i-MOS: A Platform for Compact Model Sharing

Hao Wang and Mansun Chan

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, eewanghao@ust.hk

ABSTRACT

This paper describes the develop of *i*-MOS (interactive Modeling and Online Simulation) platform for compact model sharing and dissemination. *i*-MOS is constructed using an opensource simulation engine (ngspice) and Verilog-A compiler (ADMS), together with a customized web-interface. Model users can perform simulation and model verification directly using the *i*-MOS platform. Model developers can submit their models coded in Verilog-A format and they will be available for users to test and verify. The system design, available servicse and current status of the project will be explained in different sections of this paper and an example of how a model can be incorporated into *i*-MOS will be given.

Keywords: compact model, SPICE, circuit simulation, Verilog-A compiler, online simulation

1 INTRODUCTION TO INTERACTIVE MODELING

The migration of the research community to non-traditional CMOS devices results in a large variety of device types. The process to bring new devices to circuit applications involves the development of compact models to be used in circuit simulation. With the increasing complex physics involved in the newly invented devices, it quires better and faster model development and distribution methodology.

Traditional compact model development method requires a good understanding of a SPICE-like simulation engine such as Spectre [1]. After deriving all the required current equations, the model including current-voltage, charge-voltage, transconductance and transcapacitance has to be coded in C-language and compiled together with the simulation engine. The process is usually tedious and non-productive.

To address the need for a more userfriendly modeling environment and reduce the burden for model coding, Automatic Device Model Synthesizer (ADMS) [2] has been developed. It allows model developers to implement their model using high-level language such as Verilog-A [3] and then compiled into C-code. It has been a significant advance in the model development environment. However, a model user still needs to compile their own version of SPICE-like simulator before the can run simulation with new device models. This create a barrier for model distribution in the early stage of model development and delay the enhancement based on early user feedback.

As a result of the current modeling environment, we have developed an interactive online model distribution platform called *i*-MOS (interactive Modeling and Online Simulation) on top of the Verilog-AMS and ADMS paradigm. In the *i*-MOS platform, the simulation engine is located in a dedicated server following the cloud computing approach. Model developers can submit their model through the *i*-MOS platform and users can evaluate the model directly through the internet.

2 *i*-MOS USER INTERFACE

i-MOS is developed based on the world-wide-web platform that can be executed by any web-browser. After typing the corresponding hyperlink [4], the front page of *i*-MOS will be as shown in Figure 1. It serves as a marketplace for model developers to upload their models and users to perform simulation with the available models. In addition, the platform also provide information of recent modeling activities and a forum to discuss modeling related issues.

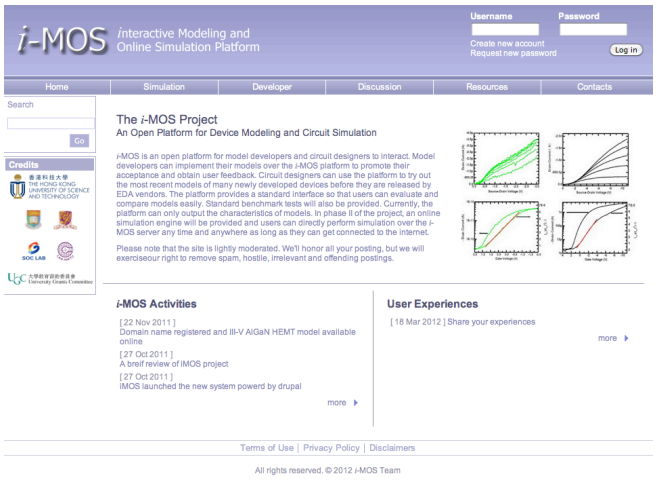

Figure 1: Browser display after entering the *i*-MOS.

2.1 Performing Simulations

Currently, *i*-MOS only allows model characteristics to be displayed or download. We are planning to allow user users to directly construct netlist and perform simulation in the future enhancement. By clicking into the model page, users may select any available models uploaded to *i*-MOS by different model developers. After model selection, the model parameter page will be displayed. Model parameters of the device to be simulated will be entered here. If users do not know what parameters to be entered, there are predefined parameter sets to be used. Finally, the biasing variables are entered and simulation will be performed. Based on the selected output parameters, the results can either displayed in graphical form or downloaded in table form.

2.2 Uploading Models

i-MOS supports the Verilog-A [3] as the modeling language to for model implementation. At the present state, the bottlenect of using Verilog-A is the lack of a well-developed opensource compiler (which will be the next step of research in this work). The most readily available opensource Verilog-A compiler is ADMS [2] but it has some limitation and only a subset of Verilog-A syntax is supported. Currently the process is only semi-automatic with some minimal human actions required.

The Verilog-A coded model is divided into different parts including model parameter definition instance parameter definition, variable definition, analog function and evaluation functions. Only a subset of Verilog-A format is suitable for compact modeling and the coding of the model is expected to follow some good modeling practice such as those specified by L. Lemaitre et. al. and G. J. Coram [5,6]. As the underlying simulation engine of *i*-MOS is derived from ngspice the Verilog-A subset specification for the compact modeling provided by G. Depeyrot et al. [7] should be followed with some additional *i*-MOS restrictions. Some of the requirements are highlighted below:

- Use only integer and real data types;
- Use only scalar but not vectors or arrays;
- Limit the contribution statements to current versus voltages;
- Limit the use of ddt, ddx, and idt to the first order;
- Limit the system tasks and functions to $vt, $temperature and $mfactor only;
- Limit the use of electrical discipline with voltage and current natures;
- Do not use module instantiation or hierarchical structures.

After the model is coded in Verilog-A following the specification defined by *i*-MOS and fully debugged, it can be uploaded to the *i*-MOS platform and the model should be available in *i*-MOS shortly.

In addition to the model in Verilog-A code, some additional information is needed for the purpose of model maintainance and interface standardization. It includes name of the model, authors, list, author contact information, icon representing the model, parameter list, input/out variables etc.

3 SYSTEM ARCHITECTURE

The web system is constructed using the LAMP solution (i.e. Linux, Apache, MYSQL, and PHP) and the overall system architecture is illustrated in Figure 2.

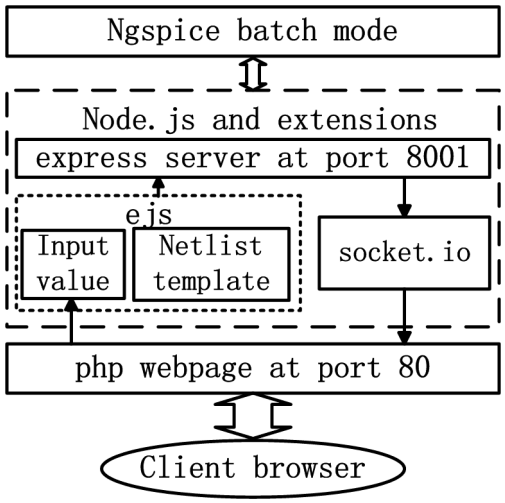

Figure 2: System architecture of *i*-MOS.

Most of the weblogics of the system is implemented using the Node.js package [8], a server side Javascript environment with event-driven programming capability for scalable web server applications. Several third party extension module including ejs, express, forever and socket.io for Node.js are also used. For each model, a template netlist file is provided to display various output characteristic curves with the model parameters denoted by their names as shown in Figure 3.

When a user input the values for the parrameters, ejs generates a temporary netlist file for this simulation session with the user adjusted parrameters. Express is then used to create an http server for Nodje.js with a specific port number other than the standard port 80. The express server is responsible for the communication between between Node.js and the ngspice simulation engine resides at the cloud server. Ngspice is evoked by the Node.js Javascript in batch mode to execute the simulation according to the temporary netlist file. When the simulation is completed, the results are redirected to part 80 of the client browser with the socket.io extension. They can then be displayed to the client browser or saved to a file.

```
…
.model CNT diameter={{=diameter}} angle={{=angle}}
+ tins={{=tins}} eins={{=eins}} tback={{=tback}}
+ eback={{=eback}} + types={{=types}} L={{=L}}
+ phisb={{=phisb}} rs={{=rs}} rd={{=rd}} beta={{=beta}}
+ Cc={{=Cc}} mob={{=mob}} Csubfit={{=Csubfit}} Cp={{=Cp}}
…

…
.model CNT diameter=1e-9 angle=0
+ tins=10e-9 eins=16 tback=130e-9
+ eback=3.9 types=1 L=115e-9
+ phisb=0.1 rs=0 rd=0 beta=16
+ Cc=7e-12 mob=1 Csubfit=0.4 Cp=40e-12
…
```

Figure 3: The device parameter segment in the provided netlist template file (top) and the generated netlist file after parameters input by the user (bottom)

4 MODEL DISSEMINATION EXAMPLE

A carbon-nanotube-FET (CNTFET) model developed by Arizon State University [9, 10] will be used to illustrate the process to share models through *i*-MOS. One reason for selecting this example is because the coding of the CNTFET model is close to the requirement specified by *i*-MOS. However, some syntax not supported by the current version of ADMS need to be modified. Some methods to handle unsupported cases such as mixed current and charge contribution, assignment to potential etc. can be found in the documentation of the QUCS project [11] which uses ADMS as a Verilog-A to C++ converter. Before the conversion, the CNTFET code is pre-processed. For instant, forloop is not supported by the current version of ADMS [12]. All forloops need to be replaced by whileloops. Fro example, the following code:

```
for (ss=1; ss1 < (2*NN); ss1=ss1+1)
    begin
        …
    end
```

need to be replaced by

```
ss1=1;
while (ss1<(2*NN))
    begin
        …
        ss1=ss1+1;
    end
```

Calculating potential from other variables is common in Verilog-A model, but it is unsupported. Statement assigning current to a voltage variable need to be re-coded. For example:

```
V(dbar) <+ I(dbar)*rd;
```

has to be recoded to:

```
if(rd > 0.0)
    I(dbar) <+ V(dbar)/rd;
  else
    V(dbar) <+ 0.0;
```

In Verilog-A code, model parameters and instance parameters are not differentiated. However, they are handled differently in SPICE. In most cases, model parameters handle process dependent characteristics while instance parameter handle characteristics related to lateral geometry. In *i*-MOS, the two type of parameter has an additional distinction because model parameters remain constant throughout the simulation while instance parameters are re-calculated and stored at each simulation point. Therefore, if certain quantities (e.g. gm, gds) are being treated as output variables, they need to be specified as instance parameter. It should be noted that the default parameter type is model parameter and instance parameter can be defined as follows:

```
//Instance Parameters
parameter real diameter =1.0e-9 from [0:inf)
    (*type="instance" info="Tube diameter [m]"
    unit="m"*);
parameter real angle = 0 from [0:30]
    (*type="instance" info="Chiral angle [deg]"
    unit="deg"*);
    …
parameter real L = 100e-9 from [0:inf)
    (*type="instance" info="Length of the nanotube
    [m]" unit="m"*);

// gm and gds defined as instance parameters
parameter real gm=0.0
  (*type="instance" info="gm" unit=""*);
parameter real gds=0.0
  (*type="instance" info="gds" unit=""*);

//Model Parameters
parameter real phisb = 0 from (-inf:inf)
  (*info="Barrier Height"*);
parameter real rs = 0 from (-inf:inf)
  (*info="Source Access resistance"*);
    …
parameter real Cp = 0 from (-inf:inf)
  (*info="Parasitic capacitance"*);
```

For each parameter, a default value, range of value and a short description is provided. All these parameters are treated as inpu and output type (IOP) for easy handling. In principle, the transconductance gm and source drain conductance gds should be available in the Jacobian matrix

in SPICE. But it is recommended that they are evaluated with explicit assignment in Verilog-A code to avoid a evaluation step between converting and compiling. In the CNTFET case, gm and gds are evaluated in the converted C-code:

```
pInst->gm=Ichan_Vg_s;
pInst->gds=Ichan_Vd_s;
```

Attention should be paid that the identifiers in Verilog-A is case sensitive but it is not in spice netlist file. So that "Variable" and "variable" are different in Verilog-A but the same in netlist. Variables differ by case only should be avoided in Verilog-A code.

In the original CNTFET model, a $table function is used but is not supported in the current version of ADMS. It has to be realized in a separated function with hand coding to the compiled C files. For example, the following C-code need to be hand-coded to the compiled files

```
NN=floor(cnt_NN_tbl(pInst->diameter, pInst->
    angle));
```

with the following statement in the Verilog-A source code

```
NN= floor($table_model (diameter, angle,
    "NN_table.tbl"));
```

The above procedure can be automated but currently the responsibility rest with the model developer or the *i*-MOS team members. With all the syntax of the Verilog-A file checked, C codes are generated by ADMS. As the device models are invoked statiscally in ngspice, the simulator needs to be compiled to include the new model library, which can be carried out only by the *i*-MOS team members. A procedure to invoke device models from a dynamic library is under construction, which may enable the immediate application of new models without the need to recompile the ngspice simulator.

After uploading the Verilog-A file to *i*-MOS from the developer page, other model information will be prompted. The *i*-MOS team will examine the model to ensure it is valid and complete. The model is then linked to the cloud server available to be used by other users.

5 CONCLUSION

Compact modeling is a very important activity to bridge newly developed device technology to circuit application. However, the infrastructure for compact modeling is not very well developed leading to significant barrier from model development to distribution. The development of Verilog-A modeling language and ADMS compiler has address the need to simplify the model development process. The *i*-MOS is developed on top of the Verilog-A and ADMS paradigm to facilitate the integration of a model into a simulation engine and distribute to users through the web platform using the cloud computing approach. The current system is not perfect and there are many areas that need improvements. We will continuously improve the quality of the service in the future and we that our effort can provide valuable support to the compact modeling community.

ACKNOWLEDGEMENT

This work is support by Hong Kong University Grant Council under the Area of Excellence Scheme with contract number AoE/P-04/08. This project is the result of a group effort that involve many students ard researchers. In particular, we would like to acknowledge the group members who have directly involved in implementing the *i*-MOS platform. They are Cyrix K. C. Tam, Hamza Zia, Xiaoxu Cheng, Lining Zhang and Zubair Ahmed.

REFERENCES

[1] http://www.cadence.com/products/custom_ic/spectre/
[2] L. Lemaitre, C. McAndrew, and S. Hamm, "ADMS-automatic device model synthesizer", IEEE CICC 2002, pp. 27-30
[3] Verilog-AMS Laguage Reference Manual, Open Verilog International, 1999
[4] http://i-mos.org
[5] L. Lemaitre, C. McAndrew, and W. Grabinski, "Standardization of Compact Device Modeling in High Level Description Language," Proc. Nanotech, pp. 372-375, 2003.
[6] G. J. Coram, "How to (and how not to) write a compact model in Verilog-A," 2004 IEEE International Behavioral Modeling and Simulation Conference, BMAS, pp. 97-106
[7] G. Depeylon, F. Poullet, and B. Duwas, "Verilog-A Compact Model Coding Whitepaper," presented at the Nanotech 2010 Proceedings, 2009.
[8] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," IEEE Internet Computing, vol. 14, pp. 80-83, 2010.
[9] A. Balijepalli, S. Sinha, and Y. Cao, "Compact modeling of carbon nanotube transistor for early stage process-design exploration," Proceedings of the 2007 international symposium on Low power electronics and design, Portland, OR, USA, 2007.
[10] A. Balijepalli, S. Sinha, and Y. Cao. PTM CNT-FET model. Available: http://ptm.asu.edu/cnt-fet/veriloga.va
[11] S. Jahn and H. Parruitte. Verilog-AMS interface Available: http://qucs.sourceforge.net/docs/verilog.pdf
[12] L. Lemaitre. ADMS project [Online]. Available: http://sourceforge.net/projects/mot-adms/