Mathematics and Computers in Simulation 28 (1986) 271–284
North-Holland

SIMULATION OF BIPOLAR HIGH-VOLTAGE DEVICES IN THE NEIGHBORHOOD OF BREAKDOWN

Qiming WU and François E. CELLIER

Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, U.S.A.

This paper investigates numerical techniques for the solution of the system of nonlinear elliptic PDE's that simulates bipolar semiconductor devices under high reverse bias voltage. Bipolar devices are usually modeled as a set of three elliptic PDE's representing the field equation and the electron and hole current continuity equations. However, these equations can be solved effectively in the case of low-voltage devices only, e.g. by use of the the BAMBI program. As shown in this paper, it is possible to simplify the model in case of high-voltage devices. A powerful software toolkit, ELLPACK, has been used to solve the resulting two-dimensional Poisson equation. ELLPACK enabled us to easily compare several numerical solution techniques for their efficiency to solve this problem. To analyze the effectiveness of the toolkit as a whole, a comparison was made between ELLPACK and a special-purpose program written by us.

1. Introduction

In order to develop new semiconductor devices and study different kinds of semiconductor devices, two-dimensional or three-dimensional device simulation is needed. Device simulation has become an important tool for Computer-Aided Design of LSI circuits, specially new MOS devices, and in the investigation of electrical characteristics of devices, e.g. the study of bipolar transistors with complex structure.

The task of device simulation is to solve basic semiconductor equations, the Poisson equation and electron and hole current continuity equations, directly for a specified domain with certain boundary conditions and doping concentration. Both Poisson equation and current continuity equations are nonlinear elliptic partial differential equations which have to be solved simultaneously.

Our specific purpose of doing static two-dimensional device simulation is to study electrical characteristics and breakdown behavior of bipolar transistors under high reverse bias voltage. In this special situation, the physical model can be simplified. For a high reverse biased junction, the currents are negligible, and the Poisson equation is the only equation to be solved. The resulting electric potential and carrier density distributions are sufficiently accurate [8]. Once the electric potential has been determined by the simulation program, the breakdown voltage can be evaluated from there.

Even with this simplification, the computation of the numerical solution for the resulting two-dimensional Poisson equation under high bias voltage is very time consuming. This is due to the fact that the resulting field exhibits very large and unequally distributed gradients which calls for a large number of grid points (at least several hundreds). To keep the computational cost in

acceptable bounds, it is important that a good initial guess for the solution considering the electrical neutrality condition is found.

ELLPACK [9] is a powerful software system for solving elliptic partial differential equations. ELLPACK is a *toolkit* in that a fair number of different discretization and solution techniques have been integrated into the program. These techniques are parameterized. Thus, to switch from one technique to another, the user simply replaces one phrase in his program by another. In this paper, the different solution techniques available in ELLPACK are discussed with respect to their efficiency to solving the posed problem.

In Section 2 of this paper, the physical problem is described. The basic semiconductor equations with corresponding boundary conditions, and the simplification of this physical problem are introduced.

In Section 3 of the paper, the ELLPACK software is briefly described. Equations and boundary conditions that are acceptable to ELLPACK, discretization methods, numerical solution techniques of the underlying linear algebraic equations, and some basic ELLPACK statements are presented.

In Section 4, it is shown how ELLPACK can be used to solve the two-dimensional nonlinear Poisson equation that models our class of devices. Newton iteration was used to linearize the nonlinear equation. Both finite differences and finite elements were used to discretize the equation. Both direct and iterative techniques were applied to the solution of the underlying linear equations.

Section 5 lists a comparison of results obtained from these different numerical methods. The execution time, storage requirements, and differences in the results are shown.

Finally, Section 6 presents further investigations and conclusions.

2. Physical model and basic equations

On a domain with known doping concentration and boundary conditions, the basic semiconductor equations: the Poisson equation and electron and hole current continuity equations are solved. For static two-dimensional device simulation, the basic semiconductor equations can be modeled as follows (normalized form) [10]:

Poisson equation:

$$\partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 = -(p - n + N_d); \tag{1}$$

Current continuity equations:

$$\operatorname{div}(\boldsymbol{J}_n) = R, \tag{2}$$

$$\operatorname{div}(\boldsymbol{J}_p) = -R; \tag{3}$$

Current density equations:

$$\boldsymbol{J}_n = -\mu_n n\ \mathbf{grad}\,(\psi) + D_n \mathbf{grad}(n), \tag{4}$$

$$\boldsymbol{J}_p = -\mu_p p\ \mathbf{grad}(\psi) - D_p \mathbf{grad}(p) \tag{5}$$

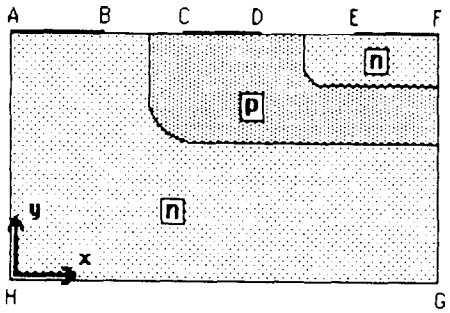

Fig. 1. Typical domain geometry of a bipolar transistor.

where ψ, n, and p are unknown variables.

$$n = \phi_n \exp(\psi), \tag{6}$$

$$p = \phi_p \exp(-\psi), \tag{7}$$

$\psi(x, y)$ = static electric potential, unknown variable
$n(x, y)$ = electron concentration, unknown variable
$p(x, y)$ = hole concentration, unknown variable
$N_d(x, y)$ = doping concentration, known physical function
$\boldsymbol{J}_n(x, y)$ = electron current density
$\boldsymbol{J}_p(x, y)$ = hole current density
μ_n = electron mobility, experimentally determined function, related to the electric field ($\boldsymbol{E} = -\mathbf{grad}(\psi)$)
μ_p = hole mobility, experimentally determined function
D_n = electron diffusion constant ($\mu_n/D_n = q/kT$)
D_p = hole diffusion constant ($\mu_p/d_p = q/kT$)
ϕ_n = electron quasi-Fermi potential
ϕ_p = hole quasi-Fermi potential
R = recombination rate, related to n, p, $\boldsymbol{J}_n$, and $\boldsymbol{J}_p$.

A typical domain of a bipolar transistor is shown in Fig. 1. Corresponding boundary conditions are listed as follows:

(1) *AB, CD, EF*:

$$\psi = \psi_0, \quad n = n_0, \quad p = p_0$$

(2) *BC, DE*:

$$\partial\psi/\partial y = Q_s, \quad \boldsymbol{J}_n \mid_n = -R_{sn}, \quad \boldsymbol{J}_p \mid_n = R_{sp}$$

(3) *FG, GH, HA*:

$$\partial\psi/\partial y = 0, \quad \boldsymbol{J}_n \mid_n = 0, \quad \boldsymbol{J}_p \mid_n = 0$$

where:

Q_s = surface charge density, experimentally determined constant
R_{sn} = surface recombination rate of electrons, experimentally determined constant
R_{sp} = surface recombination rate of holes, experimentally determined constant
$J_n|_n$ = orthogonal component of the vector J_n
$J_p|_n$ = orthogonal component of the vector J_p

In order to compute the unknown variables ψ, n, and p, (1), (2), and (3) must be solved. This set of equations is a set of nonlinear coupled elliptic partial differential equations. The numerical solution of this kind of equations includes the following steps:

1. Transform the continuous problem to a discrete one. Finite difference method and finite element method are two methods usually employed.
2. Linearize the nonlinear PDE. Newton iteration is commonly used for that purpose.
3. Determine the scheme of solving the coupled PDE's. Three techniques have been described in literature: In one, the three discrete equations are decoupled and solved iteratively [6]. The second technique solves the equations simultaneously [2]. Recently, a Block-Newton-SOR method has been employed [5].
4. Choose the numerical solution technique for the underlying linear algebraic equations. Both direct and iterative (SOR) techniques are frequently used. Some of these techniques (e.g. band structure techniques) can be further improved by numbering the equations in an intelligent way (e.g. to minimize the width of the band). This problem is referred to in the literature as the *indexing* problem.

For this set of two-dimensional PDE equations, we require hundreds if not thousands of discrete nodes as the resulting gradients are very large and unequally distributed. The numerical solution of this problem is therefore time consuming. Furthermore does the convergence of the Newton iteration depend heavily on the working condition of the device, e.g. the bias voltage on the transistor. When the reverse bias voltage of the junction is increased to hundreds of volts, eqs. (1), (2), and (3) become almost impossible to solve numerically. The computing effort grows from a couple of minutes to many hours of CPU time on a VAX 11/750.

Our purpose of device simulation is the investigation of breakdown behavior when the transistor collector is on high reverse bias voltage. Under those conditions, the model can fortunately be simplified. The currents can be assumed to be negligible.

Under reverse bias condition, when the current is not significant, ϕ_n can be approximated by its n-contact bias value and ϕ_p can be approximated by its p-contact bias value, i.e. $\phi_n = V_r$ and $\phi_p = 0$, where V_r is the value of reverse bias voltage on the p–n junction.

After making the above simplification, ϕ_n and ϕ_p have become constants. n and p can now be solved from (6) and (7), and can be plugged into (1). The Poisson equation is thus the only remaining PDE, and can be written as follows:

$$\partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 = -\left(\phi_p \exp(-\psi) - \phi_n \exp(\psi) + N_d\right). \tag{8}$$

From this PDE, we compute the electric potential ψ, and then indirectly by use of (6) and (7) the electron and hole concentrations. Finally, the focus of the problem is on how to solve the Poisson equation efficiently.

3. ELLPACK

ELLPACK is a software system for solving elliptic boundary value problems. The software includes both a very high-level problem statement language and an extensive and extensible library of problem-solving modules. Even the problem statement language itself can be extended by use of a *compiler-compiler* and a *data template processor*.

The ELLPACK system solves single linear elliptic equations of the form (9) and (10) on general domains in two dimensions and rectangular domains in three dimensions with boundary conditions of form (11):

$$(au_x)_x + (cu_y)_y + fu = g \tag{9}$$

where a, c, f, and g are functions of x and y. The ellipticity condition $ac > 0$ must be satisfied in domain $\boldsymbol{R}$.

$$au_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + fu = g \tag{10}$$

where a, b, c, d, f, and g are functions of x and y. The ellipticity condition $b^2 - ac < 0$ must hold in domain $\boldsymbol{R}$.

The corresponding boundary condition along boundary $\Delta\boldsymbol{R}$ specified for elliptic problems takes the following form:

$$p(x, y)u_n + q(x, y)u = r(x, y). \tag{11}$$

For three-dimensional problems, a term in u_{zz} must be added to (9), terms in u_{zz}, u_{xz}, u_{yz}, and u_z have to be added to (10), and the z-coordinate is to be added to (11).

There are two stages to the numerical solution of elliptic boundary value problems. The first stage, called *discretization*, replaces the continuous problem by a discrete problem with approximately the same solution. In the case of (9), (10), and (11), one obtains a system of linear algebraic equations. The second stage is the *solution* of this algebraic system.

Two discretization methods are included in ELLPACK, the finite difference method and the finite element method. For finite differences, there are 5-point star, 7-point star, and higher order HODIE methods available. For finite elements, Galerkin Splines and collocation techniques are provided.

Methods for solving the discretized linear algebraic equations can be classified as either direct or iterative. In ELLPACK, there are a number of direct solving methods provided, such as BAND GE (Gaussian elimination with scaled partial pivoting for a general band matrix), LINPACK SPD BAND (Cholesky decomposition for a symmetric positive definite band matrix), SPARSE LU UNCOMPRESSED (general sparse Gaussian elimination), and several others. There are also several iterative solving techniques available, including SOR (successive overrelaxation), JACOBI SI (Jacobi iteration with Chebyshev semi-iterative acceleration), JACOBI CG (Jacobi iteration with conjugate gradient).

Users can choose independently between the different discretization methods and solving techniques in accordance with the numerical properties of their problem.

ELLPACK users specify their problem in a simple user-oriented *ELLPACK language*. The ELLPACK preprocessor translates this program into a FORTRAN source program, called the

ELLPACK control program. This program is then compiled and linked with a precompiled *ELLPACK module library*. Finally, the program is executed, producing a solution to the problem. Tabular listings and contour plots are provided in ELLPACK.

The ELLPACK language is actually an extension to FORTRAN, that is: ordinary FORTRAN statements can be intermixed with ELLPACK statements. Elliptic PDE's, domains, and boundary conditions can be stated directly in this language by using two segments: the *EQUATION* and the *BOUNDARY* segment.

Other basic segments are the following:

DISCRETIZATION: The partial differential equation and boundary conditions are approximated by discrete linear algebraic equations. Users can choose between several discretization methods provided by ELLPACK.

INDEXING: The equations and unknowns of the discretized system are reordered to facilitate the solution of the system. Most of these techniques are meaningful in connection with particular solution techniques only. The set of indexing algorithms available in ELLPACK includes:

(1) NESTED DISSECTION: Computes the 'nested dissection' ordering of the equations. This technique is used together with sparse Gaussian elimination.

(2) RED-BLACK: The variables and unknowns are numbered as on a checker board, all 'red' points before the 'black' point. This method is primarily used in connection with REDUCED SYSTEM and SOR iteration.

(3) MINIMUM DEGREE: Computes a minimal degree ordering of the equations. Also this algorithm is used with sparse Gaussian elimination.

SOLUTION: This segment describes the modules used to solve the linear system of equations. The user can select among many direct or iterative algorithms.

OUTPUT: The solution is tabulated or plotted.

FORTRAN: The FORTRAN segments can be placed freely anywhere in an ELLPACK program. They indicate that the included statements are to be inserted into the ELLPACK control program without any preprocessing. Users can use FORTRAN segments to define various functions and/or to perform special calculations, such as computations that interact with ELLPACK modules for the solution of nonlinear or other special problems.

SUBPROGRAM: This segment defines complete FORTRAN functions or subroutines.

DECLARATION and *GLOBAL* segments can be used to enable interactions between Fortran segments and ELLPACK modules.

4. Description of solution technique

The numerical solution of the Poisson equation includes linearization of the nonlinear equation, discretization of the PDE, and solution of the set of resulting linear algebraic equations.

In order to linearize (8), Newton iteration was used. This method converges very rapidly once one gets reasonably close to the solution.

Let

$$F(\psi) = \partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 + \left(\phi_p \exp(-\psi) - \phi_n \exp(\psi) + N_d\right) = 0.$$

The formula for the Newton iteration is as follows:

$$F(\psi) = F(\psi_0) + F'(\psi_0)\cdot(\psi - \psi_0) = 0$$

and therefore,

$$F'(\psi_0)\cdot\psi = -F(\psi_0) + F'(\psi_0)\cdot\psi_0.$$

Let $\Delta\psi = \psi - \psi_0$:

$$\begin{aligned} F(\psi_0 + \Delta\psi) &= \partial^2(\psi_0 + \Delta\psi)/\partial x^2 + \partial^2(\psi_0 + \Delta\psi)/\partial y^2 + \phi_p \exp(-\psi_0)\cdot(1 - \Delta\psi) \\ &\quad - \phi_n \exp(\psi_0)\cdot(1 + \Delta\psi) + N_d \\ &= \partial^2\psi_0/\partial x^2 + \partial^2\psi_0/\partial y^2 + \phi_p \exp(-\psi_0) - \phi_n \exp(\psi_0) + N_d \\ &\quad + \left[\partial^2\Delta\psi/\partial x^2 + \partial^2\Delta\psi/\partial y^2 - \left(\phi_p \exp(-\psi_0) + \phi_n \exp(\psi_0)\right)\cdot\Delta\psi\right] \\ &= F(\psi_0) + F'(\psi_0)\cdot\Delta\psi, \end{aligned}$$

that is

$$\begin{aligned} &\partial^2\psi/\partial x^2 + \partial^2\psi/\partial y^2 - \psi\cdot\left(\phi_p \exp(-\psi_0) + \phi_n \exp(\psi_0)\right) \\ &\quad = -(\psi_0 + 1)\phi_p \exp(-\psi_0) - (\psi_0 - 1)\phi_n \exp(\psi_0) - N_d, \end{aligned} \tag{12}$$

or

$$\partial^2\Delta\psi/\partial x^2 + \partial^2\Delta\psi/\partial y^2 - \Delta\psi\cdot\left(\phi_p \exp(-\psi_0) + \phi_n \exp(\psi_0)\right) = -F(\psi_0). \tag{13}$$

Both (12) and (13) are Newton iteration formulae for the Poisson equation. The iteration algorithm works as follows:

choose initial guess ψ_0
REPEAT
  solve eq. (12) or eq. (13)
  set $\psi \to \psi_0$ or $\psi_0 + \Delta\psi \to \psi_0$
UNTIL converged.

Equation (12) lends itself more easily to implementation in ELLPACK. It was therefore used when working with ELLPACK. In our own program, about which we shall report in due course, the somewhat simpler form of (13) was used.

We tried several of the finite difference methods and of the finite element methods offered in ELLPACK. Several of the solution techniques, both direct and iterative, were investigated.

In every step of the Newton iteration, the coefficient matrix of the discretized equations is reevaluated, and the linearized equations are solved. The execution time consists of the number of Newton iteration steps multiplied by the time needed for one iteration which in turn is composed of the time needed for discretization and the time needed for the solution of the linear equations.

The ELLPACK language was used to organize the overall FORTRAN program. ELLPACK statements are interspersed with extensive FORTRAN segments. The total program consists of roughly 1500 lines of code.

5. Comparison of results

A p–n junction on a rectangular domain is simulated. Its geometry is shown in Fig. 2. In this set of simulation runs, we chose the reverse bias to be 5 volts. This represented a sufficiently simple problem to allow us to compare many different numerical algorithms without need for extensive number crunching. All computations were performed on a VAX 11/750 running VMS.

5.1. Comparison of different numerical methods

In ELLPACK, two different equation representations can be utilized. One is the *self-adjoint form*, i.e. the form of (9). The other is the *nonself-adjoint form*, i.e. the form of (10). The results from our computations show that the discretization time needed in the two cases is substantially different. When the number of grid points is chosen to be 24×14, and a non-uniform grid is specified, the discretization time for the nonself-adjoint form is 24.0 seconds whereas it is 57.0 seconds in case of the self-adjoint form. For this reason, we have used only the nonself-adjoint form in the following computations.

According to the properties of our problem:

(1) rectangular domain,
(2) both Dirichlet and von Neumann conditions exist on the boundaries,
(3) non-uniform grid,
(4) non-symmetric linear equations,

only a subset of the discretization methods and solving methods of ELLPACK can be used. (Some of the methods provided in ELLPACK work e.g. only on an equidistantly spaced grid.)

We compared the following discretization methods and solving algorithms:

(i) *discretization methods*:
- HERMITE COLLOCATION
- 5-POINT STAR,

(ii) *solving methods*:
- BAND GE
- SPARSE LU UNCOMPRESSED + MINIMUM DEGREE
- SPARSE GE NO PIVOTING + MINIMUM DEGREE
- SOR
- JACOBI CG + RED-BLACK.

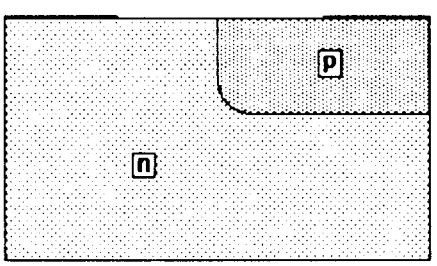

Fig. 2. Example geometry.

Table 1: Execution time for different methods in ELLPACK the number of grid-points is 20x14 (non-uniform)

DISCRETIZATION	INDEXING	SOLUTION	EXECUTION TIME (seconds) DISCR.	INDEX.	SOLUT.
HERMITE COLLOC.	——	BAND GE	92.8	---	191.6
5-POINT STAR	——	BAND GE	23.0	---	8.7
5-POINT STAR	MIN.DEGREE	SPARSE LU UNCOMPR.	24.5	0.6	2.5
5-POINT STAR	MIN.DEGREE	SPARSE GE NO PIVOT.	23.8	0.6	2.3
5-POINT STAR	——	SOR	23.7	---	11.0
5-POINT STAR	RED-BLACK	JACOBI CG	no convergence		

Table 1.

In Table 1, the execution time for the specified problem using the different numerical algorithms is listed. From Table 1 we can conclude that

(1) HERMITE COLLOCATION requires a long execution time. For a $NGRX * NGRY$ grid, we obtain 4 $NGRX * NGRY$ collocation points and 4 $NGRX * NRGY$ equations. That is the reason why HERMITE COLLOCATION is so inefficient.

(2) Iterative solution techniques are not suitable for our problem. JACOBI CG does not converge at all, and SOR takes longer execution time due to slow convergence. Usually, iterative solution techniques are very suitable to solve systems of symmetric – or at least nearly symmetric –, positive definite linear equations. This condition is frequently satisfied for Poisson equations. However, when a boundary condition of the *von Neumann type* is specified, the corresponding discrete equations do not keep their symmetry.

(3) Among the direct solving methods, the two techniques: SPARSE LU UNCOMPRESSED and SPARSE GE NO PIVOTING require less execution time. The MINIMUM DEGREE indexing algorithm takes only very little additional time, and is thus worth the effort. In this example, the difference between the general methods and the methods using sparse matrix techniques is not yet very prominent. However, it can be predicted that the sparse methods for solving the linear equations will be favored much more evidently when the number of grid points is increased.

We also compared the computational values of the electric potential obtained by the different algorithms. It was found that the electric potential differs less than $1.E-4$ volts between the different techniques used.

5.2. *Effect of grid width on the results*

When the number of grid points is varied, both the execution time and the storage requirements of different direct solving methods are influenced heavily. Our results are listed in Tables 2 and 3. From Tables 2 and 3, we can conclude the following facts:

Table 2: Discretization time and solution time with different number of grid points

NUMBER OF GRID POINTS	EXECUTION TIME (seconds) DISCRETIZATION	BAND GE	SPARSE GE NO PIVOT.
20x14	24.0	8.7	2.3
39x14	83.0	58.0	5.7
39x24	247.0	100.0	16.3

Table 2.

(1) Discretization time and solution time grow rapidly with increased number of grid points.
(2) SPARSE GE NO PIVOTING requires the least execution time and the least storage.
(3) With SPARSE GE NO PIVOTING, the discretization time is twelve times longer than the solution time. Therefore, the key to reducing the execution time is to reduce the discretization time.

We also compared the value of the electric potential as a function of the number of grid points. The maximum difference between the values found for the electric potential was below 0.15 volts.

5.3. *Comparison of execution time between* ELLPACK and our own handwritten test program

In order to investigate possibilities for reducing the execution time, we developed another program which we from now on shall refer to as the 'test program'. This program was coded from scratch except for the linear system solver module which we borrowed from the LINPACK [3] library. We used the module LINPACK BAND for this purpose. All other parts of the test program we coded in straight FORTRAN. For discretization, we used a finite difference approach (5-point star), and for the Newton iteration, we used eq. (13). However, contrary to the ELLPACK solution, most of the discretization work was done only once during initialization rather than repetitively

Table 3: Storage requirements with different number of grid points

NUMBER OF GRID POINTS	STORAGE REQUIREMENTS BAND GE	SPARSE LU UNCOMPRESSED	SPARSE GE NO PIVOTING
20x14	16554	15683	8960
39x14	61880	30579	17474
39x24	108290	52416	29954

Table 3.

Table 4 : Number of iterations and total execution time with 20x14 grid points (non-uniform)

REVERSE BIAS (volts)	NUMBER OF ITERATIONS (*)	EXECUTION TIME (seconds)
5	10	306
20	12	367
50	22	656
100	31	918

Table 4.

during each step of Newton iteration. Only minor portions had to be included into the iteration loop. As the discretization proved to be the time critical portion of the algorithm, this modification was expected to lead to significant savings in execution time.

The execution time of our test program for 20×14 (non-uniform) grid points was found as listed below:

Initial part of the discretization	3.9 (seconds)
Iterative part of discretization	0.5
LINPACK band	11.0

Comparing these results with those listed in Table 2, it shows clearly that (for our problem) ELLPACK can handle the set of resulting linear equations very efficiently, whereas its overhead during the discretization process is unacceptable. Nevertheless, we do not regret at all to have used the toolkit first. ELLPACK turned out to be *very* robust, and allowed us to compare different algorithms much more effectively than we could have done otherwise. By doing so, we gathered a pretty clear picture about the numerical properties of our problem. So far, we have merely *prototyped*. Now, we can start writing *production code* with the confidence that the chosen path shall prove successful.

5.4. The influence of varying reverse bias voltage on junction

In Table 4, we summarize the number of Newton iteration steps to convergence together with the overall execution time as a function of the reverse bias voltage. From Table 4, it becomes evident that the number of Newton iteration steps required for convergence increases rapidly with growing reverse bias voltage on junction. Therefore, the execution time grows drastically. To reduce the overall execution time, we can either try to reduce the execution time of a single iteration step, or we can try to find a better initial guess for the solution to cut down the number of iterations to convergence. As we already have optimized the cost of a single iteration (by selecting the best numerical algorithm available), there remains the second type of saving to be discussed. What we could try is to *further simplify* the model, either until there exists a cheaper algorithm to solve the so simplified problem, or until the simplified problem has an analytical solution. This solution could then be used as an improved initial guess for the solution of the

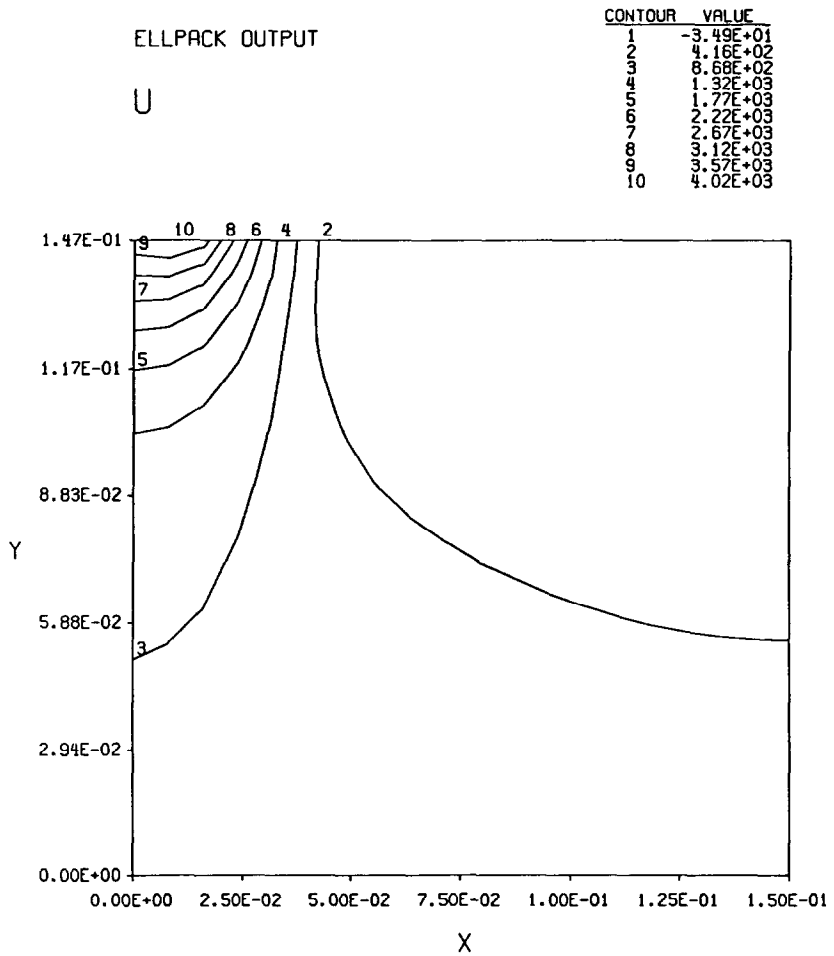

Fig. 3. Contour-plot of the electric potential distribution of a p–n junction with 100 volts reverse bias, calculated with 20×14 non-uniform grid points.

more accurate model described in this paper. One straightforward approach might be to use initially a *smaller number of grid points* than that dictated by the required accuracy to obtain a (cheaper) solution as an initial guess for the more accurate solution of the problem with more grid points. Eventually, the number of grid points could even be gradually increased during the Newton iteration. We hope to be able to automate this process to such an extent that the *grid width selection algorithm* can be included into ELLPACK as an additional module. However, this part of the program has not yet been completed.

Figure 3 depicts a contour plot of the electric potential distribution obtained by using the ELLPACK program. CONTOUR PLOT is an option directly available within the ELLPACK software. Figure 4 graphs the same electric potential distribution as a three-dimensional plot with hidden lines removed. For this purpose, we generated a file from within a FORTRAN segment of the ELLPACK program. This file is then used as input to the graphics processor of the DARE-INTERAC-

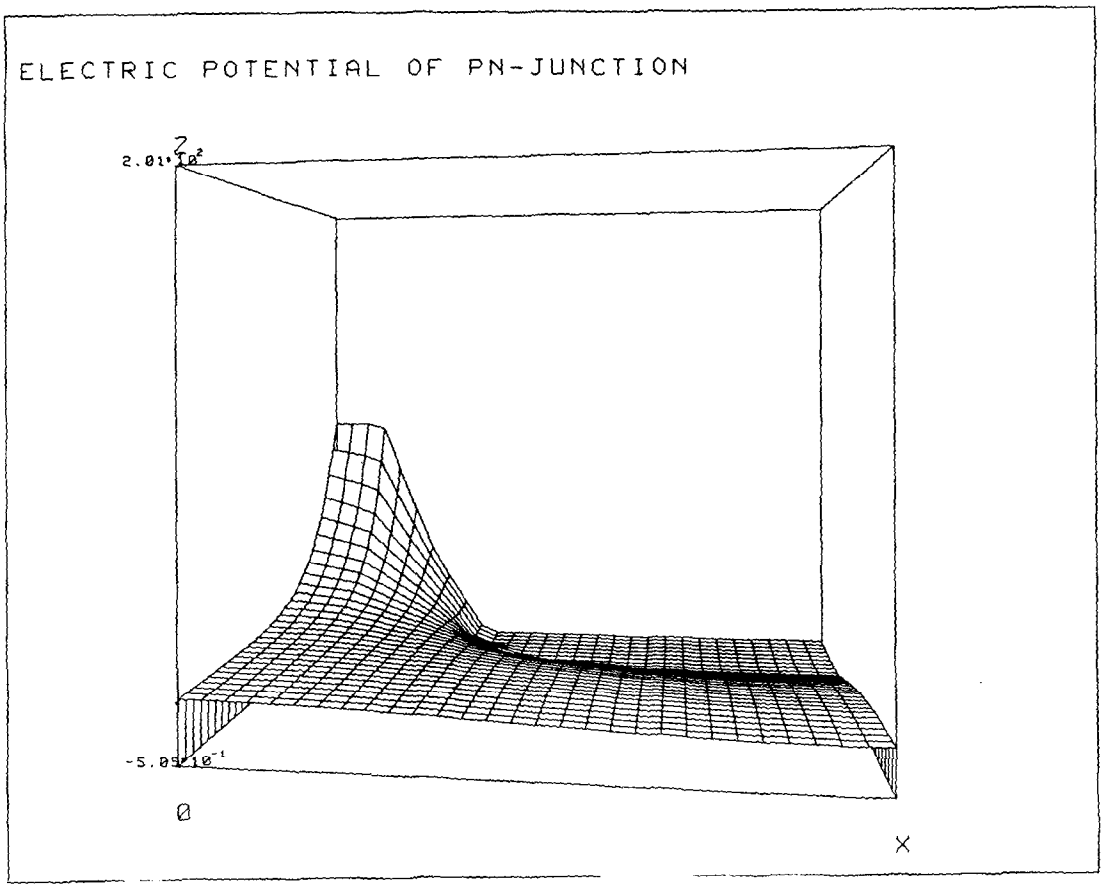

Fig. 4. Three-dimensional plot with hidden lines removed of the electric potential distribution of a p–n junction with 100 volts reverse bias, calculated with 20×14 non-uniform grid points.

TIVE [1] simulation software. The reverse bias voltage here was 100 volts, the number of grid points was 20×14 (non-uniform).

Conclusions

Several approaches have been taken to simulate bipolar high-voltage devices in the neighborhood of breakdown.

In a first attempt, we tried to use the BAMBI [4] program. BAMBI is a special purpose static device simulator. The device is described to BAMBI through its topology and doping concentration. For this reason, BAMBI seemed to be the most natural candidate for our task. Unfortunately, this approach failed totally as BAMBI was too much 'packaged up' to be adapted to our needs. BAMBI has a very simple input description language, thus whenever applicable, BAMBI is a very convenient tool to use. However, BAMBI seems to be somewhat overspecialized. Its flexibility is very limited. In particular, BAMBI seems to work acceptably well (though kind of slow) on *low*

voltage devices, whereas the computing time grows unacceptably large when the reverse bias voltage is increased. Very often, BAMBI even fails to converge when the reverse bias voltage is made sufficiently large.

Our second approach was to use ELLPACK [9]. Compared to BAMBI, ELLPACK is much less specialized. The device is described by means of partial differential equations, and even the solution technique must be at least selected, partially even programmed by the user. For this reason, ELLPACK is much more difficult to use. However, ELLPACK proved extremely useful for our task. The software is very robust. ELLPACK represents a very good compromise between convenience and flexibility. ELLPACK was used to find the appropriate model to be used for our simulation, as well as the best numerical algorithm for the task. However, ELLPACK is still too slow for actual production runs.

The third approach was to write a program of our own. This program was written from scratch except for the linear system solver for which the LINPACK [3] software was used. Preliminary results are very satisfactory. To further increase the speed of this program, we shall try to find a better initial guess for the solution by using a further simplified model. We shall also re-implement the linear system solver by making use of an *array processor* hooked to a VAX/11-780. A special-purpose user interface will be coded which shall allow to design the device interactively on a *PC-class machine*, and download the preprocessed code for further processing to the VAX. It must also be possible to use the output of SUPREM [7] simulations directly as input to the device simulator.

Acknowledgment

This work was performed under a Research Contract from Burr Brown Corp. The authors would like to express their gratitude to Burr Brown for their generous technical and financial support of this project.

References

[1] F.E. Cellier, Enhanced run-time experiments for continuous system simulation languages, *Proc. SCS Multiconference on Continuous System Simulation Languages*, San Diego, CA (1986).

[2] P.E. Cottrell and E.M. Buturla, Two-dimensional static and transient simulation of mobile carrier transport in a semiconductor, *Proc. NASECODE I Conference* (Boole, Dublin, 1979) 41–64.

[3] J.J. Dongarra, J.R. Bunch, C.B. Moler and G.W. Stewart, LINPACK User's Guide, Society for Industrial and Applied Mathematics, 1979.

[4] A.F. Franz and G.A. Franz, BAMBI 1.0 User's Guide, Internal Report, Institut für Allgemeine Elektrotechnik und Elektronik, Abt. Physikalische Elektronik, Techn. Universität Wien, 1985.

[5] A.F. Franz, G.A. Franz, S. Selberherr and P. Markowich, Finite boxes—A generalization of the finite-difference method suitable for semiconductor device simulation, *IEEE Trans. Electron Devices 30* (1983) 1070–1082.

[6] H.K. Gummel, A self-consistent iterative scheme for one-dimensional steady state transistor calculations, *IEEE Trans. Electron Devices 11* (1964) 445–465.

[7] S.E. Hansen, SUPREM-III User's Manual, Internal Report, The Board of Trustees of Stanford University, 1982.

[8] K. Hwang and D.H. Navon, Breakdown voltage optimization of silicon p-π-ν planar junction diodes, *IEEE Trans. Electron Devices 31* (1984) 1126–1135.

[9] J.R. Rice and R.F. Boisvert, *Solving Elliptic Problems Using ELLPACK* (Springer, New York, 1985).

[10] S. Selberherr, *Analysis and Simulation of Semiconductor Devices* (Springer, New York, 1984).