STEAM: Spline-based Tables for Efficient and Accurate Device Modelling

Archit Gupta, Tianshi Wang, Ahmet Mahmutoglu Gokcen, and Jaijeet Roychowdhury
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
Emails: {architgupta,tianshi,amahmutoglu,jr}@berkeley.edu

***Abstract*— A common complaint from users of device models is that the "better" the model, the longer it takes to simulate. Modelling based on interpolation between sampled data points is attractive in this context because it offers low model evaluation times. Although such "table-based" modelling has a long history, important conceptual and implementation issues have been obscure in the literature. These issues include: separating the algebraic ("DC") and dynamic ("charge/flux") components properly; extrapolation outside sampled regions; smoothness; accuracy *vs.* computation *vs.* memory trade-offs; and suitability of the table-based model for various analyses (such as DC, AC, transient, RF, *etc*., analyses).**

In this paper, we clarify precisely what functions should be sampled for a table-based device model to work properly in any analysis. We re-visit interpolation, showing that well-implemented cubic splines provide excellent smoothness and arbitrarily great accuracy at low, almost-constant evaluation cost. However, memory requirements increase with accuracy. We present a novel extrapolation scheme using passivity concepts that aids convergence. Using Berkeley MAPP, we demonstrate speedups of 150$\times$ in core BSIM model evaluations (translating to overall simulation speedups of 6–18$\times$) with relative errors of 0.001%. Our approach can convert any existing device model to a smooth/accurate table-based model with small, fixed evaluation cost. Unlike previous work, our code will be released as open source, serving as a platform for the community to evaluate and experiment with table-based models quickly and conveniently.

I. Introduction

The need for accuracy in compact models of semiconductor devices has been increasing as technologies have scaled and operating speeds/frequencies have increased.[1] As compact models have improved in their ability to accurately model a wide range of technologies and physical effects, their sizes (in terms of equations or lines of code) have also increased, leading to longer device evaluation times and resulting overall slowdowns in circuit simulation. For example, even high-level Verilog-A descriptions of widely-used MOS models such as Berkeley Short-channel IGFET Model (BSIM) and Penn State-Phillips (PSP) model contain thousands of lines of code. It is largely because of device evaluation cost that timing characterization of new cell libraries in industry typically takes months, even using banks of parallel computers. In analog/mixed-signal (AMS) design, faster simulation has consistently been the top item on designers' wish lists.

The established way to develop compact models has been to start with analytical treatments of the device's underlying physics. To faithfully reproduce measurements of real devices, it is usually necessary to augment a purely analytically-derived model with additional phenomenological equations and parameters that tune the model to fit measurements. As new physical phenomena become relevant with device scaling, analytical and phenomenological updates are usually made to the compact model, therefore, the compact model becomes larger and slower.

An alternative and very different philosophy for device modelling, proposed early in the history of simulation [1–4], is to eschew any attempt to understand the device's physics, but instead, to tabulate measured data from the device at a number of bias points and interpolate between these for evaluation. Such "table based" device models have the advantages of fast evaluation (since they involve only table lookups and interpolation), simplicity and generic applicability – in principle, they are easily applied to any kind of device for which tabulated data is available, without the need for equation development or any knowledge of its physics. This is very attractive because developing analytical compact models for modern nano-devices is typically a difficult and time consuming activity, requiring a hard-to-find combination of specialized physics, numerical analysis, and programming expertise.

However, in spite of these attractions and many publications (starting from the 1980s, see Section II for a brief review), table-based models have been far less prevalent in practice than analytically-derived compact models. Indeed, the general impression in the device modelling community is that table-based models are not adequate if accuracy is important, possibly because some of the earliest work [1] focussed on very crude digital simulation applications and did not address the issue of dynamics properly; possibly also because of the high cost of memory in the early years, of which significant amounts are required for high accuracy. Most prior publications [1–7] focus in an ad-hoc manner on one specific type of device (typically MOSFETs), rather than on generally applicable techniques, obscuring important details that can make the difference between whether a table-based model works well or not. These details include how to model dynamics correctly using tables; trade-offs between device evaluation time, accuracy and memory requirements; and extrapolation outside regions where tabular data samples are available.

Our main goal in this work is to evaluate whether analytical compact models (like BSIM, MIT Virtual Source (MVS) *etc.*) can be converted to table-based models for faster simulation while maintaining the high accuracy important for virtually all applications today. Our conclusion is that with proper formulation and implementation, table-based models evaluate much faster than the analytical compact models they are based on, while retaining accuracy that is more than adequate for the vast majority of simulation and design applications. For example, if we sample the BSIM3 model to generate a table-based version with a worst-case error of 0.001%, it evaluates about 150$\times$ faster; using the table-based model for simulation yields speedups of 6-18$\times$ over the original, depending on the analysis (see Section IV).[2] Note that prior work on table-based modelling [7] reports an overall simulation speedup of only 1.6$\times$. Moreover, with Spline-based Tables for Efficient and Accurate device Modelling (STEAM), the speedup is almost independent of the level of accuracy, *i.e.*, one can obtain even greater accuracies, if desired, without slowing the table-based model down. Increasing accuracy does require more memory, however.

We also clarify some important formulation and implementation questions that have been obscure in the literature. One key issue, that has consistently been unclear in the previous attempts at table-based modelling, is how dynamic components (*e.g.*, charges/fluxes) should be modelled as tables. Another issue

[1]The role of crude, low-accuracy simulations has been shrinking- *e.g.*, even digital circuits today require analog-level accuracy during the design process, while analog/RF simulations have always demanded the highest accuracy models available.

[2]A 150$\times$ speedup in device evaluation results in an overall simulation speedup of an order of magnitude less because simulator overheads (such as MNA equation formulation, sparse matrix solution, *etc.*) remain unchanged.

is that the $i - v$ curves of some novel nano-devices (such as memristors and FE-FETs) feature folds or "negative resistance" regions in their characteristics [8], *i.e.*, for some values of bias, multiple values of output current are possible. The question of representing such multi-valued I/O relationships using tables has not even been considered in past work. We show that the correct way to model both DC and dynamic components of a device as tables is simply to sample the f_e, q_e, f_i and q_i functions of its ModSpec description.[3] Moreover, using the ModSpec format to represent table-based models ensures that the model is suitable for any analysis (including DC, AC, transient, harmonic balance, shooting and other future analyses). We use cubic splines for multi-dimensional interpolation in this work as they provide accuracy along with smoothness (for example, continuity of 1^{st} and 2^{nd} derivatives), but our technique can easily be extended to include other interpolation methods like cubic Hermite splines [6] or monotonic piecewise cubic interpolation [3].

As noted in [10], a practical bottleneck in exploring new modelling and simulation ideas has been the lack of a convenient yet sufficiently powerful implementation platform that is openly available. Possibly for this reason, actual implementations of previous table-based models seem unavailable to download and experiment with. The recently-released open source simulator Berkeley MAPP [11], has, however, made quick and convenient prototyping possible; moreover, it includes the ModSpec device modelling system. We have implemented table-based modelling in MAPP and used its simulation capabilities for speedup, memory and accuracy explorations. Since MAPP (and our implementation) are in MATLAB, all *absolute* run times reported here are expected to be large compared to optimized implementations coded in C/C++; however, relative speedups, as well as accuracy/memory trade-offs, in C/C++ implementations are expected to be similar to the results reported here. We plan on releasing our implementation as open source (simultaneously with publication); our hope is that it will help other interested workers verify and build upon this work.

II. Background and Previous Work

A. Previous Work

Work on table-based device modelling dates back to the early 1980s. Splines were proposed in [3, 4] to interpolate branch currents in 4-terminal MOSFETS; in [3] raw device evaluation speedups of 3× were reported. Despite the fact that compact models of the time were far simpler than those today, memory requirements for tables were a significant bottleneck. In an attempt to reduce memory usage, CAzM, a macromodelling simulator, was developed in 1983 [1]. The basic idea of CAzM was to devise table-based macromodels of entire sub-circuits containing many devices but only a few external terminals. The sub-circuits were simulated using DC analysis to build tables of i-v characteristics at the external terminals. Since CAzM catered primarily to the needs of elementary digital simulations, the modelling of charges and dynamics was not emphasised.[4]

In [5], the authors attempted to address the issue of modelling dynamics by separating the dynamic and algebraic components of the terminal currents in a MOSFET as follows:

$$I_{ij}(V_{DS}, V_{GS}, V_{BS}) = I_{DC,ij}(V_{DS}, V_{GS}, V_{BS}) + \frac{d}{dt} Q_{ij}(V_{DS}, V_{GS}, V_{BS}) \,. \tag{1}$$

In (1), I_{ij} is the current between nodes i and j in the model, V_{DS}, V_{GS}, V_{BS} refer to the drain, gate and bulk voltages relative to the source, respectively; $I_{DC,ij}$ refers to the steady-state or "DC" current for the given input voltage values and Q_{ij} refers to the "charge/flux" as a function of the terminal voltages. While this approach is significantly more accurate than the approach taken in CAzM, it still assumes that all the dynamics (*e.g.*, charges) can modelled at the external terminals, *i.e.*, it ignores internal nodes/states, which are critical for accuracy (see Section III-A). More recently, in [6, 7], the authors briefly mention the usage of charges (Q_{DB}, Q_{GB} and Q_{SB}) from the BSIM model, however, no explicit information about the system of equations that represent the table-based model was provided.

[5] reports a speedup of only 1.4× in a SPICE implementation using linear interpolation (which is fast), but their work suffered from modelling errors of up to ∼40%. Using more accurate interpolation methods resulted in their table-based model being actually slower than the analytical compact models they were derived from. [7],[5] on the other hand, reports speedups in the range $1.10 - 1.61\times$ in simulations, with average errors of upto 0.5% in transient simulations. One of the downsides to most of these attempts, besides the fact that none of them are openly available for verification, is that it is very difficult to separate the speedup due to the tabulation-interpolation algorithm from the overall speedup, since this depends on the simulator being used. In this paper, we have provided separate numbers for the two quantities.

B. Relevant Background

1) Spline Interpolation

Splines are piecewise polynomials, used for interpolating a function from sample values provided at a number of sample points or *knots*, *e.g.*, $a = x_0 < x_1 < ... < x_n = b$. A key feature of spline interpolation is that it uses *local polynomials*, *i.e.*, separate polynomials between every pair of adjacent knots. Unlike global polynomial approximations (which use a single high-degree polynomial over the entire range $[a, b]$), separate low-degree polynomials can achieve high accuracy while avoiding artifacts such as the Runge phenomenon [15]. The local nature of splines also leads to fast computation and localized memory access properties during their evaluation. This is discussed in detail in Section IV-D2.

Given a set of knots and sample values at the knots, a cubic spline interpolant is not unique [15], but can be made so by imposing a few additional constraints, typically at or near the first and last knot points. So-called *natural splines* and *not-a-knot splines* are two examples of splines that result from slightly different constraints [15]. In Section III-B, we show that minor changes to natural splines result in an interpolation scheme that provides better performance in simulation.

Univariate splines can easily be extended to higher-dimensional splines using tensor products [16]. However, the efficiency (in terms of runtime) of interpolation is determined by the implementation. Consider a function (say $f(x, y)$) that has been sampled over a rectangular grid of points and that we wish to approximate with a tensor product spline, $\mathcal{H}$. Let the dimensions x and y have n_x and n_y sample points each. One of the ways to build $\mathcal{H}$ is to build splines in x for each of the n_y sample points and store the resulting $n_y \times 4n_x$ coefficients. Interpolating for the function value at an arbitrary point (p, q) requires us to evaluate all the n_y splines at $x = p$, *then* fit a spline in y with the resulting values and evaluate it at $y = q$. While computation of the first n_y splines can be done once and stored for later reuse, the second spline fit has to be done at the time of each device evaluation. While this implementation is common in the literature (*e.g.*, [7]), it is computationally expensive because building splines requires inversion of large matrices.

However, a different implementation with far superior computational properties is available [16].[6] The key realization behind

[3]ModSpec [9–11] is a recently-proposed formulation, with an openly released API, capable of modelling any device from any physical domain.

[4]Modelling internal dynamics for a macromodel is, in essence, the very difficult problem of nonlinear macromodelling [12–14].

[5]The authors seem confused about tensor product splines. In paragraph 4, line 9, they claim that for a tensor product spline, "only the coefficients of the first interpolation dimension can be precomputed and reused", which is not the case.

[6]The spline toolbox in MATLAB uses this superior implementation.

this implementation is that the $4n_x$ spline coefficients for each of the n_y y-samples are themselves functions of y. One can therefore fit a spline for each of these coefficients. This results in a set of $16n_xn_y$ coefficients which completely describe $\mathcal{H}$. In other words, $\mathcal{H}$ is a function of the form:

$$\mathcal{H} = C_3(y)x^3 + C_2(y)x^2 + C_1(y)x + C_0(y), \tag{2}$$

where $C_i(y)$ is a cubic function of y, the coefficients for which can be precomputed as stated and stored. This makes spline evaluation inexpensive and local. However, this is achieved at the cost of more memory since now we need to precompute and store $16n_xn_y$ coefficients instead of the previous $4n_x$ coefficients.

2) MODSPEC

In [9], it is shown that any device model can be completely described using the equations

$$\begin{aligned}\frac{d}{dt}\, q_e((\vec{x}(t), \vec{y}(t)) + f_e(\vec{x}(t), \vec{y}(t)) = \vec{z},\\ \frac{d}{dt}\, q_i((\vec{x}(t), \vec{y}(t)) + f_i(\vec{x}(t), \vec{y}(t)) = \vec{0},\end{aligned} \tag{3}$$

where $\vec{z}$ are so-called *explicit outputs* of the device model, $\vec{y}$ are *internal unknowns* and $\vec{x}$ (*otherIOs*) are all the inputs/outputs (IOs) that are not explicit outputs. These equations are the basis for the MODel SPECification (MODSPEC) device modelling system. MODSPEC constitutes a general and powerful underlying basis for any kind of device model, including table-based ones. (3) is easily explained with a simple example. Figure 1 shows the schematic of a diode. In the schematic, $diode(V_D|I_s, V_{th})$ represents the well-known diode equation relating the current I_D through an ideal diode with the voltage across its terminals (V_D) as

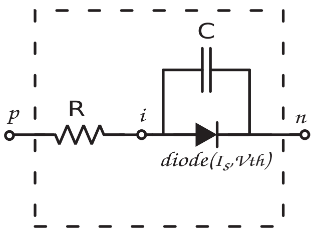


Fig. 1: Schematic of a diode with an internal node *i*.

$$I_D = I_s(e^{\frac{V_D}{V_{TH}}} - 1). \tag{4}$$

(4) does not capture the effects of junction capacitance and series resistances in a real diode, which make the model in Figure 1 more realistic. However, these can be easily incorporated by augmenting (4) to arrive at

$$\begin{aligned}\frac{d}{dt}\underbrace{\vec{0}}_{q_e} + \underbrace{(\frac{v_{pn} - v_{in}}{R})}_{f_e} = i_{pn},\\ \frac{d}{dt}\underbrace{Cv_{in}}_{q_i} + \underbrace{I_s(e^{\frac{V_{in}}{V_{TH}}} - 1) - (\frac{v_{pn} - v_{in}}{R})}_{f_i} = 0.\end{aligned} \tag{5}$$

In (5) v_{in} is an *internal unknown*, corresponding to the branch voltage between the nodes i and n in the schematic. The functions f_e and f_i in (5) represent the algebraic (DC) components of the model, while q_e and q_i represent the dynamic (charge) components. The separation of *explicit* and *implicit* equations is also important as they have different physical meanings. While the explicit equations (formed by q_e and f_e) relate the terminal currents to the *state*[7] of the model, the implicit equations in this case represent the underlying physical laws, *i.e.*, Kirchhoff's current and voltage laws, at the internal nodes. Importantly, f_e, f_i, q_e and q_i are always well-defined functions of their inputs, *i.e.*, each of them has a unique output for a given input. As a result, these functions are well suited for sampling and table-based representation.

III. STEAM

In this section, we describe our contributions and solutions to several implementation and conceptual issues that have kept previous work on table-based models from reaching their full potential.

[7]given by two variables, v_{in} and v_{pn}.

A. Table-Based modelling in MODSPEC

Interpolation schemes for table-based models implicitly assume a functional mapping between the inputs to the model and the interpolated output — *i.e.*, each entry of the table should store a single unique value. Past work has invariably used terminal (external) voltages of a device as the inputs, whereas outputs have been branch currents/conductances/charges, *etc*. In the context of the diode-model discussed in Section II-B2, this translates to a table-based model with v_{pn} as the input and the branch current i_{pn} and capacitor charge Cv_{in} as the outputs. Equation (5) points to a very important fallacy in the assumption that the tabulation of terminal voltages alone, accurately describes a model. The charge on the capacitor C is not a *function* of the terminal voltage v_{pn}. Although it *is* a function of v_{in}, it is not expressible as an algebraic function of v_{pn} directly. Ignorance towards such subtleties has kept past approaches to table-based modelling from being accurate or even correct. Typically, the charge at node *i* is calculated for a DC operating for the terminal voltage v_{pn} and is then placed across the device terminals. Figure 2 shows the effect of such an approximation on a transient simulation of the diode model (from Figure 1) in series with a resistor being fed with a sinusoidal input.

We have also confirmed that this approximation *does not* affect the results of DC analysis. For DC analysis, the contribution of the time-varying components, q_e and q_i, is often irrelevant. In such a situation, the internal unknowns can be solved for as *algebraic functions* of the terminal voltages using DC analysis. If the algebraic solution is unique (*i.e.*, no folds or "negative resistance regions"), the outputs $\vec{z}$ do have a functional relation with the terminal voltage inputs. However, devices with folds, such as FE-FETs and memristors[8], still cannot be modelled properly using only external terminal inputs; internal unknowns are necessary for DC modelling as well.

The MODSPEC formulation includes v_{in} as an internal unknown, making it is easy to express the charge as a function v_{in}. In other words, a table-based representation of a model is not well posed if terminal voltages alone can be used as inputs. Tabulating *all the functions* $(f_e/q_e/f_i/q_i)$ as functions of both the otherIOs ($\vec{x}$, or in this case, v_{pn}) and the internal unknowns ($\vec{y}$, or in this case, v_{in}) is *necessary* for the model to be correctly represented.

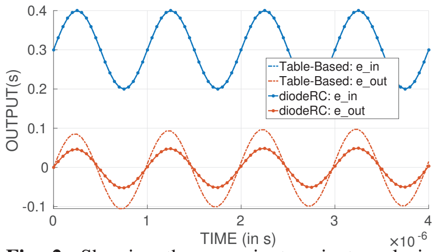


Fig. 2: Showing the error in transient analysis when internal unknowns in the diode-RC model from Section II-B2 are ignored

Although in some cases, it may be possible to eliminate the internal variables analytically, this is typically cumbersome, and not guaranteed to be possible. Most mature MOS models feature internal nodes and unknowns precisely because it is impossible to express the model using terminal quantities alone. With the functions f_e, q_e, f_i, and q_i, MODSPEC provides completely general and flexible facilities for describing models. Our approach towards table-based modelling simply creates accurate table-based representations of these functions.

Building tables of spline coefficients: The most important reason for pushing for table-based models is that the spline interpolant constructed as in Section II-B1 can be used to evaluate all the devices that share one set of process parameters. Moreover, this table needs to be computed exactly once for one *kind* of device and can be stored for later use. In most circuit simulations (esp. transient analysis/dc-sweeps *etc*.), the solution space has a very high degree of spatial locality [6]. This translates to code and data locality for a device model. An efficiently implemented table-based model can thus get a significant boost in performance by using cache locality in modern processors (see Section IV-D2 for details).

B. Passive Extrapolation

It is well known that Newton-Raphson (NR), which forms the core component of virtually any modern analysis, can query a device at highly unphysical input values [17]. However, when NR eventually converges to a solution, the state of the device at the solution should be a physically feasible one, provided the device and circuit have been modelled correctly. In this context, extrapolation plays a very crucial role. It determines how quickly (if at all) NR will return from an unphysical region to a region in which the device is expected to operate and has been sampled for interpolation. Most compact models have custom mechanisms for handling NR convergence, collectively termed limiting methods. The key idea behind limiting is to devise a smooth transition from the functions that describe device behavior to polynomials or other functions known to aid NR convergence.

For example, linear elements, like resistors and capacitors, converge well under NR. One can, therefore, think of an extrapolation scheme that allows a 2-terminal device to undergo a smooth transition from its sampled region to behave asymptotically like a resistor with a given resistance. In Section II-B1, we mentioned that spline interpolation is not unique for a given set of sample points and function values. We have devised a scheme called "passive extrapolation" that supplies additional constraints to spline interpolation to not only make it unique, but to also undergo a smooth transition and behave asymptotically like a linear or quadratic device. As an example, we can define *any* extrapolation slope for the diode model in Section II-B2, while maintaining continuity and differentiability of the device functions f_e, q_e, f_i and q_i over their entire domains.

It is important to note that the our extrapolation scheme can guarantee passivity only for a 2 terminal device. This can be achieved by setting the extrpolation slope to be the value of the device function at the extrapolation boundary divided by the distance of the extrapolation boundary from the origin. In higher dimensions, where extrapolation involves tensor product splines, passivity cannot be guaranteed. However, passive extrapolation serves as a heuristic for Newton convergence by providing flexibility in choosing an arbitrary value of extrapolation slope. For extrapolation in higher dimensional splines (formed by a tensor product of several 1-dimensional splines), each 1D spline is extrapolated using passive extrapolation.

IV. Results

A. Models tabulated with STEAM

Because of its generic applicability, we are able to apply STEAM to several device models, both simple and complex, including BJTs, MOSFETs and custom models like RC segments and the diode model in Section II-B2. The MOS models used in this paper are BSIM3 (v3.2.4) and the MVS Model (MVS v1.0.1)[18]. Both models have 2 internal nodes for representing the series resistances at the drain/source. In our experiments, we connected the source to the bulk node in order to reduce the dimensionality of the tables we needed to make, effectively turning the 4-Terminal models into 3-Terminal ones. The results in the upcoming sections figure MOSFETs since these are the most relevant to industrial practice and are expensive to compute.

B. Evaluating Model Functions

To assess speedup and accuracy, we first evaluated device functions f_e, q_e, f_i and q_i and their derivatives. To build the spline-based tables, we sampled the MOS models at uniformly spaced values of V_{DS} and V_{GS} in the range $[-3, 3]V$ with discretization step sizes varying from 3mV to as coarse as 1V. Samples over these grids were used to construct the spline interpolants. These interpolants were stored and later reused for all the experiments in the paper.

To estimate the error and speedup due to table-based modelling, we evaluated the device functions and their derivatives at points (vds_i, vgs_j), that were *randomly scattered* in the region of interest, *i.e.* $(vds_i, vgs_j) \in [-3, 3]$, for both table-based and original models.

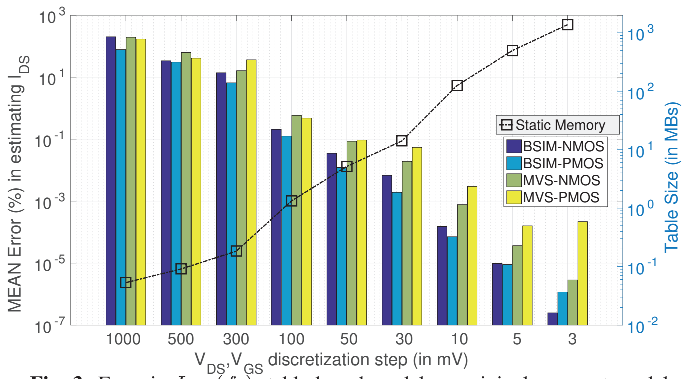


Fig. 3: Error in $I_{DS}(f_e)$: table-based model *vs.* original compact model

Figure 3 reports the percentage error in estimating the drain current I_{DS} and the static memory consumed by the tables generated. We observed that for very coarse grained discretization steps, both the value and the derivative are very inaccurate. However, the accuracy rapidly increases as the discretization is refined. Also, we observed that the error in derivative is more sensitive to the discretization step than the function value itself. This is also reflected in the sensitivity of AC analysis *wrt* the discretization step size (Section IV-C3).

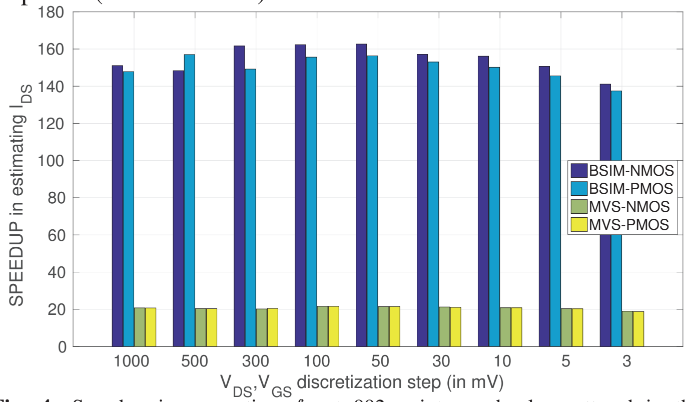


Fig. 4: Speedup in computing f_e at 992 points randomly scattered in the V_{DS}, V_{GS} plane

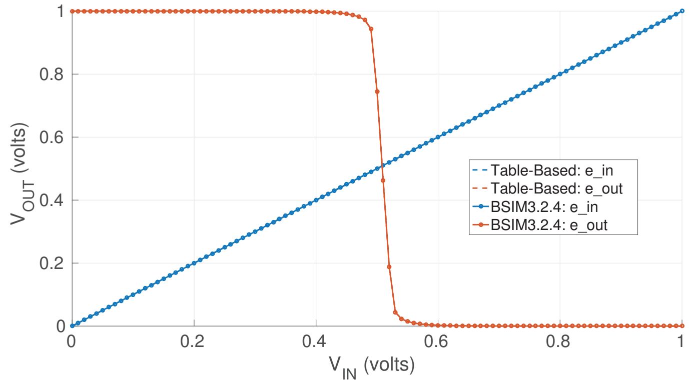
Figure 4 shows the speedup in evaluating the device functions using table-based models with spline interpolation as opposed to evaluating the compact model itself. Significant speedup ($>140\times$) is observed even for the very large tables resulting from a 3mV discretization step. Since the evaluation points are randomly scattered over the entire interpolation space, we see a slight degradation in the speedup as table sizes increase, *i.e.*, the discretization steps become smaller. Later, in Section IV-D2, we show that these effects are compensated by the spatial locality of the data that is accessed in an analysis (like DC/transient *etc.*).

C. Using table-based models inside circuits

We performed a series of experiments to evaluate table-based models for various analyses. We consider the three most commonly used analyses for circuit simulation, *i.e.*, DC, transient and AC. For each analysis, we describe suitable circuits and demonstrate the results (speedup/accuracy) of table-based models and our interpolation/extrapolation approach.

1) DC Analysis

We ran a DC sweep on a CMOS inverter circuit and a differential pair circuit (Figure 6 and Figure 8 respectively). For the inverter, V_{IN} is swept from 0-V_{DD}, whereas for the differential pair, the input voltage V_{IN} is swept in a neighbourhood ($\pm 0.1V$) of the common-mode voltage V_{CM}, which is held constant at $V_{DD}/2$. Figure 5 shows the results for the inverter simulation. It is not possible to distinguish the results from BSIM and our table-based model (discretization step - 5mV) with the naked eye as the error in simulation is limited to 8μV. We have compiled the error and speedup results for different step sizes and models in Section IV-D.



(a) V_{OUT} V/s V_{IN} for BSIM3.2.4 and table based model

Fig. 5: Analyzing the error in DC-Sweep for CMOS-Inverter

2) Transient Analysis

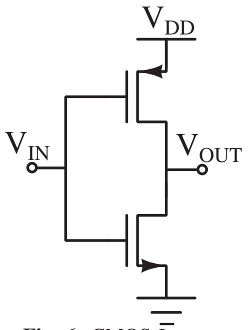


Fig. 6: CMOS Inverter

We tested table-based models in transient analysis for an inverter, a differential pair and 3/7/31/101-stage ring oscillators. While ring oscillator circuits do not require external inputs for the purpose of simulation, the remaining circuits were appropriately biased and fed with a small sinusoidal input (amplitude 0.1V, frequency 1kHz). For all these circuits, we get very similar results from the two models. This points to the fact that for transient analysis, a compact-model and a finely sampled table-based model are identical. Figure 7 shows the error between BSIM3 and its table-based model in the simulation of an inverter.

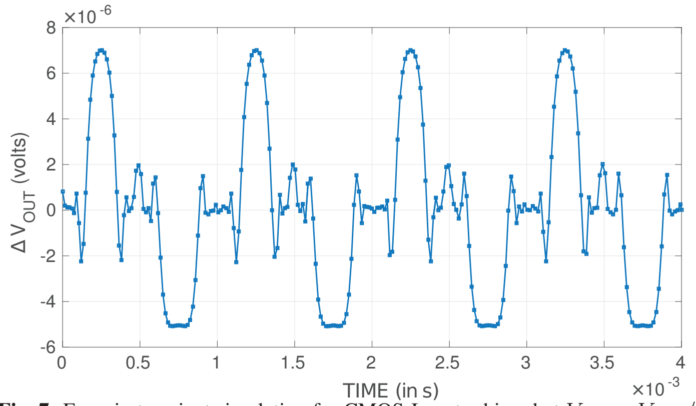


Fig. 7: Error in transient simulation for CMOS-Inverter biased at $V_{IN} = V_{DD}/2$

3) AC, shooting and harmonic balance

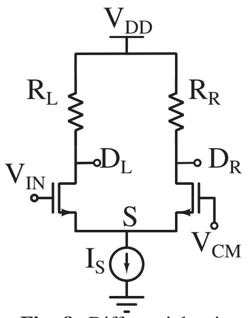


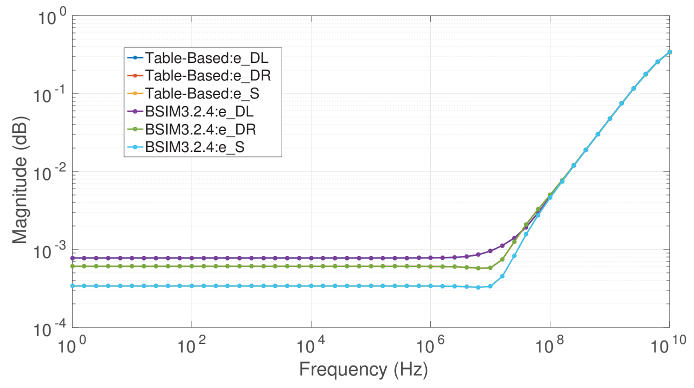
Fig. 8: Differential pair

We ran an AC analysis (*i.e.* to analyze the phase and magnitude responses) on the CMOS inverter and a CMOS differential pair, biased at $V_{IN} = V_{DD}/2$ and $V_{IN} = V_{CM} = V_{DD}/2$, respectively. Figure 9 shows the phase and magnitude plots from the simulation of the table-based model and the BSIM model.

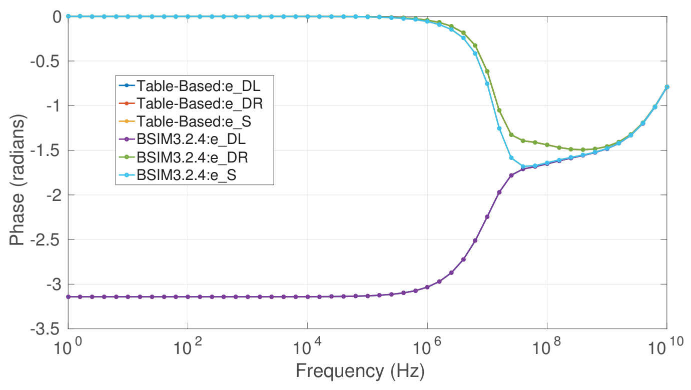
We also tested our table-based models generated with STEAM in some of the more advanced analyses like shooting and Harmonic Balance and saw substantial improvements in both convergence and speed.

D. Summary of Results

Figure 10 summarizes the speedup results for various analyses, discretization steps and circuits. MVS is a far simpler model than BSIM in terms of code complexity, therefore, it does not benefit as much from table-based representation as BSIM. The piecewise polynomial spline interpolation helps speed-up the simulation process in several ways. The most prominent of these is the conversion of a computation of device functions to table-lookup of spline coefficients. The coefficients that are obtained from table lookup are substituted in a cubic expression. This dramatically reduces the computation that is required for calculating the charges/currents in the device. Besides, the smoothness of cubic splines and passive extrapolation cause the table-based model to require fewer NR iterations for convergence.



(a) Magnitude response for MOS differential pair



(b) Phase response for MOS differential pair

Fig. 9: Overlaid plots for AC Analysis on an Inverter and a differential pair for comparing the behaviour of BSIM with a table-based model

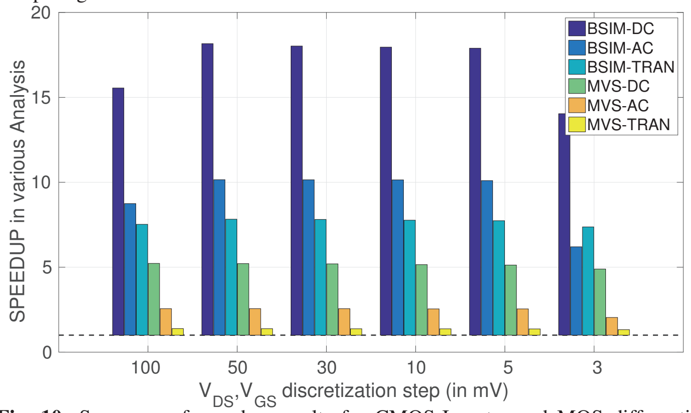


Fig. 10: Summary of speedup results for CMOS-Inverter and MOS differential pair circuits

CIRCUIT	DC	AC	TRANSIENT
Inverter	0.5%	0.9%	0.03%
Differential-Pair	0.004%	0.05%	0.004%

TABLE I: Analysis of worst-case error for table-based BSIM3 in various circuits and analyses

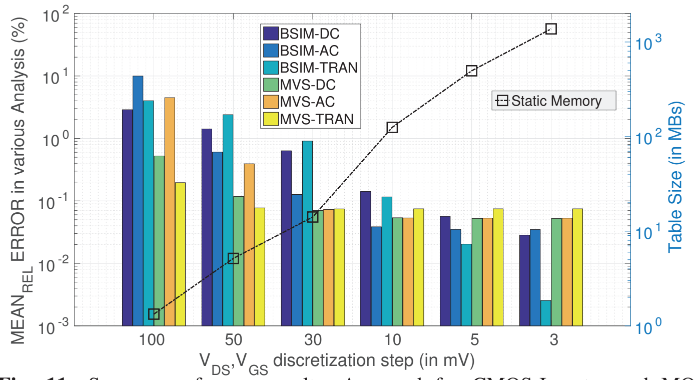


Fig. 11: Summary of error results: Averaged for CMOS-Inverter and MOS differential pair circuits

1) Error V/s Memory

Figure 11 summarizes the error *vs.* speedup trade-offs involved in table-based modelling. The worst-case error follows the same trend as the average relative error. Specifically, for a 3mV discretization step, the worst-case error in simulation is reported in Table I. Overall, the error in simulating an inverter is significantly more than a differential pair and as a result, the inverter error dominates the error plots in Figure 11.

While spline-based models are accurate enough for DC and transient analysis even with coarse grained discretization, AC analysis is very sensitive to the accuracy of the derivatives (*i.e.* the Jacobian matrix). In Section IV-B, we observed that the Jacobian approximation, in turn, is highly sensitive to the discretization step. These experiments provide useful insights into the utility of table-based models for different applications and provide a case for them for accurate and efficient circuit simulation.

2) Cache Behavior

In order to see the impact of processor caches and spatial locality of table accesses, we simulated circuits with increasing complexity and with different sizes of tables. Table II shows the timing profiles for simulations of 3 and 101 stage ring oscillators. Note that many MOSFETs in a circuit typically share a single table and more importantly, access very few portions of that table in a given NR iteration [6]. Therefore, the data being accessed can usually be accommodated in L1/L2 caches.[8] This results in constant and low computational requirements, regardless of table size, and can be seen in Table II: the evaluation time for STEAM remains unchanged (at 20s) for a 3-stage ring oscillator as the discretization step is reduced by an order of magnitude, although table sizes grow from 5.2 MB to 497 MB (See Figure 11). These observations imply that memory accesses, which form the key component of STEAM, are localized and continue to benefit performance even as both the circuit complexity and table sizes grow. As the data in Table II shows, raw device speedups are significantly higher than those for the overall transient analysis.

Simulator overheads depend on a large number of factors, and in this case, overshadow the benefits that can be reaped from table-based models for large circuits. Since Berkeley MAPP is written in MATLAB and designed to be an algorithm prototyping platform, some of the components, like the Modified Nodal Analysis (MNA) equation engine, consume an inordinate amount of time for large circuits. With explicit memory management, pointers and object-oriented programming, a simulator implemented in C/C++ can easily scale the overheads that we see in Table II. In an efficiently implemented simulator, we believe STEAM can speed up simulation by at least an order of magnitude across the board.

V. Conclusions and Future Work

We have shown that using table-based versions of modern compact models can provide significant evaluation and simulation speedups, while retaining accuracy as high as desired. Although memory requirements for table storage grow with increasing accuracy, the memory needed is practical even in low-cost computers today. We believe that this work opens the door to an era of much faster simulation without sacrificing accuracy, and may also ease the flow of compact model development for device physicists.

CIRCUIT	MODEL	Total Time	Model	Simulator overhead
3-Stage	STEAM (5mV)	20.8	4.2	16.6
Ring	STEAM (50mV)	20.7	4.1	16.6
Oscillator	BSIM	82.7	72.3	10.4
101-Stage	STEAM (5mV)	259.6	14.6	245.0
Ring	STEAM (50mV)	256.9	14.4	241.5
Oscillator	BSIM	356.8	205.0	141.8

TABLE II: Runtime analysis for 3-stage and 101-stage Ring Oscillator circuits

References

[1] William M Coughran Jr, Eric Grosse, and Donald J Rose. Cazm: A circuit analyzer with macromodeling. *IEEE Transactions on Electron Devices*, 30(9):1207–1213, 1983.

[2] AR Rofougaran, B Furman, and AA Abidi. Accurate analog modeling of short channel fets based on table lookup. In *Proc. IEEE CICC*, pages 13–1. IEEE, 1988.

[3] Tal Shima and Haruaki Tamada. Table look-up mosfet modeling system using a 2-d device simulator and monotonic piecewise cubic interpolation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(2):121–126, 1983.

[4] James A Barby, Jiri Vlach, and Kishore Singhal. Polynomial splines for mosfet model approximation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(5):557–566, 1988.

[5] Victor Bourenkov, Kevin G McCarthy, and Alan Mathewson. Mos table models for circuit simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):352–362, 2005.

[6] Rouwaida Kanj, Tong Li, Rajiv Joshi, Kanak Agarwal, Ali Sadigh, David Winston, and Sani Nassif. Accelerated statistical simulation via on-demand hermite spline interpolations. In *Proc. ICCAD*, pages 353–360. IEEE, 2011.

[7] Xiao Li, Fan Yang, Dake Wu, Zhenya Zhou, and Xuan Zeng. Mos table models for fast and accurate simulation of analog and mixed-signal circuits using efficient oscillation-diminishing interpolations. *TCAD: IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 34(9):1481–1494, 2015.

[8] Tianshi Wang and Jaijeet S. Roychowdhury. Well-posed models of memristive devices. *CoRR*, abs/1605.04897, 2016.

[9] D. Amsallem and J. Roychowdhury. ModSpec: An open, flexible specification framework for multi-domain device modelling. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 367–374. IEEE, 2011.

[10] T. Wang, K. Aadithya, B. Wu, J. Yao, and J. Roychowdhury. MAPP: The Berkeley Model and Algorithm Prototyping Platform. In *Proc. IEEE CICC*, pages 461–464, September 2015. DOI link.

[11] MAPP: The Berkeley Model and Algorithm Prototyping Platform. Web link.

[12] C. Gu and J. Roychowdhury. Model Reduction via Projection onto Nonlinear Manifolds, with Applications to Analog Circuits and Biochemical Systems. In *Proc. ICCAD*, pages 85–92, November 2008.

[13] M. Rewienski and J. White. A Trajectory Piecewise-Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micromachined Devices. In *Proc. ICCAD*, November 2001.

[14] N. Dong and J. Roychowdhury. General-purpose nonlinear model order reduction based on piecewise polynomial representations. *IEEE Trans. on Computer-Aided Design*, 27(2):249–261, February 2008.

[15] Jens Hugger. *Institute for Mathematical Sciences University of Copenhagen, Denmark (E-mail: hugger@math.ku.dk) May 24, 2007*. Citeseer, 2007.

[16] Carl De Boor. *A practical guide to splines*, volume 27.

[17] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes – The Art of Scientific Computing*. Cambridge University Press, 1989.

[18] S. Rakheja and D. Antoniadis. MVS Nanotransistor Model (Silicon). https://nanohub.org/publications/15, Oct 2014.

[8]If this were not so, dynamic memory requirements would slow the simulation dramatically as the circuit complexity increases.