Design-Adaptive Device Modeling in Model Compiler for Efficient and Accurate Circuit Simulation

Bo Wan
Cadence Design Systems
bowan@cadence.com

Emrah Acar, Sani Nassif
IBM Austin Research Center
{emrah,nassif}@us.ibm.com

C. –J. Richard Shi
Univ. of Washington, Dept. of EE
cjshi@ee.washington.edu

ABSTRACT

In this paper, a new concept of design-adaptive device modeling is introduced and an innovative method using adaptive device models to speed up circuit simulation is presented. In this method, we utilize the design specific information to generate automatically adaptive device models, which implies partial information on the more generic device model. During simulation, available adaptive device models are substituted automatically for more computationally expensive general device models. The proposed method is model compiler based and has been implemented in our compact device model compiler – MCAST, and targeted and tested for three simulators: the open source circuit simulator SPICE3, Cadence's SPECTREv4.46 and an industry in-house simulator. Experimental results show that the model evaluation time of a typical design adaptive device model is up to 10x faster compared to manually coded built-in general device model, and the overall simulation time with design-adaptive device models can be up to 8x faster with respect to simulation with general device models while achieving the same accuracy.

I. INTRODUCTION

It is becoming increasingly difficult to implement a new device model in circuit simulators manually. Table 1 attempts to quantify the levels of difficulties in implementing a new industry-standard MOSFET device model.

TABLE 1. MOSFET MODEL IMPLEMENTATIONS IN UCB SPICE3F5.

Number	Level 1	Level 3	BSIM3	BSIM4	BSIMSOI
If statements	501	609	822	1495	1757
Parameters in codes	32	38	399	610	615
Intermediate Variables	56	56	430	811	972
Total lines of code	7,673	8,634	12,348	20,113	20,664

Manual implementation of a new device model has become a big burden, even for a team of experts with extensive modeling and coding experience. Not only the implementations, but also the practices of debugging, testing, profiling and maintaining, are cumbersome and time-taking. In many aspects, the practice of manual device model implementation amounts to longer qualification times. The advent of more complex and challenging device models for nanotechnology and multi-physics technology systems could make this situation only worse.

Compact device model compiler has emerged as a solution to address this problem as suggested in [1]. In a nutshell, a compact model compiler is a design automation tool that takes a high-level behavioral description (in VHDL-AMS or Verilog-AMS languages) of a device model as its input, and applies several processes to automatically generate the implementation source code for simulating a circuit with that the requested device model. The generated source codes can be compiled and linked into a target circuit simulator. With the use of a model compiler, the processes for model implementation, enhancement and qualification are greatly shortened. Moreover, the device model can be easily maintained, modified and distributed. The automated model-compiling process also greatly enhances the ability of circuit designers to manipulate the physical operation and topology of the device. With a model compiler link, circuit designers no longer have to wait one or two years for the new device model to become supported in a commercial circuit simulator.

Model compiler may also help to expedite modeling efforts for better industrial standards. In device modeling industry, having a standard base model for a device is very difficult. Implementations in commercial and freeware simulators may differ greatly although the underlying equations are the same. Using the same device model but various implementations, different simulators may give different simulation results. Without a standard model, it is hard to verify which one is correct or which one is trustable. A device model is only completely defined by its implementation in a target simulator, and therefore it may have different definitions in different simulators. Model compiler brings the possibility to implement a bulletproof standard model code for different simulators, and the high-level models feeding into model compiler are easier to standardize. Today, public device models in Verilog-AMS and VHDL-AMS are already starting to appear on the web in public domain [2].

Several related works about model compilers have been reported previously [3]-[8]. Some of these works discussed implementation of very complex industrial device models such as BSIM3, BSIMSOI etc. and employed code-optimization techniques to improve the efficiency of the generated device model codes. However, none of these works has taken users' design information into account when implementing the device model. Their optimizations, if any, mainly focus on the implementation of a general device model.

The general practice of model compiler is to define a device model with instance parameters as independent variables. A general topology of the device is also assumed. In this approach, the device model is developed and implemented in the most

general way to support all possible uses of the device model in the end-users netlist.

In this paper, we propose to extend this traditional approach with a new feature that can generate design-specific adaptive device models to speed up the upcoming simulation tasks. Mainly, we utilize two types of user design information in a model compiler to generate adaptive device models for circuit simulation. The first is the pre-specified model and instance parameters, and the latter is the special device topology.

Pre-specified model and instance parameters can be used to simplify the complexity of the model by removal of an independent variable. In the case of designs with many devices sharing the same model and instance parameters, it is worth generating adaptive device model for such devices to speed up model evaluation time and hence the simulation times. A specific topology of device can also be used to simplify the complexity of the device model. For example, for a MOSFET device model whose bulk and source terminals are identical, the back-gate effects are not of interest, and the number of device model equations may be deferred. Furthermore, a specific topology of the device makes it possible to generate low dimensional (2-D or 1-D) table lookup model with very high accuracy. Table lookup is an attractive way to speed up device model evaluation. Previously, table lookup approaches have been applied to MOSFET transistor models [10][11][12][13]. However, all of mentioned efforts suffer from the high dimensionality, since a general MOSFET device is four-terminal and usually it is 3-D tables are generated. SOI device has even more terminals. Although some methods employ 2-D tables with some correction tables [12], accuracy is compromised to meet the massive memory requirements. In our adaptive device models with pre-specified topology, table lookup models may be chosen as 2-D or even 1-D, and accuracy can be easily improved by increasing the size of the tables. This is very important for simulating analog circuits since they often require higher accurate models.

To our best knowledge, no work has been reported on design-adaptive device modeling to gain speed-up in circuit simulations. In this paper, we demonstrate that adaptive device modeling can be easily implemented in a model compiler. It cannot be done manually.

II. DESIGN-ADAPTIVE DEVICE MODELS

Traditionally, compact device models are implemented as general device models. The general device model will be instantiated during simulation, i.e., model, instance parameters and connection topology will be assigned. Then the devices are iteratively evaluated and corresponding model evaluation results are loaded into the residual vector and the Jacobian matrix in the target simulator.

The device loading/evaluation always contributes a major, often dominant, part of the overall simulation time. It is always a huge challenge to reduce or simplify this time consuming part in the circuit simulation. The basic idea of our design-adaptive device modeling method is as follows: adapting general device model into more specific models with partial information obtained from user's netlist (or design). Using these adaptive models instead of general device models would greatly reduce the overall evaluation time. Below, we first introduce the concept.

Design-adaptive device model is a more *specific* model of a device. Compared to a *general* model, an adaptive device model has one or more of the following characteristics:

- Fixed model parameters
- Fixed some or all instance parameters
- Specific topology

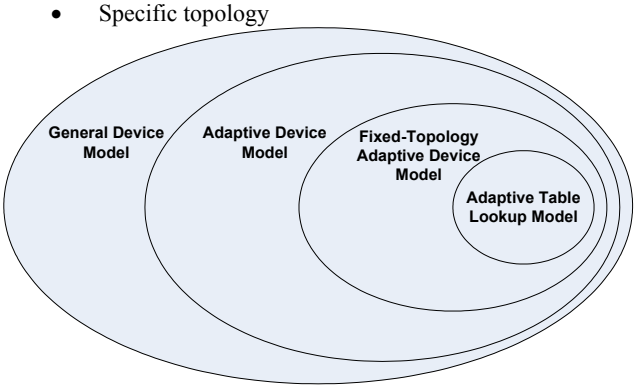

Figure 1. Device model set diagram.

Figure 1 shows the set diagram for several device model types. The class of general device model is a superset of adaptive device model which itself covers adaptive fixed-topology device model. Adaptive table lookup model can be considered as a subset of fixed-topology adaptive model.

As an example, MOSFET device model is carefully analyzed and some adaptive device model types are illustrated in Figure 2. Type (e) and (f) are fixed-topology adaptive device models.

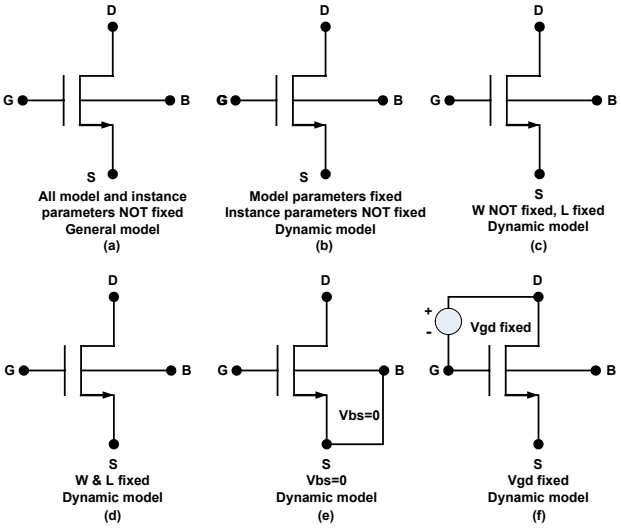

Figure 2. MOSFET design adaptive device models.

One should note that some of these adaptive device models could be combined together as a new and better adaptive device model (such as type (d) plus type (e)).

Obviously, the most specific adaptive device model is one with fixed parameters and has a specific topology. For the devices using this model, the model parameters of the adaptive models are

not needed in the netlist, since they can be utilized in the model code by the model compiler.

In adaptive device models, all model parameters and some or even all instance parameters are fixed, the calculations related to these parameters can be pre-calculated as constants in model compiler. By using Abstract-Syntax-Tree (AST) based constant propagation optimization techniques described in [4], the complexity of the adaptive device model can be greatly reduced. Moreover, for the fixed topology like type (e) in Fig. 2 (Vbs=0), the back-gate effect no longer exists and the current equations for the connected terminals can be collapsed in the model description.

Adaptive device models can be easily identified and constructed based on the circuit netlist information especially for digital circuits. For example, in many digital designs, almost all transistors share a fixed L (length) parameter and the topology of type (e) (Vbs=0) is very common. In some standard cell library based digital designs, (include inv, nand2, nor2, aoi22, oai22, etc.) Although they may have thousands of transistors, they may have just two adaptive device models: one for NMOS with fixed model, instance parameters and Vbs=0 and one for PMOS with fixed model, instance parameters and Vbs=0.

III. ADAPTIVE LOW-DIMENSIONAL TABLE LOOKUP MODELS

In a compact device model, even in our reduced adaptive device model, the model evaluation may be very expensive. The basic idea of our adaptive low-dimensional table lookup method is to replace the computation-intensive blocks by two-dimension or one-dimension tables to save the evaluation time.

Figure 3 shows some MOSFET topologies that can be used to generate low-dimensional table lookup models.

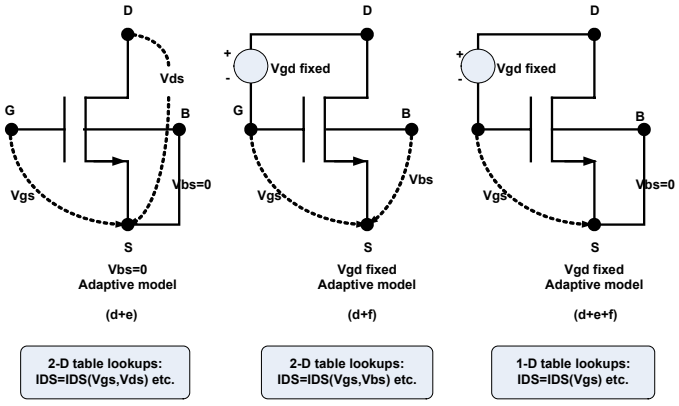

Figure 3. Topology-aware table lookup model.

After discovering the fixed model and instance parameters and fixed topologies from the netlist, the model evaluation process can be simplified as 1-D or 2-D tables. The model output variables such as branch currents, derivatives, charges and right-hand-sides, are fortunately functions of one or two terminal voltages. Even the range of the terminal voltages such as Vgs, Vds, etc. can be obtained from the netlist. It is very easy to build 2-D or 1-D tables for these adaptive device models in an automatic manner.

In MCAST, uniformly separated tables are adopted to hold model evaluation data. It is efficient to locate points surrounding the interpolation point and it works very well even for high accuracy requirement for analog circuits.

These low-dimension tables are built by MCAST and will be exported with adaptive table lookup models. Some utility routines are provided as well. Target simulators will load data tables when setting up these models and call these routines to locate corresponding table lookup models and interpolate values during simulation.

In target simulators, computational efficiency is improved by using simple bilinear interpolation. It is computationally efficient and accurate enough in our process.

IV. DESIGN-ADAPTIVE DEVICE MODELING FLOW IN MCAST

The proposed design adaptive device modeling method has been automated in MCAST. Figure 4 shows its flow.

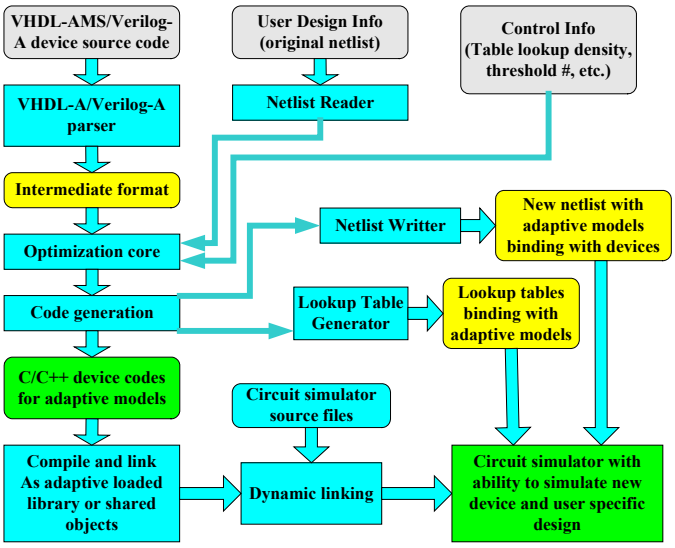

Figure 4. Adaptive device modeling in model compiler MCAST.

Our device model compiler is composed of a parser, an internal data representation framework, a code optimization core and a code generator. Through the model compiler, high-level description for the device model will be translated into C/C++ implementation codes, which can be compiled and linked into target simulator.

In MCAST, within the model compiler architecture, a netlist reader is also deployed to explore user design information (circuit netlist) along with some optional user control information (table lookup density, etc.) The optimization core then uses this information to generate the implementation code for the adaptive device models.

Code generator will output two other sets of information along with the C/C++ device codes. The first is an updated netlist with device instances bound to corresponding adaptive device models.

The simulator will use this new netlist to simulate the circuit. The second output is table lookup package, including adaptive table lookup models, table lookup data library and utility procedures, etc. The simulator will load these tables during the device setup time and look them up during simulation.

Several optimization techniques including adaptive device modeling are implemented in the optimization core.

V. IMPORTING ADAPTIVE DEVICE MODELS INTO TARGET SIMULATORS

MCAST imports a designer's netlist and generates several adaptive device models. These adaptive device models and/or a general model need to be imported into the target circuit simulator. Unfortunately, the registrations of these new models into different simulators are different. In the following, we present the implementation of multiple adaptive device models into SPICE3f5 and SPECTREv4.46.

A. SPICE3f5

SPICE has a 26-letter limitation for new devices. It is impossible to assign a letter to every adaptive device model. Instead, all of the automatic models generated by MCAST use the same model letter "N". MCAST will need to generate new netlist file to specify the multiple adaptive device models and to bind corresponding devices with these models. Figure 5 shows the comparison between the old netlist and the new netlist with dynamic binding. In the new netlist, the model "type" will be used to differentiate auto models: ADM1 means adaptive device model 1. A new input handling subroutine inp2n.c was written in INP directory to accept dynamic terminals and dynamic instance parameters for all adaptive device models under one model name.

Several Perl and Shell scripts were written to automatically load, compile and link multiple adaptive device models. Since the adaptive device models are only useful for this design, user has the option to start a script to unload them after the simulation.

B. SPECTRE

Unlike SPICE3, SPECTRE has a mechanism called compiled-model interface (CMI). CMI allows models compiled as shared objects to be dynamically loaded and installed at run time. New device model source code can be decoupled from the SPECTRE executable. In this way, it is much easier to import multiple adaptive device models without re-building the SPECTRE executable. For other circuit simulators with dynamic loaded libraries, the loading and unloading of multiple adaptive device models can be done similarly.

The loadable models are particularly important considering confidential intellectual property.

VI. EXPERIMENTAL RESULTS

To demonstrate the effect of adaptive device modeling, several MOSFET models, including level 1, level 3, BSIM3, BSIM4 and BSIMSOI, have been implemented in MCAST, linked and built in the open source circuit simulator, Berkeley's SPICE3f5, Cadence's SPECTREv4.46, and an industry in-house simulator, and compared with human optimized codes (existing built-in device model codes in SPICE3f5). Some notions are used in the comparisons: "Built-in" model is the one manually implemented, "Auto w/ Opt" model is the one automatically generated by MCAST with optimizations except adaptive device modeling, "ADM (d+e)" model is the one automatically generated by MCAST with optimizations, including adaptive device modeling, and "ADM (d+e)+Table" is the adaptive table lookup model. Here (d+e) represents the type of the adaptive device model and it means fixed parameters (d) with Vbs=0 topology. Type (d+e) is selected since it is the most popular type of adaptive device models found in MOSFET. The accuracy and efficiency of the generated adaptive device models are demonstrated by the simulation results.

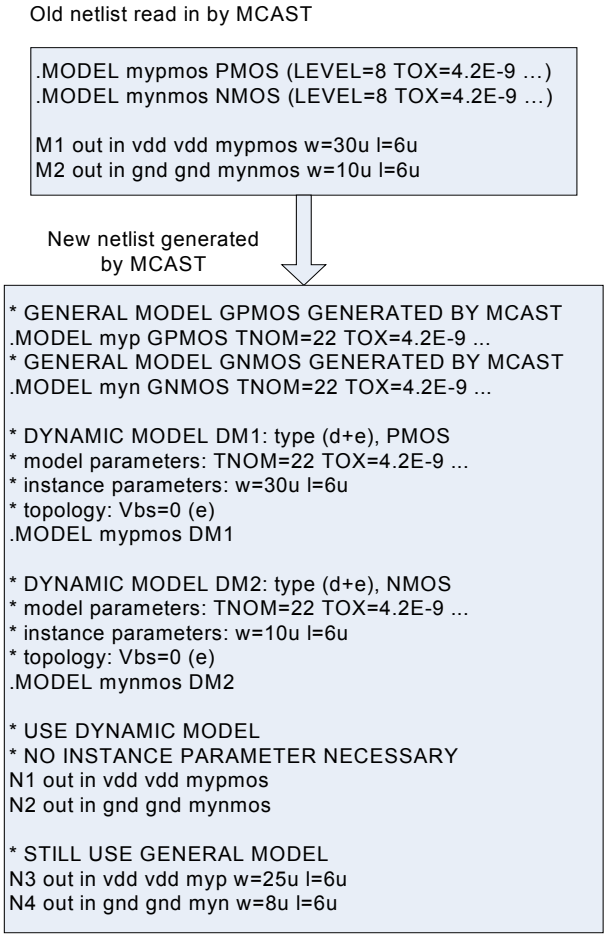

Figure 5. New netlist for simulating models with MCAST.

A. Accuracy

The automatic generated adaptive device models from MCAST are inherently very accurate. Figure 6 shows the comparison of the transient analysis of a simple inverter. The automatic generated adaptive device models yield exactly the same results as manually implemented built-in model in SPICE3f5: the absolute errors are less than 5e-6 compared to the built-in model.

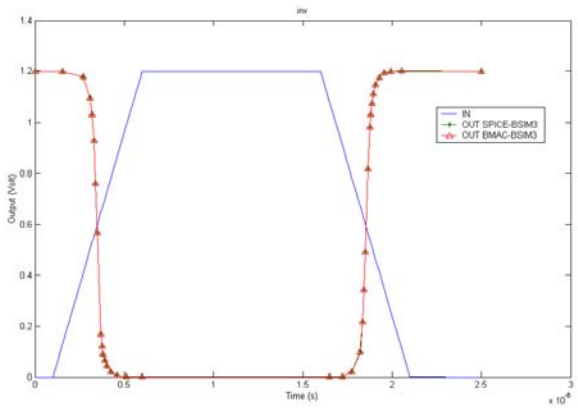

Fig. 6. Accuracy comparison: transient analysis of an inverter.

Figure 7 shows the transient simulation results of another benchmark circuits – VCO. The results with adaptive device models match well with that of the built-in models.

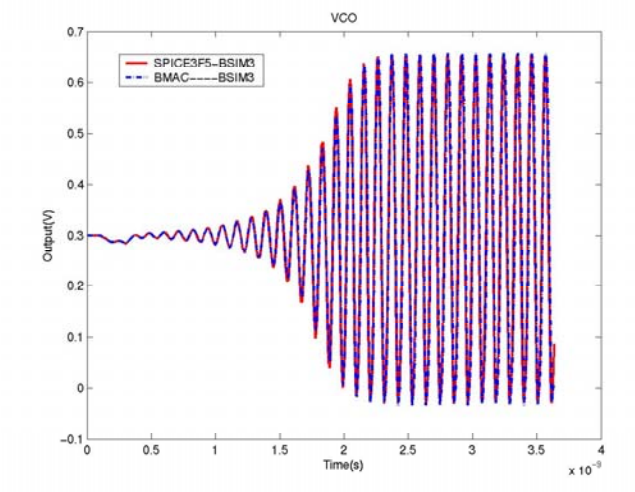

Fig. 7 Accuracy comparison: transient analysis.

The accuracy of the adaptive table lookup model can be easily improved by increasing the table sizes. In our practice, 4000 points are good enough for 2-D table lookup to get the same accuracy as the built-in model.

B. Adaptive device modeling speedup

Several analog and digital circuits are selected as benchmark circuits to test the speedup effects of adaptive device modeling. Their characteristics are summarized in Table 3. From this table, one should note that adaptive device models exist in both analog and digital circuits. In some circuits, adaptive device models repeat themselves hundred of thousand of times. For example, in a standard cell library based digital design (ACCAS) two adaptive device models cover all of the 1038 transistors.

TABLE 3. BENCHMARK CIRCUIT CHARACTERISTICS
ADM – ADAPTIVE DEVICE MODEL
ATLM – ADAPTIVE TABLE LOOKUP MODEL

Index	Circuit	#MOS	# ADM	#ADTLM
1	One-shot	22	4	2
2	VCO	10	2	2
3	Power AMP	4	0	2
4	Ring Oscillator	12	0	2
5	Boeing Comparator	38	2	2
6	Complex Cell	30	1	4
7	INV	2	0	2
8	INV Chain	8	0	2
9	NAND2	4	0	2
10	NOR2	4	0	2
11	AOI22	8	0	2
12	OAI22	16	0	2
13	ACCAS	1038	0	2
14	DFF	24	0	2
15	SRAM	6910	9	4
16	26-bit Adder	4274	18	6
17	13-bit Multiplier	9545	24	16
18	12-bit Divider	3081	22	8

Figure 8 shows a comparison among different model implementations, including adaptive device model, built-in model and auto model with normal optimizations in MCAST, of different types of models, such as level 1, level 3, BSIM3, BSIM4 and BSIMSOI. The experiment is circuit-independent and only the model evaluation times are normalized and compared. In pure comparison of the evaluation costs of the different models, the adaptive device models are 1.6x-2.69x times faster than the manually coded built-in models and 1.6x-4x times faster than the normal optimized auto models. The adaptive table lookup model is at least three times faster than conventional analytical models.

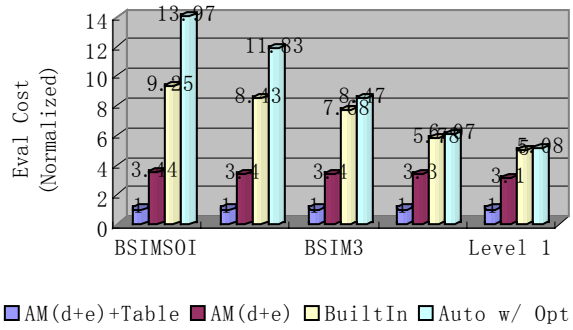

Fig. 8 Normalized model evaluation cost comparison.

From Figure 8, we can also observe that 1) the simpler the model, the less overhead there is for the automatic generated model. For example, the auto model with optimizations is just 2%-3% slower than built-in model for MOSFET level 1 and level 3 model and 2) the more complex the device model, the more speedup there is using adaptive device model. In terms of device evaluation cost, the speedup for the BSIMSOI adaptive device model is 2.69x and

for adaptive table lookup model it could go up to 9.25x faster than the manually coded built-in model.

We also compared the performances in transient analysis. Nine analog and digital benchmark circuits, including power amplifier, 12-bit divider, etc., were used to demonstrate the speedup effects of the adaptive device modeling of the MOSFET model of BSIM3 versus the built-in model (Fig. 9). For a fair comparison, fixed time-step is used for both models and only transient CPU times (device setup time not included) are compared. The performance of the built-in model is normalized to one. For most of the benchmark circuits, the speedup using adaptive models without table lookup is more than two times. For analog circuit, using adaptive models with table lookup can speed up simulation by a factor of 4. For digital circuits where many devices are in uniform size and have a fixed topology such as bodies connected to sources can be found, the simulation speedup using design adaptive table lookup models is at least 6x-8x.

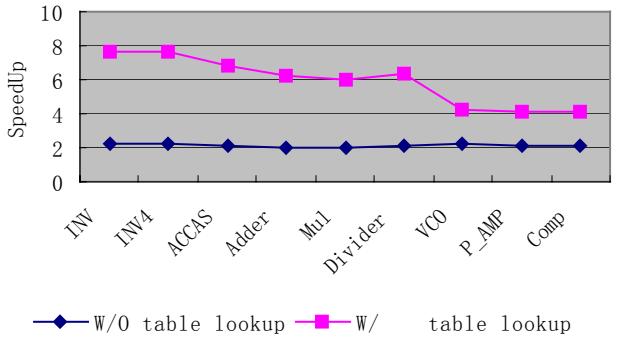

Figure 9. Normalized speedup of the adaptive device models compared to the built-in model over nine benchmark circuits.

C. *Overheads*

Significant speedup in simulation time can be obtained using our proposed design adaptive modeling method. However the design adaptive modeling and table lookup methods introduce some overhead, namely requires some time to generate and load ADM models. However, after the first model generation, the models stored in the memory can be used efficiently in many tasks demanding repetitive simulations such as cell characterization, optimization, noise and timing analyses. Furthermore, the generated device models can be cached and are easily reusable in an embedded circuit simulator via an application programming interface. Considering the potential use of our proposed technique in a higher level design automation task, the device setup time (additional time needed to load multiple tables in our method) is ignored in our comparisons of transient analysis.

VII. CONCLUSION

We have introduced the concept of design-adaptive device modeling and presented a method for generating adaptive device models to gain significant speedup in circuit simulation tasks. Adaptive device models are design specific but much more efficient to evaluate and can be dynamically loaded and unloaded into target circuit simulator. The proposed method facilitates the design information from the simulated netlist to generate adaptive device models. Our prototype implementation can work with compact device and behavioral models described by high-level languages VHDL-AMS and Verilog-AMS and generate adaptive device model implementation codes automatically.

The proposed method has been implemented in our compact model compiler MCAST and targeted for three circuit simulators: the open source SPICE3f5, Cadence's SPECTREv4.46, and an industry in-house simulator. Experimental results on a set of test circuits have demonstrated that the generated adaptive device models are very accurate with the error limited to numerical noise range, and the speedup for circuit simulation can be as much as 8x faster than simulation with human optimized built-in models.

VIII. REFERENCES

[1] K. Kundert, "Automatic Model Compilation – An Idea Whose Time Has Come", The Designer's Guide, May 2002. http://www.designers-guide.com/Opinion/modcomp.pdf

[2] Silvaco,http://silvaco.com/cgibin/news/showItem/2004_03_02_01.html, March 2004.

[3] L. Lemaitre, C. McAndrew and S. Hamm, "ADMS-Automatic Device Model Synthesizer", Proc. IEEE Custom Integrated Circuits Conference, pp. 27-30, May 2002.

[4] B. Wan, B. P. Hu, L. Zhou and C. –J. R. Shi, "MCAST: An Abstract-Syntax-Tree based Model Compiler for Circuit Simulation", Proc. IEEE Custom Integrated Circuit Conference, pp. 249-252, Sept. 2003.

[5] Tiburon Design Automation, http://www.tiburon-da.com/

[6] S. Liu, K. C. Hsu, P. Subramaniam, "AAMIT-ADVICE Modeling Interface Tool", Proc. IEEE Custom Integrated Circuits Conference, pp. 6.6/1-6.6/4, May 1988.

[7] A. T. Yang, and S. M. Kang, "iSMILE: A Novel Circuit Simulation Program with Emphasis on New Device Model Development", Proc. IEEE 26th Design Automation Conference, pp. 630-633, June 1989.

[8] R. V. H. Booth, "An Extensible Compact Model Description Language and Compiler", Proc. IEEE/ACM BMAS, pp. 39-44, Oct. 2001.

[9] H. Carter, "Modeling and Simulating Semiconductor Devices Using VHDL-AMS", Proc. IEEE/ACM BMAS, pp. 22-27, Oct. 2000.

[10] A. Rofougaran and A. A. Abidi, "A Table Lookup FET Model for Accurate Analog Circuit Simulation," IEEE Trans. Computer-Aided Design, vol. 12, pp. 324-335, Feb. 1993.

[11] M.G. Graham and J. J. Paulos, "Interpolation of MOSFET Table Data In Width, Length, and Temperature", IEEE Trans. Computer-Aided Design, vol. 12, pp. 1880-1884, Dec. 1993.

[12] T. Shima, T. Sugawara, S. Moriyama and H. Yamada, "Three-Dimensional Table Look-Up MOSFET Model for Precise Circuit Simulation", IEEE J. Solid-State Circuits CS-17, 3, pp. 449-454, 1982.

[13] T. Shima, H. Yamada and R. L. M. Dang, "Table Look-Up MOSFET Modeling System Using a 2-D Device Simulator and Monotonic Piecewise Cubic Interpolation", IEEE Trans. Computer-Aided Design CAD-2, 2, pp. 121-126, 1983