Developing a Common Compact Modeling Platform for Model Developers and Users

Lining Zhang, Muthupandian Cheralathan, Aixi Zhang, Salahuddin Raju and Mansun Chan

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, lnzhang@ieee.org

ABSTRACT

This paper described the philosophy of the *i*-MOS (interactive Moeling and Online Simulation) platfrom which is constructed to provide easy access to all the necessary tools for developing a circuit model. Through the platform, model developers can follow a well described procedure to author their models using *i*-MOS provided tools. Users can also evaluate the most up-to-date models using a simulation engine provided by *i*-MOS without downloading and compiling. To facilitate compact modeling related activities, a number of model specific functions including parameter extraction, statistical distribution analysis and reliability simulation are planned. The current status and the future directions of *i*-MOS will be described in this paper.

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

1 THE COMPACT MODELING INFRASTRUCTURE

Compact model is a very important link to bring both traditional and emerging devices to applications. With the rapid development of technology in recent year, the diversity of emerging devices has increased significantly. However, the infra-structure for developing compact models has not catched up the pace of technology development, which leads to delay in the adoption of some of the emerging devices.

The difficulty of developing a practical compact model arises from the availability of a full-suite of developemnt tools and the compatibility among these tools. To develop a model, a set of equations are first formulated and a language to describe these equations is needed. The most common tool at this stage is Matlab [1]. Recently, Verilog-A [2] has emerged as the standard language to describe the characteristics of a model. Once the model equations are finalized, the model need to be incorporated into a circuit simulator for evaluation. Another tool, which is a simulation engine, is required for this purpose.

In the early day, circuit simulators are usually written in C-language (some even with FORTRAN) and to include a model into a simulator, the model equations need to be be translated to C-codes and compiled together with the simulator as a single piece of software. The wide acceptance of Verilog-A as the modeling language has made the inclusion a model to a simulator simpler after most commerical simulators like HSPICE [3] and SPECTRE [4] have included a Verilog-A compiler. Verilog-A codes can be distributed to interested parties to perform simulations if the users can afford the price of the commercial simulators. Otherwise, users need to rely on scattered opensource Verilog-A compilers (like the Automatic Device Model Synthesizer (ADMS) [5]) and opensource simulation engines (like NGSPICE [6]).

After a user received the Verilog-A code and compile it with a simulator, the model has to be calibrated before it is used to predict circuit performance. To calibrate a model, the user need to first understand the model by collecting documents that describe the model. While some developers put a lot of efforts to document they models, many just rely on published technical papers to release the necessary information. Model calibration also requires some physical data (sometimes from well design physical structures mainly for calibration) or numerical data from more fundament physics tools (such as atomistic simulators), together with the parameter set. For simple academic models that only deal with ideal structures, the parameter sets mainly consists of structural description and is relatively easy to obtain. However, for models to match experimental results, a set of model parameters or a software to extract the parameters is required.

After collecting the model equations, a circuit simulation platform, documentations, data for calibration and device parameter set, a simple simulation can be completed. Post-simulation sometimes are necessary to address many production related issues that circuit simulations can address, but need post-simulation support. Examples are reliability and statistical distribution. Separate tools are necessary to organize the simulations and model data to achieve these objectives.

The scattering of tools and information has made the process of developing a truly usable compact model (in contrast to some models are only for publications) very inconvenient. A number of projects like the Nano-Engineered Electronic Device Simulation (NEEDS) [7] and interactive Modeling and Online Simulation (*i*-MOS) [8] platforms have started to address the inefficient infrastructure for compact modeling. The *i*-MOS project will be described in in this work.

2 THE *i*-MOS SIMULATION INTERFACE

The details simulation interface of *i*-MOS can be found from the *i*-MOS user manual available in [8]. Some key concepts will be summarized here.

2.1 Model Characteristics

The model page allows users to study the characteritics of a particular model or device. When a user click onto the model page, a number of available devices will be shown as displayed in Figure 1 and the user can select the model to be evalutated.

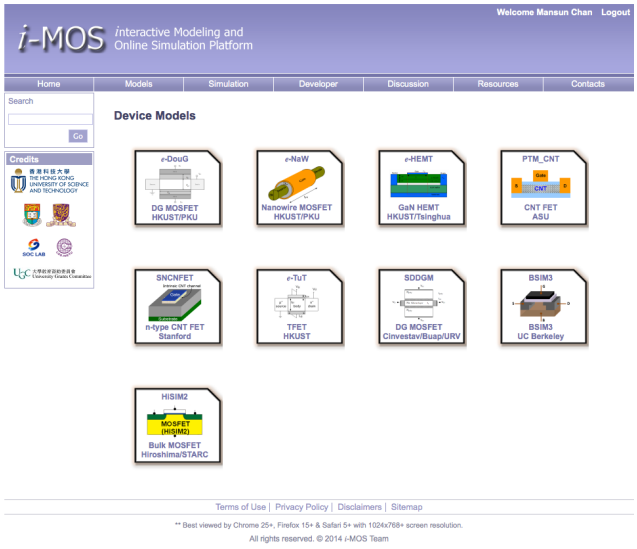

Figure 1: The model page showning the available models

Once a model is selected, the description of the model as provided by the authors will appear together with the information of the developing team and contacts. If the authors are willing to release the Verilog-A source codes, the link to the codes will also be provided here. On top of the description page, a series of tags will appear and the user just need to follow the sequence of the tag to obtain the model output.

Following the model description is the parameter page and the user can manually enter the parameters or upload parameters obtain elsewhere to the page. *i*-MOS also provide a collection of parameters (mainly for industrial models) from the public domain to reduce the burden to collect parameters. For future version of *i*-MOS, a more interactive interface will be provide for users to tune the parameters and observe the output at the same time.

After selecting the values of the parameters, users can enter the biases with the first running variable and second running variable. It is followed by selecting the output variables to display. The available output variables depend on the completeness of the model. For example, if the model only contains current-voltage characteristics, charge and capacitance will not be displayed.

In addition to user defined biasing, *i*-MOS also provides biasing conditions for some standard tests like the Gummel Symmetry Test [9], AC symmetry test and Capacitance tests. When one of these tests is selected, the output variables will be selected automatically.

After all the steps described are performed, the user is ready to run the simulation. Once the instruction to execute the simulation is entered by the users, the web-brower will format the data and pass it to the server to execute the simulation. The simulation time is monitor by the server and the user has the option to abort the simulation if excessive idle period is observed. If the simulation returns correct results, the output will be formated and displayed on the browser.

Users can compared the simulation results with their own data. To do that, the user need to upload the custom data using the "upload custom data option". The data will be displayed together with the simulation results when the "show customer data" box is checked as shown in Figure 2.

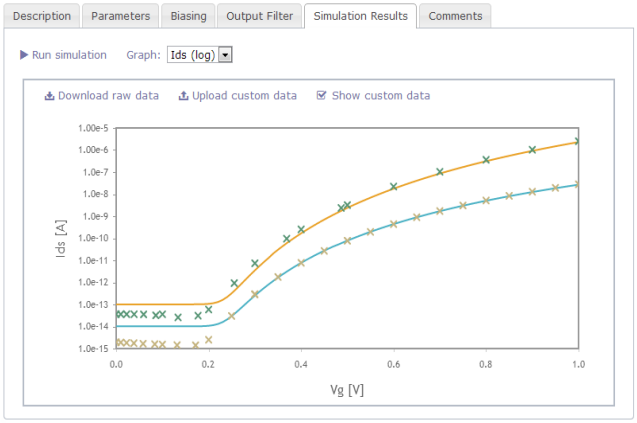

Figure 2: Output of simulation together with custom data from users

After evaluating a model, users can also provide comments and feedback on the model performance as a reference for other users as well as model developer. This is an important step for the modeling community to work together to understand the needs of users to further refine the model.

2.2 Circuit Simulation

i-MOS uses NGSPICE [6] as the simulation engine and linked the available models using dynamic library. To input a netlist, users can use the Raw Mode which input files following the NGSPICE input format can be directly pass to the server. To help users to include the models with their own customized parameter set, a simplified input mode called the Netlist input mode has been implemented. As shown in Figure 3, the Netlist mode consists of 4 different sections that need to be completed before a

simulation can be executed. They are the Circuit, Source Definition, Analysis and Output Variables.

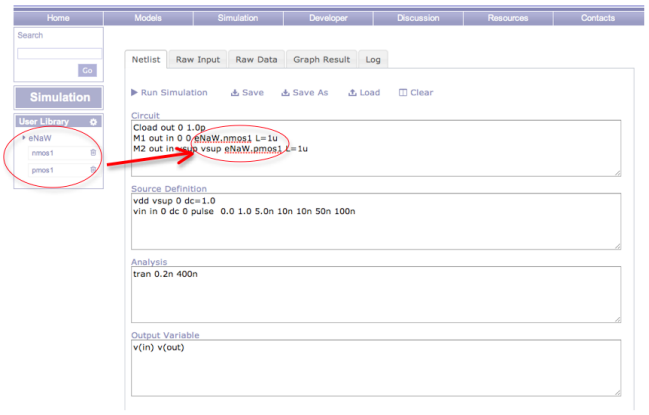

Figure 3: The netlist page showing the 4 sections: (1) circuit; (2) source definition; (3) Analysis; and (4) Output variables

The Circuit section is where the user enters the netlist following the format as specified in the NGSPICE manual [6]. To use models constructed in the model page of *i*-MOS, a user library need to be constructed and the process to construct a user library will be described in the next section. The model in the user library is specified by the name of the model as defined by the developer together the name of the parameter set defined by the user when he/she save the parameters to the user library. To associate the parameter set with an element in the netlist, the format of <model name>.<parameter set name> should be used in the instant description. For example, consider a nanowire MOSFET (eNaW) with two sets of parameters stored by the users, namely nmos1 and pmos1. To use the nmos1 model parameters, the eNaW.nmos1 description is entered in the netlist. In addition, all the instant parameters should be included here. The process is still a bit crumbersome at this moment, but we are going to implement a drag-and-drop function in the near future.

The Source Definition section allows users to enter the different power supplies or signals applied to the circuit. The sources can be fixed DC sources, ramped DC sources, time dependent sources or AC sources.

In the Analysis section, the type of simulation is specified. The available type of simulations depends on the model implementation. For example, if an active device model only has I-V equations implemented, it is normally only accurate for DC simulation and will not be able to produce the accurate delay behavior in TRAN simulation.

The Output Variable section defines the output data to be collected and plotted. Putting more than one variable on the same line will include them in the same table and plot on the same graph. Putting variables on different lines will place them on separate table and separate graphs.

After completing the netlist, the circuit can be submitted directly to the server by using the Run Simulation function on the top of the page. The results will appear in the Raw Data page or Graph Result page as shown in Figure 4. Actual data in the Raw Data page can be download and analyze using other software. For more experienced users, the circuit can be converted to the actual input file in the Raw Data page for checking and editing before submitting to the server.

Figure 4: Graphic output with an input square wave and output of a circuit super imposed on each other

2.3 The User Library

The user library provides the link between single device simulation and circuit simulation. After a user fine tuned the parameters of a model to match the need for a specific simulation, the parameter set can be saved into the user-defined user library by using the "include in user library" option. A new entry under the model that the parameter set is created for will appeared in the user library with the user defined name. It should be noted that parameters governing the device behavior are categorized into instant parameters and model parameters. Only model parameters are added to the user library and instant parameters have to be entered directly to the netlist.

Some simple functions to manage the user library are provided like loading and saving custom libraries. Pressing any model in the model library provides a shortcut to jump to that model page. Pressing any parameter set below a model loads that parameter set to the parameter-editing page. By doing so, the current parameter set on the parameter-editing page will be overwritten.

3 SUBMITTING MODELS TO *i*-MOS

i-MOS supports Verilog-A [2] as the modeling language for model implementation. It is the objective of *i*-MOS to provide a development server with a easy to use interface for model developer to compose and edit their model. However, it requires a well-developed opersource Verilog-A compiler. The most readily available opensource

Verilog-A compiler is ADMS [5] but it has limitations and only a subset of Verilog-A syntax is supported. Currently the process is only semi-automatic with some tasks to be performed internally by the *i*-MOS team.

To submit a model, a developer can enter the developer's page. The first page in the developer page shows he models submitted to *i*-MOS and their status. We hope to complete any well-developed model integration within one month after submission before the automation of this process is achieved. Developers can select to update the models they have submitted or submit new models. If a developer is going to submit a new model, he/she has to agree on the terms and conditions set by *i*-MOS. Before developers actually submit their model, the information listed in Table 1 should be readily available.

Required Information	Format
Basic information including model name, author list and author contact	Direct entry
Description of the model	Direct entry
List of reference documents	Direct entry
Graphical Model Symbol	Jpg/pdf
Model source code coded in Verilog-A (or C-codes with the correct interface to NGSPICE)	in zipped file
List of parameters and their categorization	.xls/.xlsx
List of available output variables	.xls/.xlsx

Table 1: Required information from developers before they upload their model to *i*-MOS.

The first page in the developer upload process asks for information related to the model, including model name, author list and contact, model description, reference documents etc. This information will appear in the model description page and is the most important channel to advertize the model to the users. Then the developer can upload the verilog-A code in zipped format. A few more question related to the status of model teting will be asked to reduce the workload of the team. In addition to model code, the developer also need to submit a list of parameters and how they are arranged in an Excel file. This information is going to define the appearance of the parameter page. Furthermore, the developer will also be asked to submit the available output variables again in Excel format to let users examine the state variables available in the model. These output variables will be made available for users to plot in the output filter page.

4 PLANNED FUNCTIONS

The current *i*-MOS platfrom has provided the basic functions to produce simple characteristics of devices as well as allowing the interconnect of devices to form netlist for circuit simulation. But compared with the initial promise as a simple to use communication platform for model developers and users with all necessary tools available, there is still a long way to go. Three tracks will be pursued in the future development: (1) addtion of contents under the existing format; (2) enhancing usability of the service provided; and (3) developing entirely new functionality.

For addition of contents, new models will be continuously added to *i*-MOS, more extensive collection of experimental data and atomistic simulation data availabe in the public domain will be provided and a library for model parameters will be setup. Educational contents from workshop presentation, class exercises and self-test questionaires have also been included in the planning. We will need the help of the compact modeling community to work together to enrich the content of the platform.

To enhance the usability of the service, a graphical user interface for netlist construction is the most important task followed by the automated authoring environment for model developers. But both require significant investment in time and manpower.

Finally, there are other services with high demand from the modeling community such as parameter extract algorithm, statistical parameter variation, reliability simulation and circuit/device co-design environment. These functions are potential extensions of the *i*-MOS platform and will be investigated in the near future.

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 current undergraduate programmers who perform most of the coding of the *i*-MOS platform. They are Chun Ming Au, Richeng Huang and Sheldon Meng

REFERENCES

[1] http://www.mathworks.com/help/matlab/
[2] Verilog-AMS Laguage Ref. Manual, Accellera, 2009
[3] http://www.synopsys.com/tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx
[4] http://www.cadence.com/products/cic/spectre_circuit/pages/default.aspx
[5] L. Lemaitre, C. McAndrew, and S. Hamm, "ADMS-automatic device model synthesizer", IEEE CICC 2002, pp. 27-30
[6] http://ngspice.sourceforge.net
[7] http://nanohub.org/groups/needs
[8] http://i-mos.org
[9] H. K. Gummel, Model Implementation and Verification Facilities for PSIM, Oct. 29, 1990. AT&T Bell Laboratories Technical Note