ELECTRON TECHNOLOGY
ISSN 1897-2381
http://www.ite.waw.pl/etij/
Internet Journal

COMPACT DEVICE MODELING USING VERILOG-AMS AND ADMS

L. LEMAITRE[1], W. GRABIŃSKI[1], C. MCANDREW[2]

[1]Motorola, Geneva Modeling Center, 207 route de Ferney, CH-1218 Le Grand Saconnex, Switzerland.
[2]Motorola, Phoenix, USA

Received March 25, 2003; modified June 6, 2003; published July 4, 2003

ABSTRACT

This paper shows how high level language such as Verily-AMS can serve as support for compact modeling development of new devices. First section gives a full Verily-AMS code of a simplified bipolar transistor. Each part of the code is carefully examined and explained. Second section compares different implementations of the simplified bipolar transistor in different spice simulators. ADMS, an open-source tool developed at Motorola, performs the implementation from Verily-AMS to simulators. Third section concludes the paper by describing the implementation of the EKV model into ADS using the compact model interface provided by Agilent.

1. Introduction

Emerging markets of RF applications require better modeling of electrical effects of micro-semiconductor device and technologies.

As process fabrications of the semiconductor devices approach dimensions below 0.1-micron meter the electrical effects that were negligible in the past could not be ignored any more. These new effects need to be embedded into standard compact models such as varactors, MOSFETs, BJTs as well as passive RF devices. Moreover, new compact devices, which, potential has seldom been explored until recently, are coming to market.

On-wafer inductances are good examples of these new compact devices. Today a lot of effort is under way to better understand the physical behavior of on-wafer inductances in sub-microns semi-conductor process. Better tuning of standard compact devices or building new compact devices implies a big investment of time and effort. Furthermore, compact models should be implemented into different electrical circuit simulators. The implementation involves error-prone operations such as the calculation of symbolic derivatives of complex expression. These operations have to be repeated for each electrical circuit simulator.

Most of the tasks involved can easily be done by automated using dedicated software tools with standardized high-level behavioral modeling language.

This paper presents a tool called ADMS (Automatic Device Model Synthesizer) [1]. ADMS reduces the implementation efforts of compact device definition using Verilog-AMS model description [2]. At the same time, it offers a way to substantially improve the robustness of new compact device models. Implementation of the same model across different electrical circuit simulators is automated. Comparisons between the different implementations are straightforward. Fix of bugs found in one implementation can easily be propagated to other implementations.

2. HBT Model Description

This section presents a behavioral description of a simplified HBT model. The behavioral description of a compact model is coded in Verilog-AMS language. Verilog-AMS is a high-level behavioral description language for analog circuits. The language is intuitive and easy to understand. Its syntax is close to c language syntax. Figure 1 shows the branch assignments in the HBT model. Figure 2 gives the full Verilog-AMS code that describes the HBT model.

Let us go through the code and give some details of the syntax used.

The code is divided into three main sections:
1) the header section,
2) the declaration section,
3) the constitutive equations section.

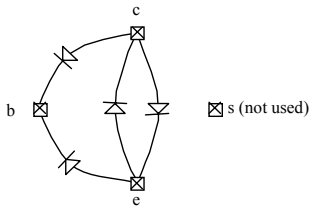

Fig. 1. Branch assignment in the HBT model.

```
`define NPN -1
`define PNP +1
module HBT(c,b,e,s);
// Nodes
   input        c,b; // input nodes
   output       e, s;// output nodes
   electrical   c,b,e,s; // all electrical
// Branches
   branch (b,c)           bc;
   branch (c,e)           ce;
   branch (e,c)           ec;
   branch (b,e)           be;
// Parameters
   parameter   real           is      = 20e-12;
   parameter   real           bf      = 225;
   parameter   real           br      = 5;
   parameter   real           nf      = 1.0;
   parameter   real           nr      = 1.0;
   parameter   integertype      = `NPN;
// Variables
   real Tdev, Vtv;
   real Ifi, Ibf;
   real Iri, Ibr;
   real arge,expe;
   real argc,expc;
//      Analog section
   analog begin
     Tdev  =  $temperature;
     Vtv   =  1.380662e-23*Tdev/1.602189e-19;
     if ( type == `NPN ) begin
        arge = V(be)/(nf*Vtv);
        argc = V(bc)/(nr*Vtv);
     end else if ( type == `PNP ) begin
        arge = -V(be)/(nf*Vtv);
        argc = -V(bc)/(nr*Vtv);
     end
       expe = exp(arge);
       expc = exp(argc);
       Iri = is*(expc-1.0);
       Ibr = Iri/br;
       Ifi  = is *(expe-1.0);
       Ibf  = Ifi/bf;
   begin
        I(ec) <+-Iri;
        I(ce) <+-Ifi;
        I(be) <+-Ibf;
        I(bc) <+-Ibr;
     end
   end
endmodule
```

Fig. 2. Verilog-AMS definition of a HBT device.

In the header, section macros can be defined. The use of macros helps improve the clarity of Verilog-AMS coding. Macros can be placed at any place in the code.

The declaration section is divided into three main sub-sections:

1. The node declaration. Terminal nodes can be declared as input node, output node or inout node. They are all electrical nodes. Verilog-AMS allows the use of various types of nodes such as mechanical nodes. Internal nodes can also be defined.
2. The branch definition. Names of branch across two nodes can be defined. This makes the coding of a model clearer.
3. The parameter declaration. Device parameters are defined in this section. Special attributes can be set to parameters. A parameter can be defined as a model parameter or as an instance parameter. Values of a model parameter only depend on the process characteristics. Values of instance parameters depend on geometries of the device. For the sake of simplicity, all parameters will be declared as model parameters, which is the default declaration.

The constitutive equations section defines the relationship between voltage potentials of the devices and branch currents.

In example presented Fig. 2 an if-based switch activates either the constitutive equations of an n-type HBT, or the constitutive equations of a p-typed HBT. The selection of the switch is determined by the value of parameter `type'.

The constitutive equations compute the magnitude of current flowing across the different branches of the HBT device. Once the current values are calculated, the values are "loaded" into the device by the means of the special branch contribution symbol "<+".

In the next Section we will present and discuss the implementation of the HBT device into three different electrical circuit simulators.

3. HBT model implementation

This Section will present the results of the implementations of the HBT compact device model described in the previous Section.

3.1. Basics on model implementation

A compact device model description is made of equations. These equations describe the relationship that exists between terminal voltages of the model, and the branch currents of the model.

When these equations are well designed, and fit with the physics of the device model its implementation into electrical circuit simulators can begin.

Most of the time implementing a device model into a new electrical circuit simulator means formatting the constitutive equations of the model into c source code.

Programming interfaces are provided within electrical circuit simulators. Programming interfaces are guidelines for c source code programmers. They give access to embedded routines of the circuit simulators. Most of them provide a support to create new model parameters. They give access to more

sensitive data like the jacobian builders of the Newton-Raphson algorithms.

3.2. Netlist used for test-bench

In the following paragraph, a basic circuit will be used to test-bench the implementation of the device model into different simulators. Netlist of the circuit is given in Fig. 3.

```
* hbt – mica netlist
* test-bench for basic implementation test
vcc 1 0 10
r1 1 4 10k
r2 4 0 10k
r4 5 0 2k
c2 4 0 1e-8
c4 5 0 1e-5
c5 1 0 1e-5
q1 2 4 5 mod1
.model mod1 npn level=3
.option dcmethod=pseudotran
.control
op
show mod1
.endc
```

Fig. 3. Basic test-bench netlist.

3.3 Model implementation in MICA

MICA [3] is the internal electrical simulator of Motorola. MICA is a Motorola-proprietary internal product. MICA provides a programming interface for new device model implementation called DPI, acronym of Device Programming Interface.

```
Using /nopt/mica1.2.0/bin/hppa8000/mica
Saving 10 output vectors.
DC analysis, iter 18, loads 18,
dx 1.624781e-17, error 5.780582e-17
Note: Option temp set to 27
HBT_m:
        level        level number     = 3
        is           no description   = 2E-11
        bf           no description   = 225
        br           no description   = 5
        nf           no description   = 1
        nr           no description   = 1
        type         no description   = -1
rb#i = 3.495725e-04
rc#i = 7.865381e-02
re#i = 7.900338e-02
v#Bint = 6.504275e-01
v#B = 1.000000e+00
v#C = 1.000000e+00
v#Cint = 9.213462e-01
v#Eint = 7.900338e-02
vb#i = -3.49572e-04
vc#i = -7.86538e-02
```

Fig. 4. Simulation results of MICA dc analysis.

The ADMS package includes a source code generator for the programming interface of MICA. The source code generator is simply called admsMica. From the Verilog-AMS description presented in the previous Section admsMica will create a set of ready-to-compile-and-link c source files. A shared library is created after linking all compiled files created by admsMica. At run time MICA will load the shared library and the new HBT device will be ready for use. Figure 4 shows the simulation results of the test-bench circuit presented in Fig. 3.

3.4. Model implementation in SPECTRE

SPECTRE [4] is the electrical circuit simulator of CADENCE. It is a commercial product. The simulator provides a programming interface called CMI, acronym of Compiled-Model Interface. The CMI is not distributed with the simulator. Access to the CMI is granted upon special request to CADENCE.

The ADMS package includes the source code generator for SPECTRE called admsSpectre. After running admsSpectre ready-to-compile-and-link c source files are created. In its present version SPECTRE does not offer the possibility to load new compact device shared library at run time. All the binary that builds SPECTRE should be linked with the compiled files of the HBT device models. As a result, a new executable is created and ready to use. Figure 5 gives the simulation results of the test-bench circuit of Fig. 3.

```
spectre (ver. 4.4.3.cmi.solaris. -- 17 Nov 00).
Simulating `HBT_spectre.ckt' on thun
at 4:14:11 PM, Fri Apr 19, 2002.
Circuit inventory:
nodes 5
equations 7
HBT 1
resistor 3
vsource 2
setTnom: `tnom' set to 27 C.
*********************
DC Analysis `opPoint'
Operating point computed in DC analysis
`opPoint' at T = 27 C.
V(B) = 1 V
V(Bint) = 650.428 mV
V(C) = 1 V
V(Cint) = 921.346 mV
V(Eint) = 79.0034 mV
I(vb:p) = -349.572 uA
I(vc:p) = -78.6538 mA
Instance: HBT
Model: myhbt
Primitive: HBT
c : V(Cint) = 921.346 mV
b : V(Bint) = 650.428 mV
e : V(Eint) = 79.0034 mV
s : val(0) = 0
Total Power Dissipation = 79.0034 mW
Convergence achieved in 22 iterations.
Total time required for dc analysis `opPoint'
was 10 ms.
Aggregate audit (4:14:13 PM, Fri Apr 19, 2002):
Time used: CPU = 120 ms, elapsed = 2 s, util.=
6%.
Virtual memory used = 631 kbytes.
spectre completes with
0 errors, 0 warnings, and 0 notices.
```

Fig. 5. Simulation results of SPECTRE dc analysis.

3.6. Model implementation in ADS

ADS [5] is the commercial electrical circuit simulator of Agilent. ADS is a commercial product. Its programming interface is called UMI, acronym of User Model Interface. The UMI is distributed with the ADS package. It is well documented and available to any users of ADS.

The source code generator of ADS is called admsAds. Like SPECTRE, the current version of ADS does not support run-time loading of compact device models. However, work on the implementation of this feature is on going. Figure 6 shows the simulation results of the test-bench circuit presented in Fig. 3.

```
HPEESOFSIM  (ver. "170" rev. "200")
Copyright Agilent Technologies, 1989-2001.
DC DC1[1] <HBT_ads.ckt>
Convergence achieved in 10 iterations.
DC Operating Point:
  V(Cint) = 921.346 mV
  V(Bint) = 650.428 mV
  V(Eint) = 79.0034 mV
  V(C) = 1 V
  V(B) = 1 V
  vb.i = -349.572 uA
  vc.i = -78.6538 mA
------------------------------------------------
Simulation finished: dataset `networks' written
in:

`/user/lemaitre/_ADS/api_170_200/compact_device
s/hbt_prj'.
------------------------------------------------
Resource usage:
    Total stopwatch time: 24.57 seconds.
```

Fig. 6. Simulation results of ADS dc analysis.

3.7. Comments on the different implementations

From above simulation results, we can easily notice that all implementations of the HBT device model are "aligned". Discrepancies between two implementations are mostly limited to the way simulators are printing results. Input Model parameters have the same names and the same default values. Device topologies are equivalent between different circuit simulators. Calculated values of voltage potentials and current flows are identical up to the 6^{th} digit. Model implementers can really compare numerical results between two implementations.

This makes the maintenance of model development a lot easier. Improvements made on the robustness of a compact device model will propagate easily to all implementations. Weaknesses on the behavioral implementation of the model can be detected in one simulator and fixes to the weakness can be propagated to all other simulators.

4. ADMS MOSFET modeling example

This Section concludes the paper by describing the implementation of a MOSFET model into ADS using the user model interface (UMI) provided by Agilent [5]. The EPFL-EKV compact model [6–8] is used to illustrate the implementation procedure.

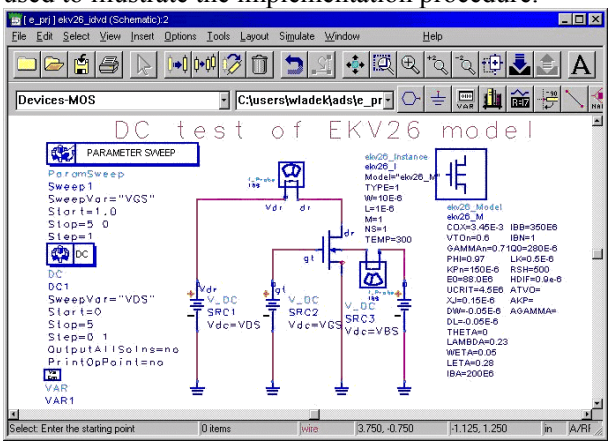

Fig. 7. ADS schematic view of a simple circuit with the EKV model implemented using ADMS tool.

Creating new ADS model consists of three main steps: Defining the parameters that the user will enter from the schematic, Writing the c-code itself and Defining the symbols and the corresponding pins. Following these steps a new ADS model can be used in linear, nonlinear (i.e. harmonic balance), transient and circuit envelope simulation modes. The ADMS processes the first two steps. Based on the Verilog-AMS compact model description, the ADMS tool generated all necessary c-code to handle model and instance parameters, model codes including all additional functions, as well as required derivatives in respect to terminal voltages. The ADMS tool also generates needed make files to simplify procedure of compiling and linking new model with the simulator.

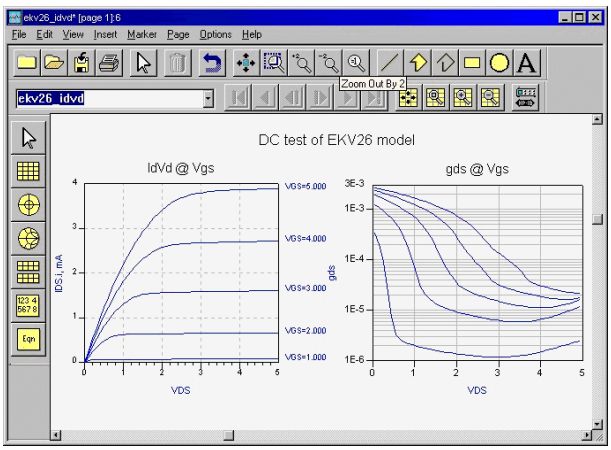

Fig. 8. Results of output current and conductances simulations.

The only user responsibility is to create a new model and its instance symbols for the ADS schematic editor. An example of a simple schematic

with the EKV model for DC simulations is shown in Fig. 7.

Figure 8 shows corresponding DC simulation results of output characteristics (drain currents, I_d, and drain conductances, g_{ds}, respectively).

5. Conclusions

The ADMS tool offers an excellent modeling environment. It allows faster development of advanced models and faster implementation into commercial IC design tools. Existing models could be smoothly extended to include important effects such as thermodynamical effects [9]. Furthermore, developers of new compact models (i.e. [10], [11] now have access to the coherent and highly reliable modeling framework simplifying model evaluation procedures and verification tasks across different simulation platforms and operating systems.

REFERENCES

1. L. LEMAITRE, C. MCANDREW, S. HAMM , *ADMS – Automatic Device Model Synthesize*, CICC 2002, Florida, USA.
2. *Verilog-AMS Language Reference Manual*, Open Verilog Int., 1999.
3. *MICA Device Programming Interface*, Documentation and Programmer's Guide, Motorola Internal Document, 1998.
4. *SPECTRE CMI Reference Manual*, Cadence Design System, 1995.
5. *Analog/RF User-Defined Models*, Agilent Technologies, April 2001
6. C. ENZ, F. KRUMMENACHER, E. VITTOZ, *An Analytical MOS Transistor Model Valid in All Regions of Operation and Dedicated to Low-Voltage and Low-Current Application*, J. Analog Integrat. Circ. a. Signal Process., 1995, **8**, 83–114, Kluwer Acad. Pub.,
7. M. BUCHER, C. LALLEMENT, C. ENZ, F. THÉODOLOZ, F. KRUMMENACHER, *The EPFL-EKV MOSFET Model Equations for Simulation*, Version 2.6, Technical Report, Electronics Laboratory, Swiss Federal Institute of Technology Lausanne, (EPFL), June 1997.
   Web Resources: http://legwww.epfl.ch/ekv.
8. M. BUCHER, *Analytical MOS Transistor Modeling for Analog Circuit Simulation*, Ph.D. Thesis No. 2114, 1999, EPFL, Lausanne.
9. C. LALLEMENT ET AL., *High Level Description of Thermodynamical Effects in the EKV 2.6 Most Model*, Proc. 9th Int. Conf. on Mixed-Signal Design (MIXDES), Wrocław, Poland, June 2002.
10. T. L. CHEN, G. GILDENBLAT, *Analytical Approximation for the MOSFET Surface Potential*, Solid-St. Electron., 2001, **45**, 3335.
11. M. BUCHER ET AL., *EKV 3.0: An Analog Design-Oriented MOS Transistor Model*, Proc. 9th Int. Conf. on Mixed-Signal Design (MIXDES), Wrocław, Poland, June 2002.