos-ak.org/premstaetten/

Presented on April 3, 2009 at MOS-AK Meeting at IHP