Extraction of Compact Model Parameters for ULSI MOSFETs Using a Genetic Algorithm

Josef Watts*, Calvin Bittner*, Douglas Heaberlin*, and James Hoffmann**

*IBM Microelectronics, Essex Junction, VT, USA
**University of Vermont, Burlington,VT, USA

ABSTRACT

Extracting an optimal set of parameter values for an FET device model is a complex problem. The final model must not only describe the performance of a set of hardware to an acceptable level of accuracy, but must satisfy criteria outside the device's allowed operating regime to ensure robust convergence properties during simulation. Traditional methods of parameter extraction which rely on gradient techniques can produce far-from-optimal solutions because of the presence of local optima in the solution space. As a result, parameter extraction has traditionally been more art than science, requiring several iterations by an experienced engineer.

Genetic algorithms are well-suited for finding near-optimal solutions in irregular parameter spaces. We have applied a genetic algorithm to the problem of device model parameter extraction and are able to produce models of superior accuracy in much less time and with less reliance on human expertise.

Keywords: Genetic Algorithms, Compact Model Parameter Extraction, BSIM3, PGAPack.

INTRODUCTION

Problem Background

Integrated circuit design relies on device-level simulation (e.g., SPICE) to ensure that circuits will function as intended. Even when static timing analysis or other high-level simulation techniques are used, the accuracy of the analysis relies on timing rules developed from results of device-level simulation [1]. Accurate device-level simulation requires a model that correctly predicts device behavior in all operating regions and for all allowed device geometries. The BSIM3 model from UC Berkeley [2] is widely used because it is physics-based and is supported by an internationally recognized industry standards group (EIA Compact Model Council). BSIM3 represents a set of equations which are implemented on all major commercial circuit simulators and many proprietary simulators. Equal in importance to having a good set of model equations is the selection, or "extraction", of values for the parameters used by the model to describe devices from a particular fabrication process [3].

Traditional model-extraction methods are based on a combination of direct parameter extraction that uses mathematical simplification of the model equations, and optimization that uses the full, highly non-linear model equations [4]. Because of the complexity of the model and data, these methods allow optimization of only a few parameters at a time. Optimization also often leads to local optima which do not result in a model that is accurate enough to be useful [5]. We believe that all commercial extraction tools use a similar approach.

Figure 1 illustrates the complexity of the solution space for an FET device model. This plot was obtained by varying the values of the BSIM3 model parameters dWg and K3 while holding all other parameters constant, and by quantifying the error of the resulting model at each point relative to a set of hardware data. The many "valleys" in the resulting surface represent combinations of dWg and K3 which are optimal locally, but result in a model that fails to describe hardware performance.

Melikian et al. [3] have proposed a global optimization technique based on minimizing the function

$$f = \frac{I_{d,lab} - I_{d,model}}{I_{d,lab}}$$

where the sum is over all lab measurements. However, the technique is susceptible to local minima because it relies on the local gradient of the function. Also, while good results for a single geometry are reported, the problem of fitting multiple geometries with a single global model is not discussed. When additional devices are considered, the function shown does not adequately address the tradeoff between goodness of fit in various regions of operation, and minimizing error over the entire set of device geometries.

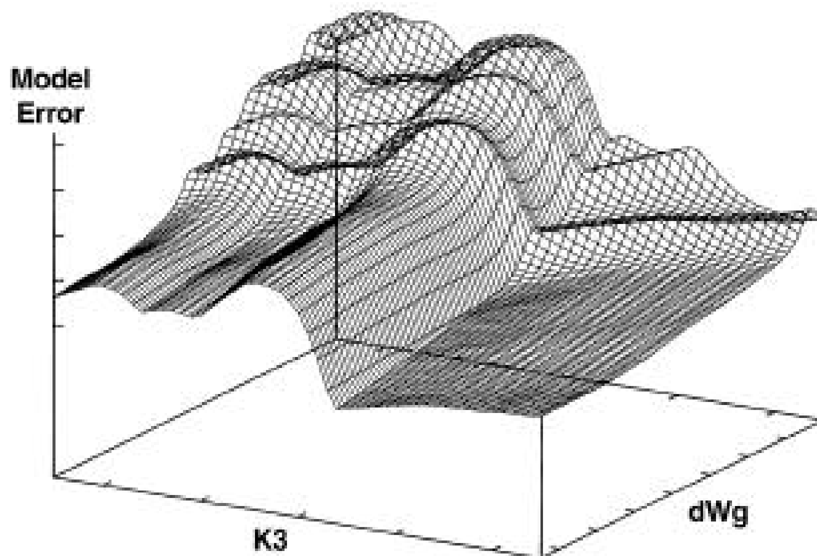

Figure 1: Device model error versus various values of BSIM3 parameters dWg and K3.

Genetic Algorithms

Genetic algorithms (GAs) are a class of computer programs that use principles from biological evolution to model adaptive behavior. Although they have been used in the pure sciences to model and study natural and social systems, they have also been applied with great success to optimization problems [6,7]. GAs tend to search many portions of a solution space simultaneouly, a characteristic which reduces their susceptibility to local optima.

In the parlance of GA optimization, an "individual" represents a potential solution in the problem domain. A population of individuals is created, usually randomly, and each individual is rated on its “fitness” or ability to satisfy a set of criteria specified by the user. The individuals judged to be most fit are allowed to “mate,” producing a new population of individuals, each having some characteristics of both parents. Less-fit individuals do not propagate. This process continues for many generations, with the occasional random mutation thrown in, until a stopping criterion is satisfied. The most fit individual of the final population is then output as the problem solution.

IMPLEMENTATION

We selected PGAPack as the platform for our system; it is a general-purpose, public-domain genetic algorithm software library developed at Argonne National Laboratories [8]. We found PGAPack to be efficient, well documented, easily modifiable and, in general, a flexible and robust platform on which to implement our extraction tool.[1]

PGAPack requires that the user define a representation for individuals in the problem domain of interest and provide a "fitness function" to evaluate each individual.

Representation

Each individual produced by our GA is essentially a string of 51 floating-point numbers, each corresponding to a parameter in the BSIM3 model. In order to ensure that certain key parameters are physically meaningful and to reduce the size of the search space, we excluded some model parameters from our representation. Parameters such as t_{ox}, which can be directly measured in the lab, are fixed at their measured value for the duration of the GA run. Values for other parameters with a strong effect on the fit to hardware data are determined by a physically motivated measurement, but are not held strictly constant. Instead, prior to evaluating an individual, our fitness function calculates appropriate values for these parameters, given the set of parameter values produced by the GA, by forcing an exact fit to pre-selected measured currents.

Fitness Function

Key to the success of any GA-based optimization is a fitness function that can condense all salient aspects of an individual into a single scalar fitness value which accurately reflects the merits of the proposed solution relative to others. Our fitness function is, by necessity,

[1] More information on PGAPack, including downloadable source code and the user’s guide cited in the References, is available on the world wide web at *http://www.mcs.anl.gov/pgapack*.

fairly complex; to assess a model fit, the function compares model performance to data measured at hundreds of bias conditions, and also evaluates model characteristics at bias conditions for which no data exists.

The lab data which fully characterize a given device design over all biases and geometries may include 10,000 to 20,000 individual measurements. Having the GA evaluate thousands of individuals at so many bias conditions is not feasible. However, because the drain current varies in a nearly linear fashion in many regions of operation and the model is constructed to reproduce the observed shape, much of this data is nearly redundant and can be culled to create a manageable set of data which still effectively captures device performance.

To provide a reasonable tradeoff between errors in different operating regions and different geometries, our fitness function translates errors from the domain of the model (drain current) to the input range of the model (applied terminal voltages). The translation has the result of similarly weighting a threshold-voltage error in the subthreshold region of a short wide transistor to a threshold voltage error in the high overdrive region of a long narrow transistor. Specifically, the function used is

$$\frac{I_{d,lab} - I_{d,model}}{V I_{d,model}}, V_g > V_{threshold}$$

$$\frac{\log\left(I_{d,lab} / I_{d,model}\right)}{V \log\left(I_{d,model}\right)}, V_g > V_{threshold} \quad .$$

We chose to use $\log(I_d)$ in the subthreshold region because the drain current is an exponential function of the gate bias in this region. In practice, a smoothing function is applied in the transition region around the threshold voltage where the actual current changes smoothly from exponential to linear. Figure 2 illustrates the translation of errors from domain to range; the gradient shown is with respect to input voltage rather than with respect to the parameters being fit.

To meet the needs of analog circuit designers, the model must accurately predict current derivatives as well as the drain current. Since drain current is a strong function of gate voltage in regions of interest for analog design, $g_m \; \frac{dI_d}{dV_g}$ will be reasonably predicted if the drain current is well predicted. However, in the saturated region of operation (no channel near the drain), the drain voltage has a weak but key effect on the drain current, which is quantified by the current derivative with respect to drain voltage (g_{ds}). After discussions with analog designers, we selected a measurement that correlates well with the way analog designers use the model. Conceptually, we plot the ratio $\frac{g_m}{g_{ds}}$ (self-gain) versus drain current for both model and measurement and then compare the plots.

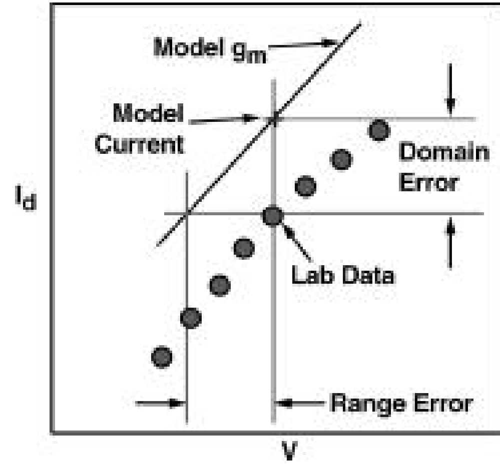

Figure 2: Translation of model errors from domain to range.

For self-gain evaluation, we prefer to compare model and hardware at the same current rather than at the same voltage because analog circuits typically establish an operating point by fixing current. But the model requires voltages as inputs and produces currents and derivatives as *output*; thus it is not feasible to evaluate at the current values measured in the lab. Instead, during initialization of our extraction tool, we fit the measured self-gain data to a power series expansion:

$$\frac{g_m}{g_{ds}} = \alpha_i \left(I_d\right)^i \qquad (i = 1 \text{ to } 5).$$

To evaluate each individual, the fitness function then compares the model self-gain at various current values to the self-gain produced by this power series at the same current rather than comparing the model directly to measured data.

It's possible to build a model that fits the measured data well, but has other undesirable characteristics. For example, a model that produces negative current derivatives at bias conditions beyond the device's physical operating range may prevent a circuit simulator from converging to a solution. Goodness of fit to measured data also does not guarantee that the model will scale linearly with very large width, or scale inversely with very large length. These possible problems are tested as each individual is evaluated; a penalty proportional to the magnitude of any problems found is added. In this way, all modeling criteria that can be quantified are addressed simultaneously rather than sequentially over the course of several iterations with a traditional extraction tool.

RESULTS

As a rough metric of our model accuracy relative to measured data, we routinely use a count of the number of data points for which the model does not predict drain current to within target thresholds set for each region of

operation. Typically, our goal is to predict currents to within 7% of hardware in the strong inversion region, within 50% in the weak inversion region, and within 100% in subthreshold. Our first experiments with the GA were on a 0.25-micron PMOS device for which a model had already been produced with our traditional methods. The model, which had been developed over a period of three weeks using a commercially available extraction tool, failed 9.9% of the data points using these criteria. Our GA-based tool produced a new model in less than a day and it failed only 3.1% of the data points. For this initial test, self-gain was not well predicted because our fitness function did not include a self-gain comparison. However, after we added the self-gain check (described earlier) to our fitness function, the GA produced a model with a similar fit for drain current and substantially improved self-gain accuracy (figure 3).

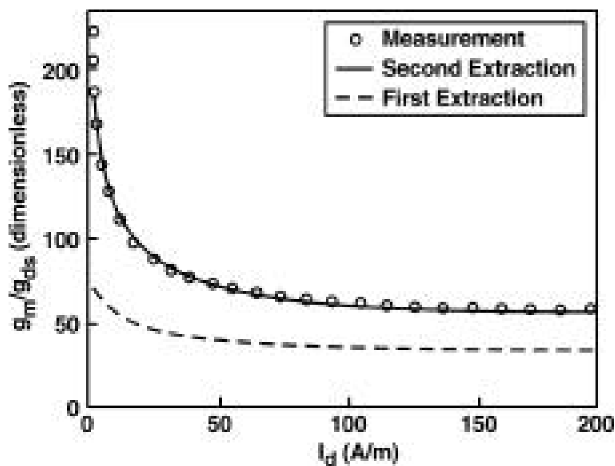

Figure 3: g_m/g_{ds} versus drain current. The first extraction fit the drain current versus voltage well in all regions, but did not predict g_{ds} well. For the second extraction, g_m/g_{ds} was included in the fitness function.

CONCLUSIONS

We have demonstrated the use of a genetic algorithm to extract model parameters for state-of-the-art MOSFET devices. This technique reduces the engineering effort required to produce a model while improving overall model quality. Our previous extraction tool required extensive training and several iterations to achieve a desirable fit, whereas the GA is simple to set up and runs relatively unattended. Due to the complexity of traditional methods, the expertise of the engineer performing parameter extraction has historically been a significant factor in the quality of the model produced. The GA, by contrast, produces excellent fits even in the hands of a novice. And since the GA optimizes all desired model characteristics simultaneously, it tends to produce acceptable models in a single pass compared to gradient-based techniques which can optimize only the model's fit to data.

Based on our experience, the technique described in this paper has become our standard method of parameter extraction and, to date, we have released models produced by GA for both our 0.25- and 0.18-micron device technologies.

REFERENCES

[1] D. Coops, J. Watts, and C. Windisch, "Timing Qualification of a 0.25-µm CMOS ASIC Library Using BSIM3 FET Models," *Proc. IEEE Custom Integrated Circuits Conf.*, 1998, pp. 14.3.1-14.3.4.

[2] Y. Cheng, et.al., "BSIM3v3 Manual," University of California, Berkeley, 1995.

[3] V. Melikian, V. Mnatsakanian, and N. Uzunoglou, "Optimization of SPICE System LEVEL3 MOSFET Transistor Models Based on DC Measurements," *Microelectronics Journal* 29 (1998), pp. 151-156.

[4] Y. Cheng, et.al., "BSIM3v3 Manual," University of California, Berkeley, 1995, pp. 6-1, 6-15.

[5] M. Kondo, H. Onodera, and K. Tamaru, "Model-Adaptive MOSFET Parameter-Extraction Method Using an Intermediate Model," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17 No. 5 May 1998, pp. 400-405.

[6] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1996.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[8] D. Levine, User's Guide to the PGAPack Parallel Genetic Algorithm Library, Argonne National Laboratory Technical Report ANL-95/18.