I.M.A.G.E.: a New CAD Tool for Device Modeling in Spice

M. Zorzi*, F. Franzè and N. Speciale

Abstract — **In this paper we describe I.M.A.GE., a new CAD tool to embed semiconductor device models and, more generally mathematical models of non electric devices inside a spice-like circuit simulator. User defined models can be developed and tested quickly, allowing efficient simulation of even large circuits. To outline the performance improvement and flexibility obtained by using our tool, we present two different application examples: the implementation of a fluorescent lamp model and a "multiterminal" smart-power bipolar device.**

1 Introduction

As the semiconductor industry grows and new technologies are introduced, a need for good and updated electrical models grows accordingly. Modern electronic design is strongly based on circuit simulation, so the accuracy and completeness of the compact models and characterization of the circuit elements and parasitics are mandatory. Unfortunately, it is difficult to develop complete, accurate, numerically robust models that work properly with the numerical algorithms in simulators and with other CAD tools in an IC design system.

For these reasons, the need to test and validate new device models is a main issue inside the modeling research community. Some different possibilities are available, but some of them really do not look very appealing.

The first and more trivial idea is to build and test a new device model as a macromodel, and implement it as a standard subcircuit. Even if this is a fast and not very difficult task, the performances are strongly affected by the complexity of the model, and this could make almost impossible its use inside a circuit containing a great number of devices.

Another solution is to directly write the source code for the model and embed it inside the simulator. This is of course the most performing approach but it is also the most difficult and time consuming: we need the source code of the simulator, the know-how about writing new models, and obviously about programming languages and compilers. Moreover, the writing-compiling-linking-running-testing cycle could be quite heavy till the model is not entirely debugged.

An intermediate solution could be to develop a model by using a behavioral language [1][2][3]. Likely we will have some benefits considering the performances comparing to the subcircuit approach, and we do not need expert insight of the simulator. However, in spite of most recent developments, this high level language is mainly devoted to digital and system applications but could be not very flexible to describe low-level device model.

A more appealing solution would be to have a tool which automatically generates the code for the new device model to embed inside the simulator. In this case, we could have all the benefits of the previous approaches without the drawbacks: the model can be written in a suited high level language which is quite far from implementation details but it is able to produce very good model performances.

Similar approaches are reported in literature [5], providing large and complete sets of operations and analyses to simulate board level circuits and embedded systems such as those that mix hardware and software, analog and digital electronics, electronics and mechanical devices for specific simulator. Differently, the proposed tool can automatically generate portable code suitable to be used with different circuit simulators based on the original Berkeley Spice [6][7] to model new compact models.

This research was motivated by the need of both a more reliable, user friendly defining procedure and a vehicle for implementing and testing new set of advanced "multiterminal" smart-power device models for circuits simulation developed by our group [8][9].

The paper is organized as follows: in the next Section we introduce the I.M.A.GE. tool and how it interacts with the simulator; in Section 3 we show the I.M.A.GE. syntax explaining how to build a model, and in Section 4 we report two examples: a fluorescent lamp model [10] and a multiterminal device [9].

2 Overview of the I.M.A.GE. tool

The program I.M.A.GE. (**I**nternal **M**odel **A**utomatic **GE**nerator), represents an efficient solution to cope with the problem of automatic code model generation and implementation. Roughly

*DEIS. University of Bologna, Viale Risorgimento 2 40136 Bologna, Italy. E-mail: mzorzi@deis.unibo.it, Tel: +39-051-2093777, Fax: +39-051-2093073.

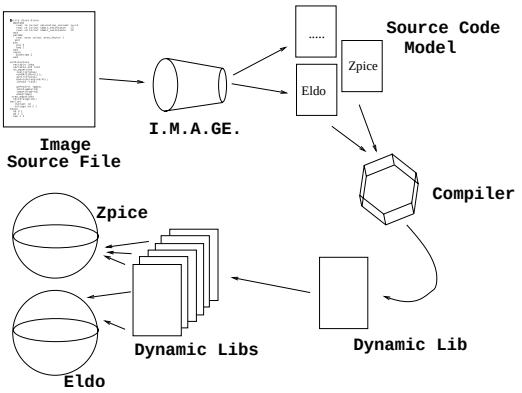


Figure 1: The I.M.A.GE.-Zpice Architecture

speaking, adding a new model in a simulator is a two-step process: first, I.M.A.GE. takes as input a file written in a simple language, describing the device model. Then, it creates a directory and the source files needed to build the model code. Though it is possible to embed the generated code inside either the standard Berkeley Spice or commercial standard CAD packages such as ELDO[11] and compile the new extended simulator, we adopted a slightly different approach, developing Zpice, a spice-based simulator which provides a dynamic support to the device models developed by our group with no limitation in the number of internal nodes, static currents and charges and external pins. By so doing, all the device models remain external to the simulator core and are dynamically load only at run-time, according to the indications in a setup configuration file (see figure 1). This solution allows great flexibility and makes the process of incorporating a new model briefly, since the core of the simulator is left unmodified and only the effective new adding library has to be compiled. Moreover no source code details know-how about the simulator is needed, as a manual integration does.

3 The I.M.A.GE. language, how to build a model

The I.M.A.GE. source language is quite intuitive since it borrows many syntax constructs from well known behavioral languages such as VHDL [4] and Spice netlist source file. The input file is composed by different sections: it must contain the model name, a list of model and instance parameters, and the internal structure of the device model.

The syntax for a generic model is reported in Figure 2: in the first line the name of the model is defined, then the sections **mparams** and **params** are used to introduce the model and instance parameters. The following sections contain the definition of the model internal architecture: **pin** and **inpin** are used to define the external and internal nodes respectively; section **architecture** defines the equations for the several internal devices, while in the **netlist** section the circuit for the internal device companion model is reported. By creating multiple architectures for the same device model, the user can simply change architectures to choose an acceptable speed versus accuracy tradeoff prototyping. Finally, the **convergence** section contains the voltages and currents to use in the simulator convergence test.

The model reference in Zpice follows the general form used by other standard devices implemented in Spice [6]. To represent a generic Zpice device, the instance name must begin with letter "y" followed by the number of terminals, the connections list and the name referencing the .model card. It is also possible to add new models for existing devices by using different level numbers. Moreover, I.M.A.GE. checks with care the model definitions and prevents the user to implement an ill-defined device that can give numerical problems during simulation, reducing convergence problems. As shown in the next section, the I.M.A.GE. language is situated at an intermediate level, not too much near to the C-code, but also not so abstract as a behavioral language, where it is almost impossible to keep under

```
entity <model> <modelName>
  mparams
   <model param>
   <model param>
   ...
  end
  params
   <instance param>
   ...
  end
  pin
    <name> <number>
    ...
  end
  inpin
    <name> <number>
    ...
  end
  architecture
   variable <name>
   dc_equations
     <equations>
   tran_equations
     <equations>
   netlist
     current|voltage <name>
   begin
     <netlist definition>
   end
   convergence
     <convergence conditions>
   end
   end
```

Figure 2: An I.M.A.GE. source file template.

control the behavior of the simulation avoiding convergence problems. This feature is accomplished by allowing both the modification of important internal quantities (voltages, currents, charges), and preventing the user to set incorrect values.

4 Examples and Performance Comparisons

As first example of the I.M.A.GE. use, we present the implementation of a dynamic fluorescent lamp model [10].

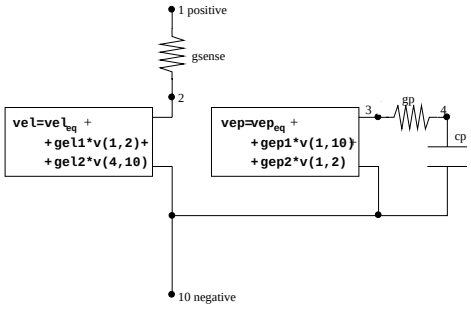
Basically, the definition of a device needs an high and a low level view: the former is the entity containing model and instance parameters; the latter is the architecture, containing the netlist and the device equations. The definition of the high level view is a simple task, since this can be obtained directly from the macromodel in [10]; it contains 8 parameters (A1-4, B1-4), 4 voltage controlled voltage source blocks (BA, BB, BP, BL) a couple of resistances and a capacitance.

We have to define the entity view and, according to the macromodel, the parameters (A1-4,B1-4). This is achieved respectively with the **entity** and **mparams** sections. Finally, the definition of the low level section involves the following three steps: (*i*) creation of the companion model as an I.M.A.GE. netlist, (*ii*) definition of **dc_equations** and **tran_equations**, (*iii*) definition of the pin names for external and internal netlist nodes. The companion model is obtained from a linearization of BB, BA, BP and BL equations; this leads to the circuit and netlist in figure 3.

It is worth noting that the companion model has only 5 nodes (3 internal and 2 external), against the 7 nodes of the macromodel, since BA and BB are calculated in the I.M.A.GE.'s **dc_equations** section.

Another important feature of I.M.A.GE. source language is that the model equations in **dc_equations** and **tran_equations** sections are entirely written in C, and we can keep under control the behavior of the model internal quantities, like voltages or currents, to a low level, thus avoiding some convergence problems that are hard to control in a high level language. In this way, we have access to the branch voltages, whose variation can be limited using internally defined functions, to prevent floating point overflows.

Figure 4 shows comparisons between macromodel and I.M.A.GE. model lamp output voltage using 0.17A rms sine input waveform. Comparison between macromodel and I.M.A.G.E. output shows a very good agreement. Nevertheless it is worth noting the improved performance of the generated




```
  ..
dc_equations
  ..
  BA=a1/(vpl)+a2+vpl*(a3+vpl*a4);
  BB=(b1/(vpl)+b2+vpl*(b3+vpl*b4));
  ..
 end
 ..
netlist
  voltage vpl       4  10
begin
  vsrc              3  10          vep
  vsrc              2  10          vel
  cond              3  4           gp
  cond              1  2           gsen
  cap               4  10          cp
..
end
```

Figure 3: High-Frequency Dynamic Fluorescent Lamp companion model

device, that is about 4 time faster than the PSpice macromodel.

A second example is represented by the implementation of compact models for transistors used in an advanced smart-power technology with junction isolation and vertical power current flow [8]. In particular, the VIPower^{TM} technology combines an emitter switching power stage with low voltage bipolar and MOS transistors. All the low-voltage circuitry is integrated inside a *p*-type buried-layer

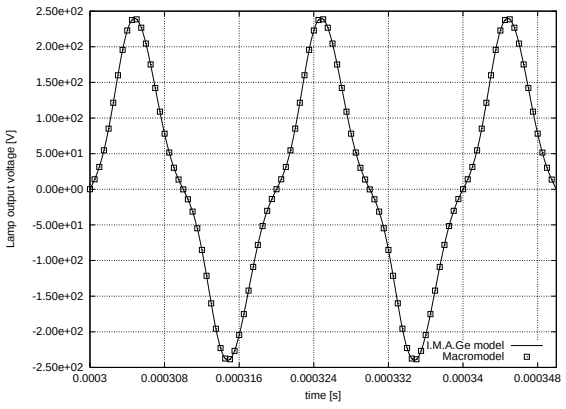


Figure 4: Comparison between macromodel and I.M.A.GE. model lamp output voltage with 0.17A rms sine waveform stimulus.

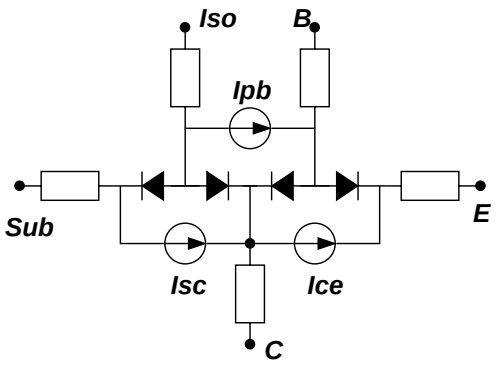


Figure 5: Compact model for a VIPower^{TM} low-voltage *npn* bipolar device with parasitic effects.

isolation-well diffused in the n-type epitaxial layer which acts as the collector of the vertical power device. For this reason, the substrate is not fixed biased and can undergo abrupt and strong voltage variations (up to 1500 V). Thus, during the design phase of the low-voltage circuitry it is compulsory to carefully model the interactions among different semiconductor regions. Therefore, suitable new compact models for these "multiterminal" devices have to be defined, since standard BJT and MOSFET models can not predict at all these parasitic effects. The definition of compact model for these devices is a very complex task and so is their testing. By using the code produced by I.M.A.GE., it is possible to evaluate different modeling approaches and numerical implementations effectively and in short time.

Figure 5 shows the schematic circuit diagram of the proposed "multiterminal" compact model for a *npn* transistor, where the parasitic bipolar action among the several layers has been represented with diodes and voltage dependent current generators [9].

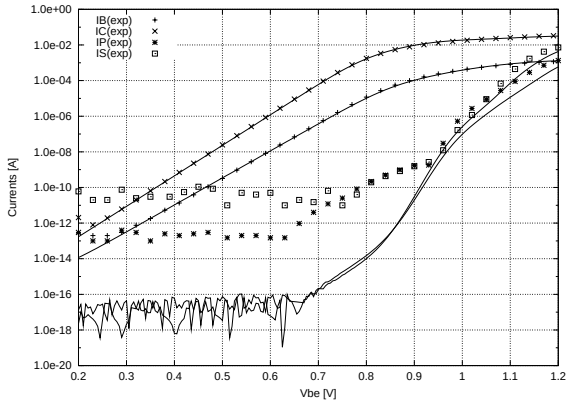


Figure 6: Comparison between simulated (continuous lines) and measured (dots) currents for a *npn* transistor ($V_E = 0$, $V_B = V_C = V_{ISO} = V_{SUB}$=[0.2:1.2]).

Figure 6 shows the comparison between measured and simulated obtained with the following Zpice netlist:

```
Five Terminals  NPN
..
.MODEL npn5t npn5t Issb =6.3695e-17
+      Isnb =7.1832e-17
..
*  # C B E Iso Sub
Y1 5 1 2 3  4   5  npn5t
*Analisys and outputs
.DC VB 0.2 1.2 0.05
.PRINT DC
+@Y1[ic] @Y1[ib] @Y1[ie] @Y1[ip] @Y1[is]
.END
```

At present, the implemented models are used to predict possible failures on the low-voltage CMOS and bipolar control circuitry induced by variations in the substrate voltage associated with operations of the power device integrated on the same substrate.

5 Conclusions

We presented I.M.A.GE., a CAD tool to generate portable code to embed new device models in different circuit simulators. We showed the tool flexibility especially for modeling non conventional complex devices and defining robust numerical code. Finally, we reported two application examples to outline the results obtained using the proposed method.

6 References

[1] *"HDL-A Reference Manual"*, Mentor Graphics, 1995.

[2] *"SPECTRE HDL Reference"*, Cadence Design Systems, 1998.

[3] J. Davis et al. *PTOLEMY II, Heterogeneous Concurrent Modeling and Design in Java*, Univ. of California.

[4] P. J. Ashenden *The Designer's Guide to VHDL*, Morgan Kaufmann Publishers, Inc. 1996.

[5] F. L. Cox, W. B. Kuhn, J. P. Murray, S. D. Tynnor *"Code-Level Modeling in XSPICE"*, Proc. of ISCAS'92. Vol 2 , 1992 , pp: 871-874.

[6] A. Vladimirescu, *The SPICE Book*, J. Wiley, N.Y. 1994.

[7] T. Quarles, *Adding Devices to Spice3*, Memorandum No. UCB/ERL M89/45 24 April 1989.

[8] N. Speciale, G. Onofri, S. Graffi, G. Masetti, S. Palara, G. Privitera *"A New DC Model for Five-Terminals Bipolar Devices used in Smart-Power ICs"*, Proc. of ISCAS'96, Atlanta, 1996.

[9] N. Speciale, A. Leone, S. Graffi, G. Masetti *"Unified Approach for Modeling Multiterminal Bipolar and MOS Devices in Smart-Power Technologies"*, Proc. of 27^{th} ESSDERC97 Stuttgart, Germany, Sept. 22-24 1997.

[10] M. Sun, B. L. Hesterman *"PSpice High-Frequency Dynamic Fluorescent Lamp Model"* IEEE Trans. on Power Elect. Vol 13, No 2 March 1998 pp. 261–272.

[11] *"ELDO UDM User's Manual"*, Mentor Graphics, 1999.