SIMULATION OF SEMICONDUCTOR DEVICES AND PROCESSES Vol. 4
Edited by W. Fichtner, D. Aemmer - Zurich (Switzerland) September 12-14, 1991 - Hartung-Gorre

Automatic Device Characterization

Péter Verhás and Siegfried Selberherr
Institute for Microelectronics, Technical University of Vienna, Austria

Abstract

An automatic device characterization tool was developed. This tool chooses bias points automatically adapting itself to nonlinearities for simulation of semiconductor devices and builds up a table for interpolation. This table can later be used for fast calculation of device behavior instead of invoking the simulator during device parameter extraction.

1 Introduction

Shrinking the semiconductor devices requires accurate investigations by means of simulation. Numerical modelling was first suggested by Gummel [1] in 1964. Since that time numerical semiconductor device simulators have become well–known and widely used. Today semiconductor device development without simulation is unimaginable. The daily work of an engineer using simulators is to produce input for the simulators, run them and read and interpret the results. This work contains a lot of tedious actions and significant research is done nowadays to automatize it as much as possible.

One approach is to build device characterization tools that calculate the characteristics of a device using simulators over a given region of input parameters by automatically adapting itself to the nonlinearities and building up a table. This table is used later for interpolation. It can be used to derive basic device parameters or applied in circuit simulators [2]. Such a tool was developed using a two–dimensional algorithm. It has been applied for MOSFET simulation using the simulator MINIMOS [3].

In this paper a procedure is detailed which creates the interpolation table automatically. First a one–dimensional algorithm is discussed in section 3. This algorithm is extended for two dimensions in section 4 and experiments were made using this algorithm using the MOS simulator MINIMOS.

2 Requirements for automatic table generation

The requirements can be grouped into two categories: the conditions that automatic device characterization needs, and what it should fulfill.

The automatic table generation needs a simulation environment where it is possible to start a simulation automatically for several bias points. This requirement is fulfilled by the VISTA system [4].

The requirements for the automatic table generator are the following:

- The bias points, for which the simulator is started, should be calculated automatically.
- The points that are simulated should cover the bias domain dense enough to enable accurate interpolation.

- The points should be more dense at nonlinear regions and sparse at linear regions. This way the interpolation error does not depend on the location and is lower everywhere than a given limit.
- The storage of the result of the simulations should be compact.
- The structure of the points should be suited for a fast interpolation tool.
- The different bias points should be simulated in an order that the result of previous calculations can be used for the initial solution estimation in the next simulation step.
- Bias points which are computationally cheap (low voltages) should be calculated first and initial solution extrapolation should be applied for the higher voltages.
- Parameters should easily be estimated

3 One–dimensional mapping

The one–dimensional algorithm is similar to the algorithm published in [5] with some extensions.

A function $f: R \mapsto R$ can be defined with a finite number of points $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ within an interval. The slope of the function in $[x_i, x_{i+1}]$ is

$$s_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}. \tag{1}$$

The nonlinearity of the function at the point x_i is defined as

$$d_i = s_i - s_{i-1}. \tag{2}$$

The definition of the resolution is

$$\alpha = \max_{i=2}^{n} d_i. \tag{3}$$

Using this definition the resolution is strongly linked with the error of the interpolation based on the points $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$.

To characterize a device with one free parameter (like drain voltage) one should specify the range of the characterization (i.e. minimal and maximal value for the parameter), the required resolution of the mapping and an estimated value for the average size of an interval λ. The algorithm:

1. Calculate the function at the points $x_0 = x_{min}$, $x_1 = x_{min} + \lambda$ and $x_2 = x_{min} + 2\lambda$.
2. Set $i = 1$.
3. If $d_i > \alpha$ then the last interval was too long, refine it calculating a new point inside.
4. Estimate a new value for λ according to the nonlinearity of the function at the point x_i using the expression
$$\lambda_{new} = \frac{\alpha}{d_i} \cdot \lambda_{old}. \tag{4}$$
5. Set $i = i + 1$ and $x_i = x_{i-1} + \lambda$.
6. If $x_i < x_{max}$ calculate $f(x_i)$ and go to 3.

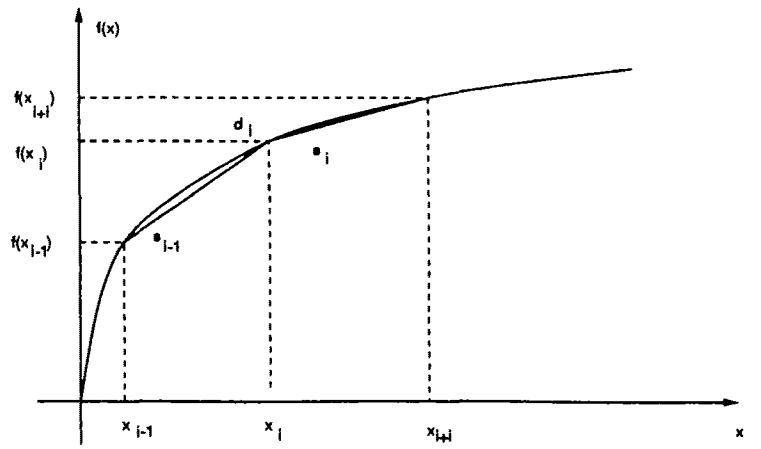

Figure 1: Definition of nonlinearity

During experiments it turned out that some refinement of the algorithm above is required. These refinements do not effect the base of the algorithm and the adaptation of the step size due to nonlinearity. To avoid infinite shortening of λ due to numerical inaccuracies of the simulator we defined a minimal value λ_{min}. A function being almost linear near the point x_{min} causes the problem that the calculated value of λ after the third step is too high. We introduce λ_{max} that stands for the maximal allowed value of λ. As the above algorithm does not calculate the $f(x_{max})$, the interpolation is inaccurate for points being larger than the last calculated. Therefore the value $f(x_{max})$ is always calculated, and if $x_{max} - x_i < \lambda_{min}$ then x_i is set to x_{max} at point 6. The program does not use the value of λ as input but the approximated number of the required points N. The calculation of λ is defined by the expression

$$\lambda = K \cdot \frac{x_{max} - x_{min}}{N}. \tag{5}$$

The value of K is an arbitrary positive value in the order O(1). It is to note that for functions which are almost linear around the point x_{min} a value of $K > 1.0$ fits, and for functions which are strongly nonlinear at the start $K < 1.0$ is better.

The algorithm is very sensitive to the value of α and specifying a too low value causes too many points to be calculated. As an extreme case $[(x_{max} - x_{min})/\lambda_{min}] + 1$ points will be calculated for $\alpha = 0.0$. Fortunately we do not use the one dimensional algorithm alone and the two–dimensional case gives the possibility to eliminate this weakness.

4 Two–dimensional mapping

To map a function $f(x, y)$ on the domain $x_{min} < x < x_{max}$ and $y_{min} < y < y_{max}$ using the one–dimensional algorithm one can define some $g: [z_{min}, z_{max}] \mapsto R^2$ functions and apply the algorithm for the composite function $h(z) = f(g(z))$. The function $g(z) = (\vec{x} \cdot g(z), \vec{y} \cdot g(z))$ defines a curve on the domain. The simplest way is to choose functions of the form

$$g(z) = (A_x z + B_x, A_y z + B_y) \tag{6}$$

where A_x, B_x, A_y, B_y are constant values and

$$(x_{min} - B_x)A_y = (y_{min} - B_y)A_x \tag{7}$$

holds. In this case the curves defined by $g(z)$ are straight lines including the point (x_{min}, y_{min}). Without loss of generality we can assume that for any $z_1, z_2 \in (z_{min}, z_{max})$ the expression $(z_1 - z_2)^2 = \|g(z_1) - g(z_2)\|^2$ is true. This scaling is useful when the two axes have the same dimension (voltage in our experiments). In other cases one can set the values of z_{min} and z_{max} to 0 and 1 respectively.

We say that two such lines g_1 and g_2 are neighbour lines on a subset of the set of this type of lines if there is no line g_3 in the subset for which $A_{x1}A_{x3} < A_{y1}A_{y3}$ and $A_{x3}A_{x2} < A_{y3}A_{y2}$ holds assuming $A_{x1}A_{x2} < A_{y1}A_{y2}$. ($A_{\nu i}$ means the A_ν coefficient for the function g_i and ν stands for x and y.)

The algorithm of the 2D mapping needs N, the number of the required points, the values $x_{min}, x_{max}, y_{min}, y_{max}$ and estimated values of α for g_1 and g_2 as input. The algorithm is:

1. Calculate the value of λ. (Detailed later.)

2. The first two lines of the mapping be

$$g_1(z) = (x_{min}, z), \quad z \in (y_{min}, y_{max}) \tag{8}$$

$$g_2(z) = (z, y_{min}), \quad z \in (x_{min}, x_{max}). \tag{9}$$

   Map the lines using the one-dimensional algorithm, and after mapping calculate values for α that would have been the best for the mapping. Set $i = 3$.

3. Search neighbouring lines g_l and g_k for which there is

$$z \in (\max(z_{min_l}, z_{min_k}), \min(z_{max_l}, z_{max_k})) \tag{10}$$

   that

$$\|g_l(z) - g_k(z)\| \geq \beta\lambda. \tag{11}$$

   Let

$$g_i(z) = (A_{xi}x + B_{xi}, A_{yi}y + B_{yi}) \tag{12}$$

   where

$$\Gamma_{\nu i} = \frac{1}{2}(\Gamma_{\nu l} + \Gamma_{\nu k}) \tag{13}$$

   and Γ and ν stand for all possible combinations of A, B and x, y. If there are no such lines then stop. Set z_{min} so that

$$\|g_l(z_{min} - g_k(z_{min})\| = \beta\lambda \tag{14}$$

   and set z_{max} so that

$$z_{max} = \max\{z: \vec{x} \cdot g_i(z) \leq x_{max} \wedge \vec{y} \cdot g_i(z) \leq y_{max}\}. \tag{15}$$

   Set

$$\alpha_i = \frac{1}{2}(\alpha_l + \alpha_k). \tag{16}$$

   Map the line g_i using the one-dimensional algorithm, and after mapping calculate a value for α_i that would have been the best for the mapping.

4. Set $i = i + 1$ and go to 3.

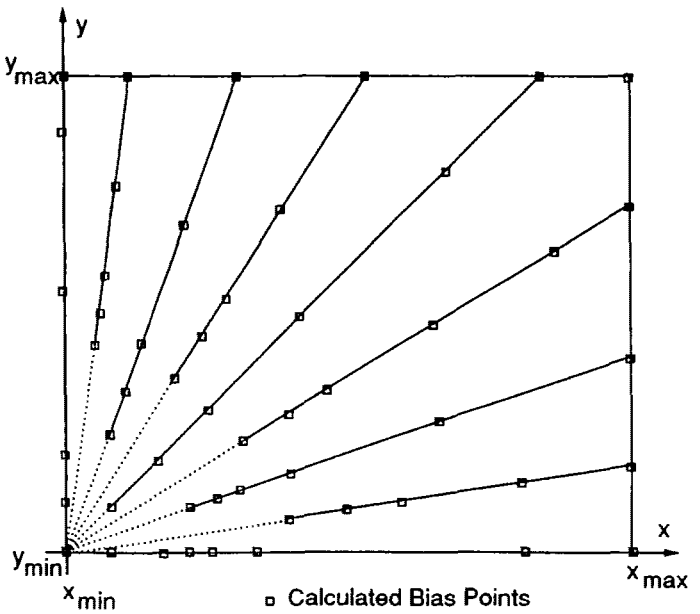

Figure 2: Geometry of calculated points

One can see that following the instructions the geometrical place of the lines is determined by the values of $x_{min}, x_{max}, y_{min}, y_{max}, \lambda$ and β. Therefore the total length L of the lines can be calculated, and a function

$$L = \gamma(\lambda) \tag{17}$$

can describe the relationship. One can also see that $\gamma(\lambda)$ is monotonically decreasing. On the other hand if the total length of the lines L and the number of the expected points n are known, the average step size l_s can be calculated.

$$l_s = L/N = \frac{\gamma(\lambda)}{N} \tag{18}$$

Assuming that the distance between neighbouring points should be approximately the same on one line and between neighbouring lines we can say that the value of λ and l_s should be the same. Putting equations (17) and (18) together we get an equation

$$\lambda = \frac{\gamma(\lambda)}{N} \tag{19}$$

which can be solved iteratively. The value of β in equation (14) is an arbitrary constant value around 2.0.

In steps 2 and 3 we calculate α values for lines that are already mapped. In this action we know the length of the line and the expected average interval length l_s. These two values define the expected number of the points on the given line, n. If we have less than n points, we just take the resolution of the curve. If the number of the calculated points k is larger than n, the used α value was too small. In this case we calculate the resolution of the line according to equation (3) neglecting those $k - n$ points for which the value d_i is the highest. This resolution value gives an approximation for α that would have been good for the mapping. We use these values in equation (16). The only exception is the calculation of α_3 that should not follow this equation because g_1 and g_2 are orthogonal.

5 Interpolation

The interpolation in one–dimension can be done using arbitrary scheme. Using the a-priori knowledge that the function is monotonic one can for instance use monotonic spline interpolation as it is suggested in [2].

The 2D interpolation is based on some one–dimensional interpolation. To interpolate a value for the point (x, y) we should examine the lines

$$g_x(z) = (x, z) \tag{20}$$

and

$$g_y(z) = (z, y)\,. \tag{21}$$

Unless $x = x_{min}$ the set $S_x = g_x \cup g_1 \cup \ldots \cup g_n$ has only finit number of points. Similarly if $y \neq y_{min}$ the set $S_y = g_x \cup g_1 \cup \ldots \cup g_n$ is finite. Choose the line g_ν if $|S_\nu| > |S_\eta|$. Calculate the value of the function f at the points S_ν using one–dimensional interpolations along all lines g_i for which $g_\nu \cup g_i \neq \emptyset$ and using the line g_ν interpolate a value for $f(x, y)$ based on the points S_ν.

This interpolation produces a smooth surface over all the domain except for the region where $|S_x| = |S_y|$. In this region the interpolated function has discontinuities because the direction of the interpolation changes. However it is to note that these discontinuities are within the range of the error of the interpolation. To remove these we build up an equidistant dense mesh over the domain and later we use this mesh to interpolate. For each point of the mesh we use the interpolation described above and when the mesh is ready we apply a 2D low pass filtering to increase smoothness and remove discontinuities [6].

6 Results

As a typical example we have applied the algorithm to simulations of strong inversion characteristics of an N-channel MOSFET using MINIMOS. The boundary values were $U_{Dmin} = 1.0V, U_{Dmax} = 8.0V, U_{Gmin} = 1.0V, U_{Gmax} = 6.0V$. This boundary covers the normal operating region of a typical switching MOSFET for open state, but does not include the subthreshold region. To calculate the points of this region is enough for most of the parameter extraction tools. We have set the value of the expected points $N = 200$. The algorithm produces 187 points. Extending the boundary to $U_{Dmax} = 14.0V$ including the breakdown region the algorithm produces 195 points.

As shown in Figure 3, the points are more dense at low drain voltages where the function $I_D(U_D, U_G)$ is nonlinear, sparse at the saturation region and dense at the breakdown region where the function is strongly nonlinear. One can see very dense points in the breakdown region. This was generated because the value λ_{min} was set too low and numerical inaccuracies forced the algorithm to decrease λ down to λ_{min}.

Our experiments show that the value of K in equation (5) should be around 1.5 for simulation of MOS transistor strong inversion characteristics with U_D or U_G as parameter and $x_{min} \approx U_{th}$. In the example that is shown in Figure 3 the value $K = 1.61$ was used.

To check the error of the interpolation we calculated a 0.2V equidistant mesh over the larger area using the simulator and compared it to the interpolated values. The relative error of the interpolation is below 2% everywhere on both of the domains. This error is small enough for any practical purpose and the simulation can be replaced by table interpolation.

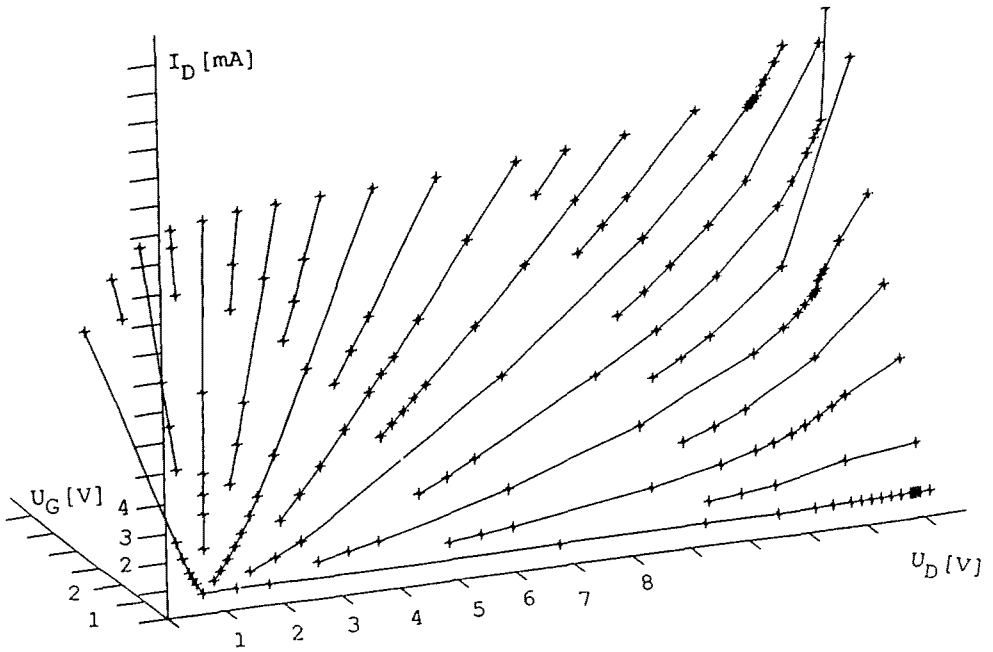

Figure 3: MINIMOS simulation, 195 points

7 Conclusion

In this paper we have described a method to characterize devices automatically. The characterization covers a two–dimensional domain and so the algorithms described in [5] are extended from "Curve–Tracer" into "Plane–Tracer" mode. The algorithms support initial solution estimation for the device simulator and this improves the simulator convergence. The method produces a data structure that can be used for interpolation of scalar or distributed physical quantities with good accuracy. The number of the required points is low taking into account that the simulation time can radically be reduced by the initial solution estimation. The algorithm was constructed taking into account the possibility of further multi–dimensional expansion.

Acknowledgements

This project is supported by the research laboratories of: AUSTRIAN INDUSTRIES - AMS Int. at Unterpremstätten, Austria; DIGITAL EQUIPMENT Corp. at Hudson, USA; SIEMENS Corp. at Munich, FRG; and SONY Corp. at Atsugi, Japan.

References

[1] H.K. Gummel, *A self-consistent iterative scheme for one-dimensional steady state transistor calculations.*, IEEE Trans. Electron Devices, ED–11, pp. 455–465, 1964

[2] Takeshi Shima, Hisashi Yamada, and Ryo Luong Mo Dang, *Table look-up mosfet modeling system using a 2-d device simulator and monotonic piecewise cubic interpolation.*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD–2, pp. 121–126, 1983

[3] S. Selberherr, *Three dimensional device modeling with MINIMOS 5*, In Proceedings VLSI Process/Device Modeling Workshop, pp. 40—41, 1989

[4] F. Fasching *et al.*, *An Integrated Technology CAD Environment*, Proc. Int. Symp. on VLSI Technology, Systems and Applications, Taipei, Taiwan, 1991

[5] Robert W. Dutton and Zhiping Yu, *Algorithms for curve-tracer mode in simulation of devices with highly nonlinear characteristics*, In Proceedings VLSI Process/Device Modeling Workshop, pp. 24–27, 1990

[6] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes*, Cambridge University Press, 1st edition, 1988