Hierarchical Multi-Dimensional Table Lookup for Model Compiler based Circuit Simulation*

Bo Wan and C.-J. Richard Shi
Department of Electrical Engineering, University of Washington
{wanbo,cjshi}@ee.Washington.edu

Abstract— In this paper, a systematic method for automatically generating hierarchical multi-dimensional table lookup models for compact device and behavioral models with any number of terminals is presented. The method is based on an Abstract Syntax Tree representation of analytic equations. Expensive part of the computations represented by abstract syntax trees are identified and replaced by two-dimensional table lookup models. An error-control based optimization algorithm is developed to generate table lookup models with the minimal amount of table data for a given accuracy requirement. The proposed method has been implemented in the model compiler MCAST and the circuit simulator SPICE3. Experimental results show that, compared to non-optimized compilation based simulation, the simulation using the proposed table lookup optimization method is about 40 times faster and achieves sufficiently accurate results with error less than 1-2%.

Index Terms— Model Compiler, Abstract-Syntax-Tree, Hierarchical Multi-dimensional Table Lookup, Optimization, Circuit Simulation.

I. INTRODUCTION

Manually implementing a compact device model into a circuit simulator is becoming increasingly difficult. It takes on average one to two years for a new device model to become available to circuit designers in a commercial circuit simulator after it is first developed by model developers [1]. This sets a big barrier between model developers and circuit designers; on one hand, a lot of new models are created each year but only a small portion of them are implemented, while on the other hand, the need of using new models is increasing.

In modern deep sub-micron designs, many new effects such as leakage currents need to be considered, which may not be captured in a previous developed device model. Therefore, circuit designers would like to have more freedom to modify device models to meet their specific requirements. Unfortunately, currently there is no convenient way for circuit designers to add the specific effects into their circuit simulator. They have to wait for simulator vendors to take action.

Several compact device model compilers are emerging as a solution for this problem [2][3][4][5][6]. With a model compiler, designers can describe models in high level behavioral languages such as VHDL-AMS or Verilog-A(MS), and then compile automatically to a target simulator. The process for model development and qualification is therefore greatly shortened.

However, a major bottleneck for the mainstream use of model compiler technologies is that the efficiency of automatically generated model is not as good as of manually written device model. It has been shown in [7] that it can be typically 10 to 1000 times slower even for MOS Level 1 model and simple circuits due to the high evaluation cost of automatically generated model. The speed further deteriorates as the complexity of a model and the size of a circuit increase.

To improve the simulation efficiency of automatically generated models, optimization technologies in the process of model compilation become crucial. Some techniques have been reported in [2], which are compiler based and do not trade off between the accuracy and the speed. Results in [2] show that the efficiency can be close to that of manually written codes.

In this paper, we present a systematic method to automatically generate hierarchical multi-dimensional table lookup models for devices with any number of terminals and any set of equations. Table lookup is an attractive way to speed up the simulation by trading off memory and a little bit of accuracy. It has been applied to the simulation of MOSFET transistors [8][9][10][11][12] before. However, all the previous efforts were ad hoc, and designed specifically for a particular device with particular set of equations (MOSFETS in most cases). No works report using table lookup for general device models with any set of equations and any number of terminals (for example, BSIMSOI has six terminals), as required in model-compiler based circuit and behavioral simulation.

This paper details a systematic table lookup method and its implementation in the MCAST model compiler to generate accurate hierarchical multi-dimensional table lookup models for analytical compact devices. In particular, we describe in Sections II and III the use of Abstract Syntax Tree to build table lookup hierarchy and a table lookup algorithm. An error-control based method for table sizing is presented in Section IV. Section V describes test results with our implementation on MOSFE Level 3 model and a set of benchmark circuits.

II. ABSTRACT SYNTAX TREE REPRESENTATION

A compact device model compiler can read compact device models described using high-level behavioral languages such

* This research was supported by DARPA NeoCAD program under Grant No. N66001-01-8920 and NSF CAREER Award under Grant No. 9985507.

as VHDL-AMS or Verilog-AMS, and automatically generate device simulator codes that can be linked with a circuit simulator such as SPICE.

A compact device model is described as a set of time-dependent ordinary differential equations. These equations must be formulated before they can be solved. Using automatic modeling techniques described in [2][14][20][22], these equations can be transformed into a set of nonlinear functions (2.1) to calculate their corresponding entries in the Jacobian matrix and the right hand side (RHS) vectors. These functions will be evaluated during simulation.

$$y_i = f_i(x_1, x_2, \cdots, x_m, c_1, c_2, \cdots, c_n) \qquad (2.1)$$

where x_i are independent variables, such as voltages of device terminals. Since *if-else-endif* block is frequently used when describing complex device models such as BSIM3 and BSIMSOI, c_i are used to formulate condition descriptions. The functions f_i are currently composed of the following operators {+, -, *, /, ^, log, exp}. The operators in c_i include {>, >=, ==, <, <=}. Each function f_i is mapped to an Abstract-Syntax-Tree (AST) that forms the foundation of MCAST and the optimization algorithms.

Figure 1 shows one of the AST of a MOSFET level 1 model. Full description of this model can be found in [2]. The root of the tree is variable Ids, where leaf nodes can be constants or terminal voltages. Different from traditional AST used in computer science, we introduce a new type of *Switch (SW)* node to represent the widely used *if-else-endif* structure in VHDL-AMS. One SW node represents one condition in (2.1).

III. Hierarchical Two Dimensional Table Lookup Algorithm

High computational complexity is a major challenge for device model evaluation. The basic idea of our table lookup method is to replace computation-intensive blocks by two-dimension tables to save the evaluation time. Below, we first describe a table build up algorithm.

A. Building the hierarchy of tables

Our table lookup method starts with the calculation of the evaluation costs of all of the basic operators {+, -, *, /, ^, log, exp, Boolean operators}, etc. The evaluation cost of an operator is an empirical value and is defined as the relative ratio of the running time of the operator to the running time of the "+" operation. This is achieved by taking the average value of 10^6 tests. The evaluation cost of "+" is assigned to 1. Since the evaluation costs may be different on different machine, they are measured in real time when the compiler runs.

The building process of the hierarchical table lookup model is a *reduction* process in which a sub-tree representing a computation-intensive block of the AST is reduced to a two-dimension table. Table 1 shows the reduction algorithm, which is a depth-first, recursive algorithm. It starts from the root of the AST to be optimized, but the real reduction process is bottom-up from the leaf nodes.

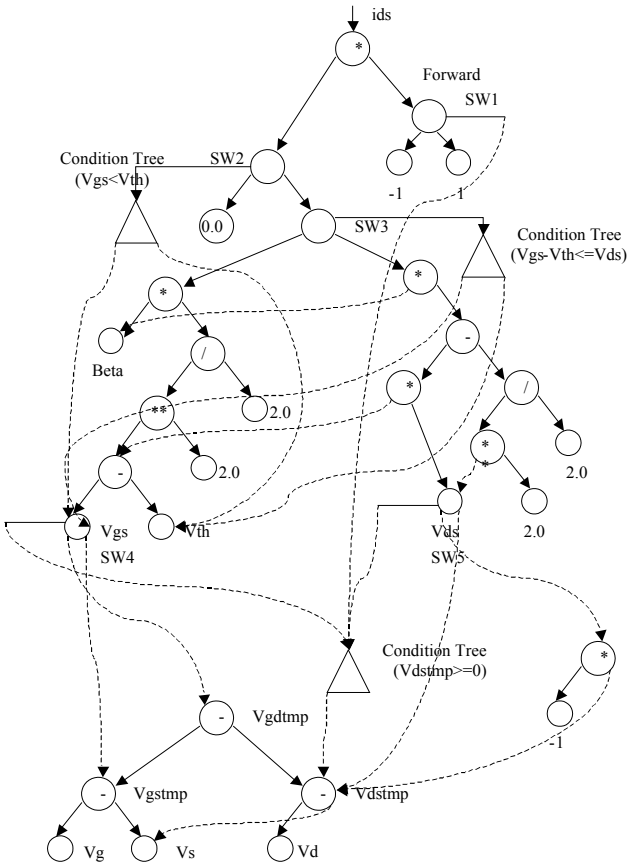


Figure 1. An AST example for MOS Level 1 model.

TABLE 1. REDUCTION ALGORITHM

Algorithm: Reduction
Input: AST Tree Node T This algorithm begins with the root of AST Output: Reduced AST with tables
1. Reduction for T's left child if exists 2. Reduction for T's right child if exists 3. Set related variables for T 4. Combine, if success, return 5. For T's left and right children, if they have been reduced to a table, reset their related variables. 6. Reset T's related variables 7. Set T's calculation cost 8. If T is leaf node, return 9. If T's number of related variables > 2, set T as a bottleneck node, return 10. If T's calculation cost > evaluation cost threshold && T's number of related variables == 2, reduce T to a table.

The details of some steps are explained below:

- A node T's related variables are those node voltages that affect T's evaluation. In step 3, T's related variables are the sum of its children's related variables.
- In order to contain as more operations as possible in the reduced two-dimension table, step 4 has a combination process that helps to build the table upward as high as possible in AST, and thus we can reduce the number of tables. The combination process will try to combine T and its children's tables together if the tables exists and they share the same set of related variables.
- In steps 5 and 6, for T's each child C, if C has been reduced to a table, C's related variables will be reset to contain only one related variable that is C's name. Therefore, we can reduce the number of related variables and can build multi-level tables further based on the new related variable. Accordingly T's related variables are reset based on the children's new related variables.
- In step 7, T's calculation cost is calculated as the sum of T's children's calculation. The calculation cost of leaf nodes, such as the primary device node voltage node, parameter node and constant node, etc., are set to a very small number in practice.
- In step 9, a bottleneck node B is recognized if it has more than two related variables. A bottleneck node can not be reduced to a 2-D table. But B's related variables still have to be reduced to the name of B, and B could become the base related variable of up-level tables.
- Step 10 shows the real condition for T to be reduced to a 2-D table. The evaluation cost threshold is assigned to the evaluation cost from a 2-D table.

Figure 2 illustrates the reduction progress on a MOSFET level 1 AST (simplified for clarity). After the reduction, three tables, A, B and C, are created hierarchically. Table C's relative variables are Vds and B, which itself is also a table.

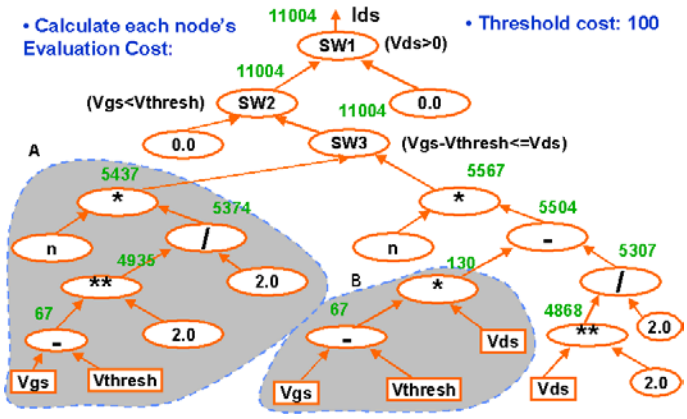


Figure 2(a) AST with evaluation cost.
Assume threshold cost is 100. Sub-tree A and B will be reduced

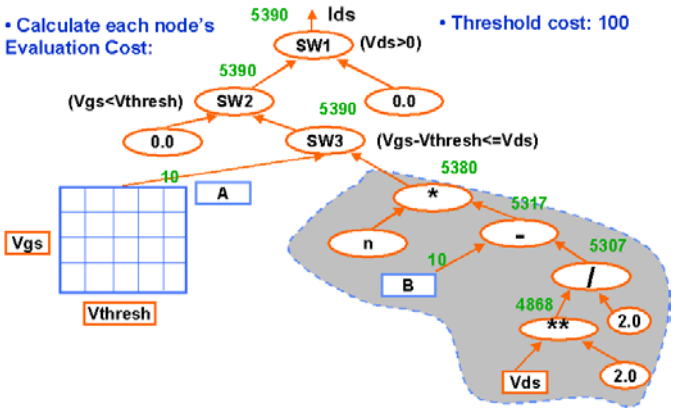


Figure 2(b) Multi-level table reduction.

B. Code generation of the table lookup model

MCAST model compiler generates C/C++ codes for the device model based on the reduced AST. When reaching a table, instead of outputting a block of evaluation codes, a routine of bilinear interpolation [13] for two dimensional table lookup is generated. The computation-intensive block of evaluation codes will also be output but in a separated routine which will be used later on to fill in the table. Bilinear interpolation is adopted since it is computation lightly and it is accurate enough in our process. To locate the four points surrounding the interpolation point, bi-section search is used. One should note that the table spaces are not uniformly separated because dimension variables may change on logarithmic scale and table looked-up variable from the lower level may become clustered or sparse in the dimension for the higher level tables.

C. Evaluation of the table lookup model

The setup routine in a target simulator is modified to fill in the tables for each instance of the device. Compared to the iterative load operation, the running time of the one time setup operation is relatively small [14]. If a circuit to be simulated does not have many new device instances, MCAST has an option to allow the tables to be filled by MCAST and the setup routine in the target simulator only needs to read in the tables, which saves the time for filling the tables.

During the simulation, the computation-intensive blocks are replaced by the computation lightly interpolation processes, therefore, the simulation time is saved.

Huge speedup can be obtained using our proposed hierarchical multi-dimensional table lookup method. But table lookup does introduce errors in the calculation. Simulation result may be wrong if error is not controlled. Beside that, the non-convergence problem may get worse if the circuit is sensitive to the inaccurate calculation of the equivalent conductance (derivative). The additional errors coming from the table lookup may cause the circuit failed to converge. In the following section, we introduce an error-oriented method to control the sizes of the lookup tables.

IV. Error Controlled Table Sizing

As mentioned in the previous section, the table lookup model should have several tables. These tables should be appropriately sized due to the saving requirements of memory capacity and computation time. The aim is to find a set of minimized table sizes such that in the worst case the errors of the interpolated values are less than a given relative error. An error analysis method [15] is used to set the table sizes.

Beginning with a given maximal allowed relative error (Emax), a nonlinear multivariable function is represented by an AST and a given set of intervals for input variables. The AST representing the nonlinear function is decomposed into switch nodes and calculation nodes, each of which is either a double operand operator or a single operand operator, with the restriction for the choice of operators as {+, -, *, /, ^, log, exp}.

For the error analysis, the AST needs to be modified following the rules in Table 2 with an exception that if either A or B is a constant instead of a variable, the modification is unnecessary. The purpose of the modification is making the formal error analysis (will be discussed later) possible.

TABLE 2. AST MODIFICATION RULE FOR ERROR ANALYSIS.

A * B	Exp(logA + logB)
A / B	Exp(logA – logB)
A ^ B (B is a constant)	Exp (B * logA)
A ^ B	Exp(exp(logB + log(logA)))

Since the logarithm function is undefined for arguments that are smaller than or equal to zero. A transformation of a product of two variables is needed for variables that may have negative values (Fig. 3). Similar transformation are required for / as well as ^.

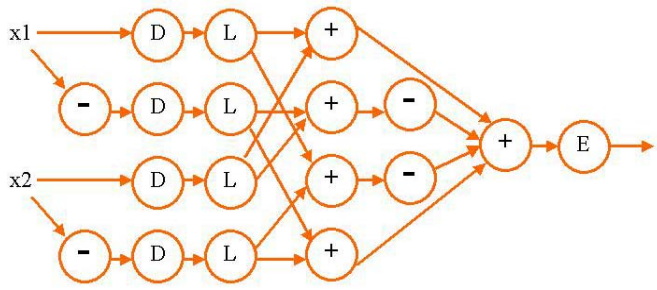
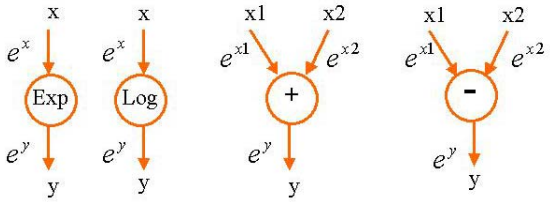


Fig.3. Transformation of variables that may have negative values. Legend: (D) ideal Diode, only positive values can get through.
(L) Log (-) Minus (+) Add (E) Exp

After the modifications and the transformations, the operators like {*, /, ^} will be eliminated from the AST. This modified AST has been isolated as several sub-trees. As mentioned before, each sub-tree is replaced by a two dimensional table. For each of these sub-trees, the error driven sizing algorithm, which consisting of two major steps, is performed to set up an appropriate size of the table. Each of the two steps is a recursive processing along the modified AST.

- First, the intervals of the function and all of the intermediate variables are calculated bottom up rippling from the leaves of the AST. Since the modified AST contains just plus, minus nodes or one incoming node, the intervals are calculated as follows: When a node has one incoming node, its interval is the operation result upon the child's interval. The interval of a plus node is a sum of the intervals of its two children. The interval of a minus node e.g. x1-x2 is (x1min-x2max, x1max-x2min).
- Second, the relative error for each node is calculated top-down staring with the maximal allowed error of the root of the tree and rippling down to the leaf nodes. The error of any node is given by the following equations [15]:



$$e_x = \frac{\ln(1+e_y)}{\max(|x_{\min}|,|x_{\max}|)}; \qquad \text{(Exp)}$$

and

$$e_x = \left|x_m^{\pm e_y} - 1\right|, x_m = \begin{cases} x_{\min}; x_{\min} > 1 \\ x_{\max}; x_{\max} < 1 \end{cases}; \qquad \text{(Log)}$$

For a plus or minus node y to its children x1 and x2:

$$e_{x1} = \left|\frac{e_y}{2}\cdot\left(1+sign(y)\cdot\frac{\min(|x_{2\min}|,|x_{2\max}|)}{\max(|x_{1\min}|,|x_{1\max}|)}\right)\right|$$

$$e_{x2} = \left|\frac{e_y}{2}\cdot\left(1+sign(y)\cdot\frac{\min(|x_{1\min}|,|x_{1\max}|)}{\max(|x_{2\min}|,|x_{2\max}|)}\right)\right|$$

In this way, all of the nodes will get their largest possible relative errors, which will ensure that in the worst case the overall error will be restricted in the given maximal relative error.

After obtaining the interval and relative error of the variable in the table lookup sub-tree, its table size is simply set to be the interval divided by the relative error.

V. Experimental Results

As an example, MOSFET level 3 model [16] has been implemented by MCAST, linked and built in the open source circuit simulator, Berkeley's SPICE3f5, to compare 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 in SPICE3f5, "Non-optimized" model is the one automatically generated by MCAST but without any optimizations, "Table lookup" model is the one automatically generated by MCAST

with optimizations, including table lookup. The accuracy and efficiency of the generated table lookup model are demonstrated by the simulation results.

A. Accuracy

The automatic generated table lookup model of the level 3 model from MCAST is very accurate. Figure 4 shows the comparison of the I-V curves. The automatic generated model without table lookup yields exactly the same results from the manually implemented built-in model of level 3 in SPICE3f5. The simulation results also show that the table lookup model is accurate: the errors are constrained below 2% of the built-in model.

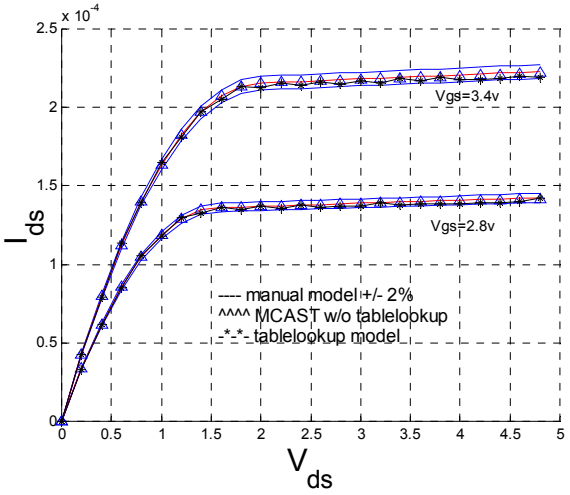


Fig. 4. Accuracy comparison: I-V curves.

Figure 5 shows the transient simulation results of one of our benchmark circuits – power amplifier. The result with table loop up matches well that with analytic evaluation.

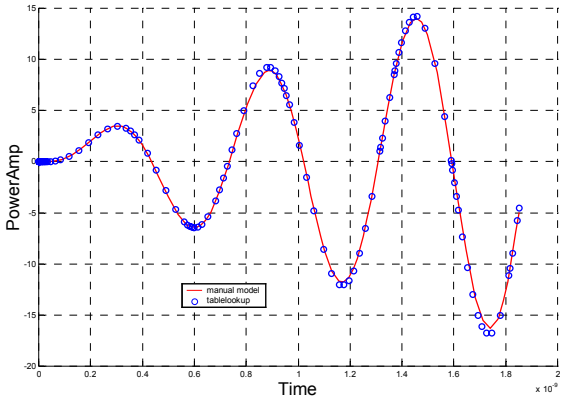


Fig. 5 Accuracy comparison: transient analysis.

B. Performance

Figure 6 shows a comparison among different model implementations, including table lookup model, Built-in model and Non-optimized model, of different devices, such as diode, MOSFET level 1 and level 3. The experiment is circuit-independent and only the model evaluation times are compared and normalized. In pure comparison of the evaluation costs of the different models, the table lookup model is at least three times faster than the built-in model and 20-40 times faster than the non-optimized model.

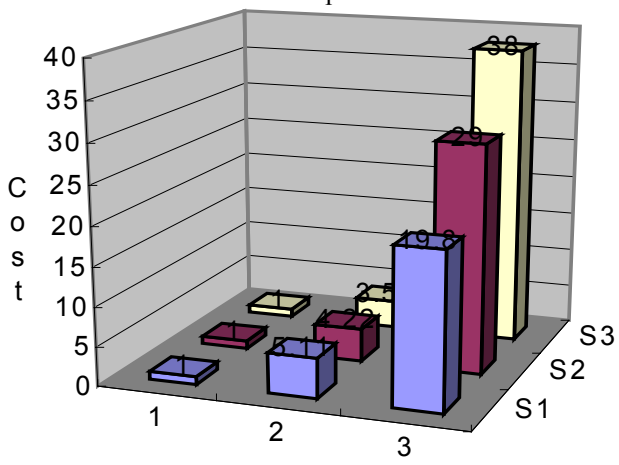


Fig. 6 Normalized model evaluation cost. (1) Table lookup model. (2) Built-in model. (3) Non-optimized model. (S1) Diode model. (S2) MOSFET Level 1 model. (S3) MOSFET Level 3 model.

We also compared the performances in transient analysis. Eight analog and digital benchmark circuits, including Power Amplifier and 8-bit Adder, etc., are used to demonstrate the speed-up results of the table lookup model of the MOSFET model of level 3 versus the built-in model (Fig. 7). We use the device loading time per iteration here for comparison to ignore the convergence effect. The performance of the built-in model is normalized to one. For most of the benchmark circuit, the speed-up is more than two times.

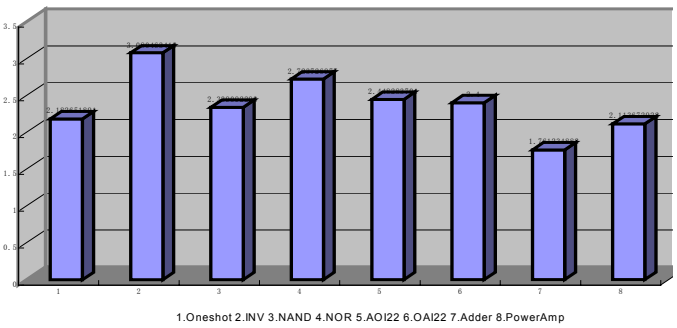


Fig. 7 Speed-up of the table lookup model compared to the built-in model over eight benchmark circuits.

C. Table Sizing

To find out the relationship between the accuracy and the memory requirement, a simple CMOS inverter was tested. We swept the capacity of all tables per instance of the device (MOSFET level 3 NMOS transistor) from 500 points to 20,000 points and collected the overall errors of one of the major variables, e.g., Ids of the pull-down transistor.

Figure 8 indicates that when the table size is small, accuracy is almost proportional to the capacity of the tables (errors are small). Accuracy can be easily improved by extending the table sizes. This corresponds to region 1.

But when the capacity of all tables exceeds a limit point, e.g. 4,000 points in this test case, the gain of accuracy is very limited and accuracy will not be improved by increasing the size of the tables. This corresponds to region 2.

The break point will change depending on the type of function that is being tabled. It is higher for function with complex behavior than for simple function. Fortunately, by setting the overall error allowed to be 2% for the major evaluation variables, the proposed table sizing method usually can find the appropriate sizes for all tables.

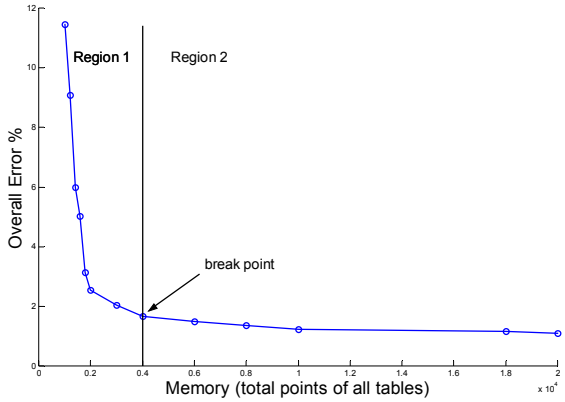


Fig. 8 Error Vs Memory.

VI. Conclusion

We have presented a systematic and automatic method for generating hierarchical multi-dimensional table lookup models for model-compiler-based precise circuit simulation. Any compact device and behavioral model described using high-level languages VHDL-AMS and Verilog-A(MS) can be used. The proposed method is based on an Abstract Syntax Tree representation of behavioral model equations for any devices with arbitrarily number of terminals. A method capable of generating lookup tables subject to a given accuracy requirement but with the minimal amount of memory for storing the data table has been developed.

The proposed method has been implemented in our compact model compiler MCAST and targeted the SPICE3 simulator. Experiment results on a set of standard test circuits have demonstrated that the generated table lookup models are accurate with the error in the range of 1-2%, but at least three times faster than human optimized built-in models, and 30-40 times faster than automatic generated models without optimizations. Furthermore, the proposed error-controlled automatic table sizing method yields nearly minimal table sizes.

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

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

[4] 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.

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

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

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

[8] 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.

[9] 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.

[10] T. Shima, Ts. 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.

[11] 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.

[12] B. R. Chawla, et al. "MOTIS-An MOS Timing Simulator", *IEEE Trans. Circuits and Systems*, Dec. 1975.

[13] H.W. Press, *Numerical Receipt in C*, Cambridge University Press, 1993.

[14] L. W. Nagel, *SPICE2 – A computer program to simulate semiconductor circuits*, Univ. of California, Berkeley, ERL Memo ERL-M520, May 1975.

[15] D.M.W. Leenaerts, W.M.G. van Bokhoven, *Piecewise Linear Modeling and Analysis*, Kluwer Academic Publishers, 1998

[16] HSPICE manual, *Avanti! Corp*. 1999.

[17] Y. Cheng and C. Hu*, MOSFET Modeling & BSIM3 User's Guide,* Kluwer Academic Publisher, 1999.

[18] MAST/Saber User Manual, *Analogy Inc*.

[19] H. A. Mantooth, http://mixedsignal.eleg.uark.edu/paragon.html

[20] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation,* Kluwer Academic Publishers, 1988.

[21] V. Litovsky and M. Zwolinsky, *VLSI Circuit Simulation and Optimization*, Chapman & Hall, 1997.

[22] Vlach, Jiri and Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhodl, 1994.

[23] Coleman, Thomas F. and Varma, *The Efficient Computation of Sparse Jacobian Matricies Using Automatic Differentiation*, Cornel Theory Center Technical Report CTC95TR225, 1996.

[24] A. Griewank, et. Al. "ADOL-C, A Package for Automatic Differentiation of Algorithms Written in C/C++", *ACM Transaction In Mathematics Software*, 1990.

[25] T. L. Quarles, *Analysis of Performance and Convergence Issues for Circuit Simulation*, U. C. Berkeley, Memorandum No. UCB/ERL M89/42, April 1989.