SC5271.9SA

Copy No. 4

A093843

12

NUMERICAL METHODS FOR 2-DIMENSIONAL PROCESS MODELING

W.D. Murphy and W.F. Hall
Rockwell International Science Center
P.O. Box 1085
Thousand Oaks, California 91360

and

C.D. Maldonado and S.A. Louie
Rockwell International Microelectronics R&D Center
3370 Miraloma Avenue
Anaheim, California 92803

82

January 1982

Semi Annual Technical Report No. 3
for Period 1 July 1981 - 31 December 1981

Sponsored by

Defense Advanced Research Projects Agency (DoD)
DARPA Order No. 3984
Under Contract No. MDA903-80-C-0498
Issued by
Department of the Army, Defense Supply Service-Washington
Washington, D.C. 20310

Rockwell International
Science Center

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER / 2. GOVT ACCESSION NO. AD-A110 431	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NUMERICAL METHODS FOR 2-DIMENSIONAL PROCESS MODELING (14)	5. TYPE OF REPORT & PERIOD COVERED Semi-Annual Technical Report 07/01/81 — 12/31/81
	6. PERFORMING ORG. REPORT NUMBER SC5271.9SA
7. AUTHOR(s) W.D. Murphy, W.F. Hall, C.D. Maldonado, and S.A. Louie (15)	8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0498
9. PERFORMING ORGANIZATION NAME AND ADDRESS Rockwell International Science Center P.O. Box 1085 Thousand Oaks, California 91360	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 3984
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Project Agency 1400 Wilson Boulevard Arlington, Virginia 22209 (TIO/Admin.)	12. REPORT DATE January, 1981
	13. NUMBER OF PAGES 75
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Department of the Army Defense Supply Service-Washington Washington, D.C. 20310 82	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

very large scale integration (VLSI), diffusion, solid state, electron devices

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Detailed documentation for a user's version of the 2-D process modeling code developed under the present contract has been completed, and is provided as an appendix to this report. This code, which has been named MEMBRE (for Multidimensional Efficient Moving Boundary Redistribution), will follow a single dopant type through multiple process cycles typical of MOSFET fabrication, including nonuniform growth of a field oxide. Input of data is made in the SUPREM format, utilizing as far as possible

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

389 949

subroutines from SUPREM. Simulation of dopant redistribution during a complete fabrication schedule typically requires a few minutes of CPU time on the IBM 3033 or 370/168. A CDC version of the code runs approximately two-and-one-half times faster on the Cyber 176.

SUMMARY

This technical report contains the initial issue of documentation designed to support the use of the two-dimensional process modeling code MEMBRE (for Multidimensional Efficient Moving Boundary Redistribution), developed under Contract MDA903-80-C-0498, DARPA Order No. 3984. This documentation and the associated software modifications were completed during the third six-month period of performance on this contract. During the first year, a fast algorithm for computing dopant spread in two dimensions was found, a code incorporating nonuniform motion of the oxide-silicon boundary was written around this algorithm, and the code was thoroughly tested for speed and accuracy on the full range of annealing and oxidizing process steps. In the current period, an IBM version of the code has been written, incorporating some forty nine input/output subroutines from SUPREM, and allowing multiple cycles of dopant redistribution to be calculated in a single pass.

In its present form, MEMBRE deals only with oxidation/drive-in, ion implantation, and etching steps. Release of this code within the integrated circuit fabrication community is planned to demonstrate the utility of fast turnaround 2-D modeling for VLSI process design, and thereby to encourage the development of a complete 2-D process simulator.

TASK OBJECTIVES

The overall objective of this program is to develop fast and accurate methods for computer modeling of the two-dimensional spread of dopants and other defects during VLSI circuit fabrication. Our goals for the first year were to demonstrate a fast algorithm for calculating nonlinear diffusion of a single dopant during nonuniform oxide growth, and to provide this algorithm in a form suitable for incorporation into a general process simulator such as Stanford's SUPREM. These goals have been accomplished.

The specific objectives for the second year include:

1. Effective transfer of the basic algorithm to the integrated circuits community;
2. Extension of the code to treat multiple interacting species and three-dimensional redistribution; and
3. Exploration of the computational requirements posed by better physical models for the underlying processes of chemical reaction and defect generation and migration.

The documentation contained in this report represents a significant step in achieving transfer to the integrated circuits community. The speed and potential utility of this method have been publicized at two recent international meetings, resulting in several requests for copies of the basic code. In January, the process/device modeling group at Stanford will receive this code for their consideration as an augmentation of SUPRA.

TECHNICAL PROBLEM

The fabrication of VLSI devices requires production of features of submicron size and separation. Electrical characteristics such as threshold and punchthrough voltages will be sensitive to dopant spread into critical areas adjacent to the original features. Experimental control of this spreading, without guidance from accurate computer modeling, will be costly, tedious, and time-consuming. However, the use of standard numerical methods to achieve an adequate modeling capability is also costly and time-consuming. One should therefore seek advanced methods, drawn from areas such as fluid dynamics, where considerable effort and ingenuity have been expended in recent years to develop fast and accurate solvers for the characterization of multidimensional, time-dependent phenomena.

GENERAL METHODOLOGY

Based upon our own ongoing research in computational nonlinear aerodynamics, we identified several promising approaches to the development of a fast solver for two-dimensional diffusion problems. After a preliminary screening, a few of these were selected for adaptation to the problem of dopant spread during oxidation or annealing. These algorithms were tested for speed and accuracy on the problem of nonlinear dopant diffusion into the channel region of a MOSFET, as well as on simpler problems for which the actual dopant profiles could be accurately obtained by other means.

The algorithm finally selected for further development provided not only exceptional speed, but also a natural extendability to interacting defect species and three-dimensional diffusion. Test cases to explore the speed and utility of the algorithm in such applications are presently being formulated.

TECHNICAL RESULTS

The process modeling code MEMBRE for which the documentation is provided in the Appendix following is a first step toward computer-aided 2-D process design. The spread of a single dopant species through multiple process cycles, from implantation through nonuniform growth of a field oxide, is calculated by MEMBRE at a fixed number of grid points, according to specifications provided by the user in the standard SUPREM format. Run times on the IBM 3033 or 370/168 for a complete fabrication schedule are typically a few minutes. Examples of redistribution of field, channel, and source/drain implants in a MOSFET structure are given at the end of the Appendix.

IMPORTANT FINDINGS AND CONCLUSIONS

The speed with which MEMBRE can predict the effect of process conditions on 2-D dopant spread should make the code a useful tool in the iterative design of VLSI fabrication processes. Many of the most common features in MOSFET fabrication fall within the modeling capabilities of the present code. However, it should be remembered that this code is intended only as a demonstration of 2-D modeling capabilities. It is not a complete process simulator, nor has it been optimized for its present use. Rather, it is presented in a format designed to encourage adaptation and extension. The basic algorithm, whose software implementation is included in MEMBRE, is capable of solving problems of much greater complexity.

SPECIAL COMMENTS

The release of MEMBRE to the integrated circuits industry will begin shortly. However, requests from firms outside the United States pose a problem: the Department of Defense, which has funded this work, requires that something of comparable value be provided to DoD in return. Defining appropriate items for exchange to which DoD presently does not have access is likely to take considerable time. A draft agreement is being prepared for users within the U.S. to guard against the uncontrolled re-release of MEMBRE, and to assure DoD access to improvements and extensions of the present code that may be generated by the users.

IMPLICATIONS FOR FURTHER RESEARCH

There are at least three directions in which the present work can usefully be extended: development of a complete 2-D process simulator, inclusion of interactions between different defect species, and refinement of the numerical methods to include such options as variable and adaptive gridding. The first of these, and to some extent the last, would be appropriate for a dedicated process modeling group such as that at Stanford. In the second category are included such phenomena as electrical interactions of boron and arsenic, oxidation-enhanced diffusion, and concentration-dependent oxidation. Within VLSI structures of submicron size, these effects may produce serious consequences. During the remainder of the present contract, it is our intention to investigate the computational requirements posed by these phenomena, and to test for this application the capability of the fast algorithm on which MEMBRE is based.

APPENDIX

SOFTWARE FOR VLSI PROCESS MODELING IN TWO SPATIAL VARIABLES WITH NONUNIFORMLY MOVING BOUNDARIES – MEMBRE*

W.D. Murphy**	(805) 498-4545, Ext. 130 COMNET 253-2130
C.D. Maldonado†	(714) 632-8111, Ext. 5866 COMNET 252-5866
W.F. Hall**	(805) 498-4545, Ext. 189 COMNET 253-2189
S.A. Louie†	(714) 632-8111, Ext. 5672 COMNET 252-5672

*This work was supported in part by the Defense Advanced Research Projects Agency under Contract MDA903-80-C-0498.

**Rockwell International Science Center, P.O. Box 1085, Thousand Oaks, CA 91360.

†Rockwell International Microelectronics R&D Center, P.O. Box 4761, Anaheim, CA 92803.

CONTENTS

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

PREFACE

During the Spring of 1976, two of the authors of this report – C.D. Maldonado and W.D. Murphy – got interested in developing a code in one dimension for the nonlinear thermal redistribution of boron impurities in SOS device structures. Although this process modeling code, called REPAC (Rockwell Electronics Process Analysis Code), was successful in producing accurate redistributions of ion-implanted profiles, it did not have the nice input-output features of Stanford's SUPREM code, and, consequently, never received the universal usage that the latter code enjoys. Nevertheless, a great deal was learned from solving many LSI problems with REPAC. In particular, the grid size necessary to obtain accurate results both with experimental data and linear analytic solutions was thoroughly investigated. Also, it was observed that much more stable solutions with coarser grids were possible if the moving boundary value problem was mapped into a fixed one. This translation-stretching transformation is obvious in one dimension, but slightly more complicated in two.

Although REPAC solves essentially the same diffusion equation as does SUPREM, the numerical procedure is completely different, a fact that is very important in two dimensions. The numerical procedure is the classical method of lines with the resulting system of stiff ordinary differential equations solved by a robust, automatic variable step-and-order integrator that is optimally first to fifth order. The user need only specify an error tolerance, and the integrator automatically guarantees that the time-integration step size satisfies this criterion. A similar feature is built into the two dimensional code. Some of the details of this technique are described in "Nonlinear Thermal Redistribution of Boron Impurities in SOS Device Structures," by C.D. Maldonado and W.D. Murphy, J. Appl. Phys., 49 (9), 1978.

Our hope in undertaking the present effort was that a means could be found for solving the two-dimensional redistribution problem in times short enough that the code could be used, much as SUPREM is used, to iteratively design fabrication schedules for VLSI chips. Three approaches came to mind immediately:

(1) High order Galerkin B-splines, an extension of the method used in Ref. [2].

(2) Approximate factored schemes (AF) of Beam and Warming.

(3) Modifications of the method of lines to handle two spatial variables with moving boundaries.

Our first attempts to use B-splines in two dimensions proved not to be efficient enough because of the large banded matrices that were generated. Furthermore, we did not always get the improved accuracy from higher order B-splines because the rapidly changing implanted profiles require a minimum number of grid points to resolve the fields. Finally, initial conditions were more difficult to approximate by least squares and B-splines than when using the method of lines.

Although AF schemes have wide use in fluid dynamics because of the simple tridiagonal or block tridiagonal matrices that arise and the absence of any requirement to iterate at each time level, the major deficiencies seem to be that no efficient algorithm for changing the size of the step has been developed, and the method reduces to first order in time when nonuniform time steps are employed. Thus, it may be possible to solve each time level efficiently using AF schemes, but, possibly, too large a number of time steps may be required when integrating several processing cycles (large amounts of real time).

It was clear from our earlier one-dimensional work that the moving boundaries in two dimensions should be mapped into a rectangle. C.D. Maldonado extended the translation-stretching transformation to two spatial variables, but the mapped diffusion equation is no longer self-adjoint (a cross derivative term appears). This term must be handled carefully by the numerical procedure and finer grids are required to accurately approximate this term. W.F. Hall showed that the self-adjointness is preserved under an orthogonal or conformal transformation and developed one which maps the moving problem into an infinite strip. Unfortunately, the derivatives of this conformal transformation are singular at $t = 0$, and consequently, the numerical procedure is slow for $0 \leq t \leq .01$. In addition, the complex coefficients require more computer time than the simple ones that occur when translation-stretching is employed. On the other hand, self-adjoint operators allow for larger grids and error tolerances than non self-adjoint ones. Thus, for large real time redistributions the conformal approach may be superior, but for our test cases ($0 \leq t \leq 2$ hours) the translation-stretching algorithm was two to three times faster. Therefore, this report documents only the latter procedure.

If one extends the method of lines directly to two dimensions, one notices that large banded matrices arise when solving the stiff ordinary differential equations. Using general banded matrix solution techniques results in both excessive computer time and storage allocation. However, A.C. Hindmarsh[8] pointed out to us that his code, GEARBI, makes use of SOR to handle the Jacobian matrix when it has a regular block structure, and suggested to us that such a technique might be useful for VLSI problems.

Unfortunately, the cross-derivative terms from the translation-stretching transformation when discretized yield a Jacobian matrix that doesn't exhibit Property A[7], so the SOR algorithm would fail to converge. However, an approximate Jacobian missing the cross derivative is quite satisfactory, since the Jacobian only acts to accelerate convergence in a Newton-like iterative technique. This approximate Jacobian has five non-zero diagonals and is well suited for SOR. Using these ideas, W.D. Murphy developed a VLSI code first for a drive-in problem and then for the full moving boundary case which uses GEARBI as its main integrator. For a single process cycle, the computer execution times on the CDC 176 are usually under 30 seconds and are much better than we originally thought was possible. Furthermore, CPU time and storage requirements are "roughly" linear with the number of spatial grid points. This is a great improvement over banded matrix techniques, which require time and storage proportional to $L(M)^3$ and to $LM(3M+1)$, respectively, where L is the larger and M the smaller dimension on the computational grid. In addition, extensions to coupled species diffusion and to three spatial variables are easily accomplished with this approach.

Our primary objective in this effort was to provide the integrated circuits industry with a practical means of extending process design to two dimensions. The algorithm described above meets this need. There remains, however, the task of incorporating these numerical methods into a complete process modeling code. As the first major step toward this goal, S.A. Louie has taken a number of the input-output routines in SUPREM and interfaced them with an IBM version of the two-dimensional code. IBM, rather than CDC, was selected because the former machine is compatible with equipment available to the process/device simulation group at Stanford as well as many of the current users of SUPREM.

S.A. Louie has extended the IBM version so that the following process cycles can be computed:

(a) oxidation/drive-in

(b) ion implantation

(c) etching .

Only slight modifications were made to SUPREM input routines to make them compatible with our two-dimensional code, thereby making it relatively easy to use by engineers familiar with the Stanford code.

Our two-dimensional process simulation code with SUPREM input routines has been named MEMBRE (Multidimensional Efficient Moving Boundary Redistribution Computer Program). It has been written and documented in such a way that changes to the diffusion function, growth model, and initial implant can be easily carried out. It is our hope that MEMBRE will be as useful a tool for the design of VLSI process cycles as SUPREM has been for LSI ones.

1.0 INTRODUCTION

The class of problems which is solvable with MEMBRE can be characterized as nonlinear dopant redistribution within a silicon substrate whose surface may be simultaneously undergoing nonuniform oxidation. In addition, the usual implantation and oxide stripping steps available in SUPREM have been incorporated with appropriate extensions to two dimensions.

The user may specify his own models for diffusion and oxidation, or may rely on those provided as part of MEMBRE. Multiple cycles of implantation, oxidation, and annealing can be run without restarting the calculations, although at present all cycles would have to share the same spatial grid. For simplicity, MEMBRE has been set up for separate uniform intervals in the two spatial dimensions.

Interactions between different dopant species have not been included in this code. While there is no difficulty for the basic algorithm in handling such cases, considerable additional programming would have been required to provide and specify the various options.

The geometry for the general process step modeled by MEMBRE is illustrated in Figure 1 for an SOS structure. Because the device is symmetrical about the axis $y = 0$, only the shaded region of Figure 1(a) is used in formulating the problem. The window opening ($|y| < a$) in the impenetrable mask allows the implant to penetrate an initial, thin layer of oxide and distribute itself two-dimensionally within the substrate. The growth of a field oxide, with lateral penetration of the bird's beak under the mask edge, is shown in Figure 1(b). During this step, which includes segregation of dopant across the oxide boundary as well as diffusion within the substrate, the upper boundary of the substrate moves downward in a nonuniform fashion. A simple annealing step can of course be modeled by immobilizing this boundary.

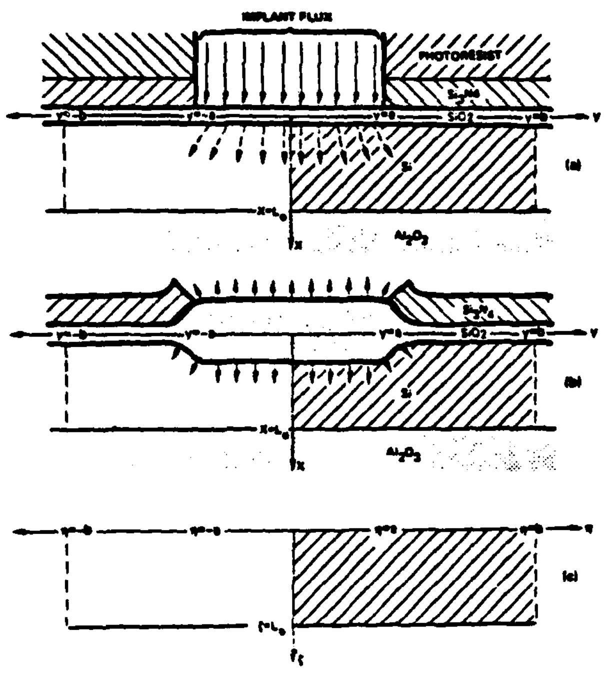


Figure 1.
Initial SOS geometry and corresponding transformed geometry that takes place during the growth of a field oxide are shown in (a) and (b), respectively; while in (c) is shown the stationary rectangular (shaded region) computational domain in the transformed (ξ,η) coordinate system.

FORMULATION

The present code provides an analytic form for the two-dimensional implant distribution N(x,y,0) taken from Furukawa, et al.[1]:

$$N(x,y,0) = \frac{N_d \times 10^4}{2\sqrt{2\pi}\,\sigma_p} \exp\left[-\frac{(x-R_p)^2}{2\sigma_p^2}\right]\left\{\mathrm{erf}\left[\frac{(y+a)}{\sqrt{2}\,\sigma_L}\right] - \mathrm{erf}\left[\frac{(y-a)}{\sqrt{2}\,\sigma_L}\right]\right\}, \quad (1)$$

where N_d is the dose, R_p the projected range, and σ_p and σ_L are the projected and lateral standard deviations, respectively. Values for these parameters are either read in or calculated from other input data such as implant energy and type.

Redistribution of the implant within the silicon substrate is modeled as diffusion with a concentration-dependent diffusivity D(N):

$$\frac{\partial N}{\partial t} = \nabla \cdot [D(N)\nabla N] \ . \qquad (2)$$

Boundary conditions at the line of symmetry ($y = 0$) and at the back of the substrate ($x = L_0$) are taken as zero dopant flux normal to these surfaces. The same condition has been enforced at the right-hand edge of the computational region ($y = b$). At the moving boundary, particle conservation requirements combined with an equilibrium segregation relation lead to the condition

$$\vec{n} \cdot D\nabla N = \vec{n} \cdot \hat{x}(k - m)(\partial U/\partial t)N \ , \qquad (3)$$

where U is the local oxide thickness (along $\hat{x}$), k is the segregation coefficient, m is the ratio of silicon thickness consumed to oxide thickness produced, and $\vec{n}$ is the local normal at the oxide-substrate boundary.

The dependence of the diffusivity on dopant concentration is taken as[2]

$$D(N) = \left\{\frac{1 + \delta[\gamma + \sqrt{\gamma^2 + 1}]}{1 + \delta}\right\}\left[1 + \frac{\gamma}{\sqrt{\gamma^2 + 1}}\right]D_0 \ , \qquad (4)$$

where D_0 is the intrinsic diffusion coefficient for the given dopant in silicon, $\gamma = N/2n_i$ with n_i being the intrinsic carrier concentration, and δ is a dopant-dependent fitting parameter ($\delta = 19$ for boron; $\delta = 100$ for arsenic).

TRANSLATION-STRETCHING TRANSFORMATION

The oxide boundary of the substrate region may be stationary or moving during a given process step; the other three substrate boundaries remain fixed. In order to simplify the numerical integration procedures, it was found desirable to map this region onto a rectangle of constant dimensions. The specific mapping implemented in the present code locally scales the depth dimension to a fixed length L_I and leaves the lateral dimension untouched. Thus, denoting by $x_0(y,t)$ the depth of the oxide boundary at position y and time t, the transformation

$$\xi = \left(\frac{x - x_0}{L_0 - x_0}\right) L_I \, , \tag{5a}$$

$$\eta = y \, , \tag{5b}$$

$$\tau = t \, , \tag{5c}$$

to computational coordinates (ξ,η,τ) is used. For a single oxidation step one has $x_0 = mU$, and the diffusion equation (2) in the new coordinates takes the form:

$$\frac{\partial N}{\partial \tau} = \left[\frac{L_I^2 + m^2U_\eta^2(L_I - \xi)^2}{L^2}\right] \frac{\partial}{\partial \xi}\left[D\frac{\partial N}{\partial \xi}\right] + \frac{\partial}{\partial \eta}\left[D\frac{\partial N}{\partial \eta}\right]$$
$$- \frac{2mU_\eta}{L}(L_I - \xi)\frac{\partial}{\partial \eta}\left[D\frac{\partial N}{\partial \xi}\right] + \frac{m(L_I - \xi)}{L^2}[U_\tau L - (U_{\eta\eta}L + 2mU_\eta^2)D]\frac{\partial N}{\partial \xi} \, , \tag{6}$$

where $L = L_0 - x_0$ and U_α denotes the partial derivative $\partial U/\partial\alpha$.

The boundary conditions at the sides of the rectangle are modified due to the non-orthogonality of the transformation (5), taking the form:

$$- m(L_I - \xi)\left[\frac{U_\eta}{L}\frac{\partial N}{\partial \xi}\right] + \frac{\partial N}{\partial \eta} = 0 \tag{7}$$

at $\eta = 0$ and $\eta = b$ for all ξ. The condition at the back of the substrate is unchanged:

$$\partial N/\partial \xi = 0 \quad \text{at} \quad \xi = L_I \,, \tag{8}$$

while at the oxide boundary ($\xi = 0$) one has

$$\frac{L_I}{L} D \frac{\partial N}{\partial \xi} + \frac{m^2 U_\eta^2}{L} L_I D \frac{\partial N}{\partial \xi} - m U_\eta D \frac{\partial N}{\partial \eta} = (k - m) U_\tau N \,. \tag{9}$$

The model for oxide growth implemented in the present code consists of three separately adjustable regions. In the first region, near the line of symmetry ($0 < \eta < b_1$), an oxidation rate $\dot{U}_I$ given by the Deal-Grove model[3] is assumed:

$$U_I(\tau) = - (\alpha/2) + [(\alpha/2)^2 + \beta(\tau - \tau_\ell) + \alpha U_0 + U_0^2]^{1/2} \,, \tag{10}$$

where τ_ℓ is the time at the beginning of the present oxidation step, α and β are the usual constants of the model, and U_0 is the local oxide thickness at τ_ℓ. In the second region ($b_1 < \eta < b_2$) a form of interpolation is used to simulate the bird's beak (see Penumalli[4]):

$$U(\eta,\tau) = \left[\frac{U_I(\tau) - U_0(\eta)}{2}\right] \mathrm{erfc}\left[\frac{\sqrt{2}\,(\eta - \eta_0)}{K_0 U_I(\tau)}\right] + U_0(\eta) \,, \tag{11}$$

where η_0 and K_0 are parameters chosen to fit experimental data on the shape of the bird's beak. See Section 3 of this Appendix for more details about these two parameters.

In the third region ($b_2 < \eta < b_3$) the oxide thickness is frozen at $U_0(b_3)$.

In the computer code, regions 1 and 3 consist of the first and last two mesh points in the η direction, respectively. Region 2 consists of all other points in between. The code assumes $U_\eta = 0$ in regions 1 and 3.

The oxide growth model for a uniformly moving boundary is characterized by equation (10). Note that if a uniformly moving boundary cycle follows the growth of a bird's beak (equation (11)), then U_0 will be a function of η and one should write $U_I(\tau,\eta)$.

The partial derivatives U_η and $U_{\eta\eta}$ are obtained using centered numerical differentiation formulas while U_τ is obtained analytically.

NUMERICAL PROCEDURE

The numerical procedure is the classical method of lines which employs a robust integrator for solving the resulting system of stiff ordinary differential equations. The computational rectangle is covered by a uniform grid with coordinates (ξ_i,η_j) and meshwidths $\Delta\xi = \xi_{i+1} - \xi_i$ and $\Delta\eta = \eta_{j+1} - \eta_j$. Spatial derivatives in equations (6-9) are discretized using the following centered difference approximations:

$$\frac{\partial N_{ij}}{\partial \xi} \approx \frac{N_{i+1,j} - N_{i-1,j}}{2\Delta\xi} \tag{12a}$$

$$\frac{\partial N_{ij}}{\partial \eta} \approx \frac{N_{i,j+1} - N_{i,j-1}}{2\Delta\eta} \tag{12b}$$

$$\frac{\partial}{\partial\xi}\left[D(N)\,\frac{\partial N}{\partial\xi}\right]_{ij} \approx D_{i+\frac{1}{2},j}\,\frac{(N_{i+1,j}-N_{ij})}{(\Delta\xi)^2} - D_{i-\frac{1}{2},j}\,\frac{(N_{ij}-N_{i-1,j})}{(\Delta\xi)^2} \tag{12c}$$

$$\frac{\partial}{\partial\eta}\left[D(N)\,\frac{\partial N}{\partial\eta}\right]_{ij} \approx D_{i,j+\frac{1}{2}}\,\frac{(N_{i,j+1}-N_{ij})}{(\Delta\eta)^2} - D_{i,j-\frac{1}{2}}\,\frac{(N_{ij}-N_{i,j-1})}{(\Delta\eta)^2} \tag{12d}$$

$$\frac{\partial}{\partial\eta}\left[D(N)\,\frac{\partial N}{\partial\xi}\right]_{ij} \approx \frac{D_{i,j+1}(N_{i+1,j+1}-N_{i-1,j+1}) - D_{i,j-1}(N_{i+1,j-1}-N_{i-1,j-1})}{4\Delta\xi\Delta\eta} \tag{12e}$$

where $N_{ij} = N(\xi_i,\eta_j,\tau)$, $D_{i\pm\frac{1}{2},j} = D\left[\frac{1}{2}\,(N_{i\pm1,j}+N_{ij})\right]$, $D_{ij} = D(N_{ij})$, etc. For the boundary conditions we illustrate the basic idea by discretizing equation (9):

$$\frac{L_I}{L}\,D(N_{1j})[1+m^2U_\eta^2]\left[\frac{N_{2j}-N_{0j}}{2\Delta\xi}\right] - mU_\eta D(N_{1j})\left[\frac{N_{1,j+1}-N_{1,j-1}}{2\Delta\eta}\right] = (k-m)U_\tau N_{1j} \tag{13}$$

where the subscript 1j denotes a point on the boundary $\xi = 0$, and 0j represents a point to the left of this boundary. Equation (13) is solved for N_{0j} and the resulting expression is used in equation (12c) when $i = 1$. The remaining boundary conditions are handled in a similar way.

This spatial variable differencing leads to a semidiscrete system of nonlinear ordinary differential equations. The equations corresponding to the interior mesh points have the form

$$\frac{dN_{ij}}{d\tau} = f_{ij}(N_{ij},N_{i-1,j},N_{i+1,j},N_{i,j+1},N_{i,j-1},N_{i+1,j+1},N_{i-1,j+1},$$
$$N_{i+1,j-1},N_{i-1,j-1},\tau)\ . \tag{14}$$

Unfortunately, equation (14) is stiff[5], and consequently, the Jacobian, $\partial f/\partial N$, is needed to converge the corrector equation in the linear multistep method used to integrate it. The Jacobian is used in a Newton-like method having coefficient matrix

$$P = I - \Delta\tau \, \beta_0 \, \partial f/\partial N \qquad (15)$$

where β_0 is a scalar associated with the order of the corrector equation and $\Delta\tau$ is the time meshwidth. Most of the computer time is used in solving a linear system of the form

$$Px = b \, , \qquad (16)$$

and, therefore, it is critical that efficient methods are used for solving equation (16). The successive overrelaxation (SOR)[6] method is a natural technique for solving equation (16), but the Jacobian matrix has nine non-zero diagonals and doesn't exhibit Property A[7]. SOR may be employed, however, if the cross-derivative term (12e) is ignored when evaluating $\partial f/\partial N$ *but not when evaluating f*. If this term were present, the solution of equation (16) would have to be obtained by the relatively slow banded matrix techniques employed by Warner and Wilson[2]. Other advantages of SOR are minimal storage requirements and as P changes from time step to time step, very accurate initial solutions provide fast convergence. In contrast, banded matrix solvers must start from scratch each time step and do not use any previous information.

The numerical solution of (14) is performed by a variable-order and variable-step-size stiff integrator written by Hindmarsh[8], which is optimally first to fifth order accurate in the time variable. Because of the uniform centered differencing in the spatial directions, the difference operators are second order accurate in ξ and η. The strong points of this method are the efficient method for solving equation (16) and the excellent error control of the integrator which allows both the time step and the order of integration to be selected in an optimal manner.

For an extensive study on the use and power of GEARBI, see Murphy[9]. For greater insight into the physical problem and a number of computational results, see Murphy, Hall, and Maldonado[10].

2.0 SUBROUTINE DESCRIPTION

The version of MEMBRE that we are distributing is the IBM double precision one which contains SUPREM interface input routines and is capable of computing the following process steps:

(a) oxidation/drive-in
(b) ion implantation
(c) etching .

Although the CDC single precision version is 2.5 times faster than the IBM one, it can only consider a single process step and does not contain SUPREM input routines.

We describe below those subroutines that are inherent to the two-dimensional process model. SUPREM input routines will be listed afterwards in abbreviated form for quick reference. All subroutines contain a number of comment cards explaining their major function. Our purpose here is not to duplicate all of this information but instead to discuss some of the technical details of each routine.

SUBROUTINE NAME: ASET

PURPOSE: This routine computes the coefficients of the Jacobian, $\partial F/\partial N$ (see equation (14)), where the terms arising from equation (12e) have been ignored. This routine is called by the GEARBI package to accelerate convergence of the corrector equation in a Newton-like scheme. See Refs. [8,9] for more details.

DESCRIPTION: The code is divided into three sections depending upon whether η lies in the interval $[0,b_1]$, $[b_1,b_2]$, or $[b_2,b_3]$, referring to equations (10-11). Indices JB1 and JB2 refer to the point η equal to b_1 and b_2, respectively. Note that U_η is assumed to be zero in regions 1 ($0 < \eta < b_1$) and 3 ($b_2 < \eta < b_3$). Consequently, the Jacobian is greatly simplified here. For functions given by equation (11), regions 1 and 3 will contain only a few points.

The diagonal element of the Jacobian is stored in the vector AJ, while the off diagonal terms are stored in the matrix AA using the following format:

AA(, 1)	contain	$\partial f_{ij}/\partial N_{i-1,j}$
AA(, 3)	contain	$\partial f_{ij}/\partial N_{i+1,j}$
AA(, 2)	contain	$\partial f_{ij}/\partial N_{i,j-1}$
AA(, 4)	contain	$\partial f_{ij}/\partial N_{i,j+1}$.

The unknowns are numbered from left to right and bottom to top, i.e., $N_{1,1}$ is the first unknown and corresponds to the spatial point given by the origin (0,0); $N_{NX,NY}$ is the last point in the upper right-hand corner.

SUBROUTINE NAME: COSET

PURPOSE: This routine is part of the GEARBI package and is used to initialize coefficients in the backward finite difference equations used to solve equation (14).

DESCRIPTION: See Ref. [8].

FUNCTION: D(U)

PURPOSE: This function routine computes the diffusion function given by equation (4). Any other diffusion function may easily be substituted.

DESCRIPTION: Note that COMMON BLOCK /PARM3/ stores a number of constants so that arithmetic operations can be minimized. These variables have the following values:

CETA = δ
DBETA1 = $D_o/(1+\delta)$
DBETA2 = $.5D_o/n_i(1+\delta)$
DBETA3 = $.5\delta D_o/n_i(1+\delta)$
HALFNI = $.5/n_i$.

Note that physically N should never be negative; however, for very coarse grids, round-off errors and poor approximations to equation (12e) may force a few numbers in the bird's beak region to go negative. The code will set the diffusion function equal to D_o for this case. The user should refine his grid if this ever happens.

SUBROUTINE NAME: DEC

PURPOSE: This routine in the GEARBI package uses Gaussian elimination with partial pivoting to upper triangularize a matrix.

DESCRIPTION: See Ref. [11].

SUBROUTINF NAME: DIFFUN

PURPOSE: This routine uses the method of lines to discretize equations (6-9) using the difference expressions given by (12-13). Explicitly, this subroutine computes f_{ij} (equation (14)).

DESCRIPTION: This subroutine is divided into the same three regions as in ASET. UFCT is called to compute $U(\eta,\tau)$ and its derivatives. Various load routines are used to compute the finite difference expressions. Note that if JB1 = NY, only region 1 computations are performed. This case would occur if there is a *uniformly moving boundary* with no previous nonuniformly moving boundary. f_{ij} is stored in the vector YDOT using the same numbering convention as in ASET.

Note that the cross derivative is discretized (equation (12e)) in this routine, but is ignored in ASET. Thus, the difference expression f_{ij} includes the cross derivative, but the Jacobian doesn't. This strategy has the effect of using a slightly inaccurate Jacobian. No appreciable harm is done, however.

SUBROUTINE NAME: DRIVBI

PURPOSE: This is the main driving routine of the GEARBI package.

DESCRIPTION: See Ref. [8].

SUBROUTINE NAME: DU

PURPOSE: This subroutine computes the diffusion function given by equation (4) and its derivative $\partial D/\partial N$. The latter is required for Jacobian computations. Both may easily be changed if the user has a better diffusion function or wants to experiment.

DESCRIPTION: See comments under FUNCTION: D(U).

SUBROUTINE NAME: INIT

PURPOSE: This subroutine computes the initial profile given by equation (1) in the ξ-η space.

DESCRIPTION: Since $U(0,y) = U_0$, the value of x used in equation (1) is given by $x = (L_0 - mU_0)\xi/L_I + mU_0$. The values of N in equation (1) are computed and added to those numbers stored in the solution vector YO. This technique allows for multiple implants by successive calls to INIT or usage of an initial background by presetting Y0 to a constant before entering INIT. This routine may be easily changed to accommodate other possible initial conditions.

SUBROUTINE NAME: INTERP

PURPOSE: This routine is used by the GEARBI package to perform interpolation in the τ direction for the user's required output.

DESCRIPTION: See Ref. [8].

SUBROUTINE NAME: LDNXI1

PURPOSE: This subroutine computes $N_\xi(0,\eta)$ using the boundary description given by equation (13). The Jacobian of this value is also computed.

DESCRIPTION: Only region 2 ($b_1 \leq \eta \leq b_2$) boundary derivatives are computed by this routine. UFCT was called earlier to obtain moving boundary information ($U(\eta,\tau)$, $U_\eta(\eta,\tau)$, $U_\tau(\eta,\tau)$). $N_\xi(0,\eta)$ values are stored in UXI1, while Jacobian information is stored in UXI2.

SUBROUTINE NAME: LOADDX

PURPOSE: This subroutine computes the discretized difference equation in the ξ direction using (12c). Homogeneous boundary condition data is also loaded.

DESCRIPTION: DSAVE(I) contains $\left[\frac{N_{ij} - N_{i-1,j}}{(\Delta\xi)^2}\right] D\left[\frac{1}{2}(N_{ij} + N_{i-1,j})\right]$. Note that by equation (12c) $\frac{\partial}{\partial\xi}\left[D(N)\frac{\partial N}{\partial\xi}\right]_{i,j} \approx \text{DSAVE(I+1)} - \text{DSAVE(I)}$.

SUBROUTINE NAME: LOADDY

PURPOSE: This routine is the same as LOADDX except the difference expressions are formed in the y (or η) direction.

DESCRIPTION: See LOADDX above.

SUBROUTINE NAME: LOADJX

PURPOSE: This routine computes the Jacobian (with respect to N_{ij}) of the difference equations formed in LOADDX above.

DESCRIPTION: The vector DSAVE contains the values $\frac{1}{(\Delta\xi)^2} D\left[\frac{1}{2}(N_{ij} + N_{i-1,j})\right]$ and DUSAVE contains $\frac{1}{2(\Delta\xi)^2} \frac{\partial D}{\partial N}\left[\frac{1}{2}(N_{ij} + N_{i-1,j})\right]\left[N_{ij} - N_{i-1,j}\right]$.

All Jacobian information in the ξ direction can be computed using these two vectors. For example, *ignoring coefficients of the diffusion equation*, we can write

$$\frac{\partial f_{ij}}{\partial N_{i-1,j}} \approx \text{DSAVE(I)} - \text{DUSAVE(I)}$$

and

$$\frac{\partial f_{ij}}{\partial N_{i+1,j}} \approx \text{DSAVE(I+1)} + \text{DUSAVE(I+1)} .$$

SUBROUTINE NAME: LOADJY

PURPOSE: This routine computes the Jacobian of the difference equations formed in LOADDY above.

DESCRIPTION: See LOADJX. Differences are in the η (or y) directions.

SUBROUTINE NAME: LOADNX

PURPOSE: This routine computes $N_\xi(\xi,\eta)$ for $\xi \neq 0$ using the difference expression (12a).

DESCRIPTION: UXI stores the value $(N_{i+1,j} - N_{i-1,j})/2\Delta\xi$. Note for the right boundary $N_\xi = 0$.

SUBROUTINE NAME: LOADYX

PURPOSE: This routine uses the discretization (12e) to form information needed to compute the cross derivative in region 2 $(b_1 \leq \eta \leq b_2)$.

DESCRIPTION: The matrix DER stores the values $\dfrac{D(N_{ij})\left[N_{i+1,j} - N_{i-1,j}\right]}{4\Delta\xi\Delta\eta}$. The cross derivative (12e) may be formed as the difference of two expressions in the DER matrix.

SUBROUTINE NAME: MAIN

PURPOSE: This is the main program for MEMBRE which is a modified version of the SUPREM MAIN subroutine. This subroutine sets data, calls SUPREM inputs, and initializes parameters for the integrators; then it calls PROCESSOR to process the STEP which may be one of the following: (a) initial oxide growth, (b) etch, (c) initial implantation, or (d) drive-in/oxidation.

DESCRIPTION: An initial background is set to $|N_B|$, 10^6, or 5×10^{14} depending upon whether the substrate impurity is B, As, or any other, respectively. CETA (δ in equation (4)) is set to 19 for B and 100 for As. Values for P and Sb could also be set here when they become known, but this may also require the use of a new diffusion function.

The growth model parameters (see SUBROUTINE UFCT and User's Guide to Input Data) are determined from the values of RATO and YPEN. If either is zero, the uniformly moving boundary is assumed (IPASS = 1). In addition, if no previous nonuniformly moving boundary has occurred, JB1 is set to NY, indicating only region 1 calculations are to be performed; otherwise, JB1 and JB2 are unchanged. This decision process is monitored by the variable JPASS. On the other hand, if RATO and YPEN are both greater than zero, the nonuniformly moving boundary is assumed (IPASS = 0) and JB1 and JB2 are set to 2 and NY - 2, respectively. This setting will require that three region calculations will be performed.

The vector MATP is used to supply information to the GEARBI package about the structure of the Jacobian (see comment cards in subroutine DRIVBI).

Variables beginning with the letter R store differencing information used in equation (12).

The step is processed via the modified SUPREM subroutine STPRC, which now contains calls to our subroutines INIT (for initialization of an implant), TWOD (for etching or drive-in/oxidation), and OUTPT (for output).

SUBROUTINE NAME: MESH

PURPOSE: This routine uses equation (5) to transform the mesh from the computation (ξ,η) space to the physical (x,y) one.

DESCRIPTION: UFCT is called to obtain $U(\eta,\tau)$, and equation (5) is employed. Note that this data together with the solution vector is written on TAPE 20. U_0 in equations (10-11) is adjusted for the total accumulation of etched oxide up to this time level, and the results are stored in the vector UYT.

SUBROUTINE NAME: OMEGA

PURPOSE: This routine is called by the GEARBI package to compute the overrelaxation parameter.

DESCRIPTION: See Refs. [6] and [8].

SUBROUTINE NAME: OUTPT

PURPOSE: This subroutine prints the solution N on paper.

DESCRIPTION: The solution is printed along rows of constant y-values from $y = 0$ to $y = b_3$.

SUBROUTINE NAME: PDBD

PURPOSE: This routine loads the diagonal term of the discretized Jacobian.

DESCRIPTION: These diagonal terms have been computed in ASET and stored in the common block /AJ/ AJ(1).

SUBROUTINE NAME: PSETBI

PURPOSE: This routine is called by the GEARBI package to perform a Newton-like iteration on the corrector equation in the multistep method used to solve equation (14).

DESCRIPTION: See Ref. [8].

SUBROUTINE NAME: SOL

PURPOSE: This routine solves the linear system $Ax = b$ using the output of DEC.

DESCRIPTION: See Refs. [8] and [11].

SUBROUTINE NAME: SOLBI

PURPOSE: This subroutine is called by STIFBI and solves a blocked linear system by the block-SOR iterative technique.

DESCRIPTION: See Ref. [8].

SUBROUTINE NAME: STIFBI

PURPOSE: This routine performs one step of the integration of equation (14).

DESCRIPTION: See Ref. [8].

SUBROUTINE NAME: TWOD

PURPOSE: This routine sets the initial oxide growth, calculates the oxide that is stripped for etching, or calls the integration package to perform drive-in/oxidation.

DESCRIPTION: This subroutine has three parts. The first part computes the initial oxide growth and is initiated by a STEP card when the user specifies TYPE = DEPO. In this step, the program calculates the initial oxide thickness U_0 = TIME * GRTE, where TIME and GRTE are inputted from the STEP card.

The second part concerns the etch step. Here the program calculates the amount of oxide stripped and is governed by TIME * ERTE inputted from the STEP card.

The third part is for drive-in/oxidation step. Here the subroutine calls DRIVBI, the main driver routine for the GEARBI package. Note that the parameter EPS is set at 10^{-4} and governs the pseudo relative error tolerance for the integrator. If the user is willing to accept a relative error in the time integration of three significant figures rather than four, then EPS can be set to 10^{-3}. The result will be faster run times.

SUBROUTINE NAME: UFCT

PURPOSE: This subroutine covers two oxide growth models corresponding to the nonuniformly and uniformly moving boundary cases, respectively. This routine computes $U(\eta,t)$ and its derivatives $U_t(\eta,t)$, $U_\eta(\eta,t)$, and $U_{\eta\eta}(\eta,t)$, for these two models.

DESCRIPTION: For the nonuniformly moving boundary case, the formulation is as follows:

$$U(y,t-T_0) = \tilde{U}(y,t-T_0) + U_s \tag{17a}$$

$$\tilde{U}(y,t-T_0) = \frac{Z(t-T_0)}{2}\,\mathrm{erfc}\left[\frac{\sqrt{2}\,(y-y_0)}{K_0\,\tilde{U}(t-T_0)}\right] + \tilde{U}_0 \tag{17b}$$

$$Z(t-T_0) = \tilde{U}(t-T_0) - \tilde{U}_0 \tag{17c}$$

$$\tilde{U}(t-T_0) = -\frac{\alpha}{2} + \left[\left(\frac{\alpha}{2}\right)^2 + \beta[(t-T_0)+t_0]\right]^{1/2} \tag{17d}$$

$$t_0 = \frac{\tilde{U}_0^2 + \alpha\tilde{U}_0}{\beta} \tag{17e}$$

where U_s is the total accumulation of etched oxide prior to time, T_0, K_0 is the ratio of lateral to vertical oxide growths, y_0 is the symmetry position of the complementary error function, and β and β/α are the linear and parabolic rate coefficients of the Deal-Grove model[3].

The oxide growth model for the uniformly moving boundary case is formulated as follows:

$$U(y,t-T_0) = \tilde{U}(y,t-T_0) + U_s \tag{18a}$$

$$\tilde{U}(y,t-T_0) = -\frac{\alpha}{2} + \left[\left(\frac{\alpha}{2}\right)^2 + \beta[(t-T_0)+t_0]\right]^{1/2} \tag{18b}$$

$$t_0 = \frac{\tilde{U}_0^2 + \alpha\tilde{U}_0}{\beta}\,. \tag{18c}$$

The oxide growth model is *uniform* or *nonuniform* depending upon whether the value of the parameter IPASS is greater than zero or equal to zero, respectively.

The last portion of UFCT computes numerically the values of $U_\eta(\eta,t)$ and $U_{\eta\eta}(\eta,t)$ for both models. It should be noted that these partial derivatives may not be zero even for the uniformly moving boundary case because $\tilde{U}_0$ will be a function of η if a previous cycle used the formulation given by equation (17).

Observe that t_0 (TPAST) is updated at the end of each processing step and U_s (USSUM) at each etching step; also, $U_t(\eta,t)$ and $U(\eta,t)$ are updated at the end of each process step.

This routine may be easily changed if the user wishes to employ another UFCT.

SUPREM SUBROUTINE LIST

The following subroutines are taken from SUPREM's input section. A few of the subroutines required minor modification to fit into the two-dimensional code.

1. Subroutine BNDRS
   Determines DZ, DY1, DY2, INTF, IPNT1, NCC to set up grid size.
2. Subroutine COMST
   Handles all comment cards.
3. Subroutine ERSET
   Handles the error processing for input syntax.
4. Subroutine FGRID
   Initializes the grid spacing from the first grid card.
5. Subroutine GDSYA
   Checks syntax of the grid card.
6. Subroutine GETLN
   Gets an input line and processes it.
7. Function GETRL
   Gets a real number from the buffer.
8. Subroutine GRDNW
   Is used for grid modification by the grid card, the etch step, and the low temperature deposition step.
9. Function IFTYP
   Is used to determine file type: 1 → A or ASCII file
   2 → B or binary file .
10. Function IGTEL
    Gets an element value from the buffer.

11. Subroutine IMPLN

    Determines the method of computation for the ion implantation profile parameters.

12. Subroutine INITE

    Initializes values of segregation, surface transport, oxide diffusion, silicon intrinsic diffusion, and silicon oxidization diffusion coefficients.

13. Subroutine IPLNT

    Sets ion implant range, standard deviation, and third moment ratio statistics.

14. Function ISCOM

    Compares a source character string with a comparison string.

15. Function ISHIFT

    Shifts the top byte of the integer word to the bottom byte.

16. Function ITYPE

    Tests for character types. 1:letter, 2:number, 3:signed real number, 4:blank, 5:comma, 6:equal sign, 7:invalid.

17. Function LGTOP

    Gets option value from buffer.

18. Subroutine MDSYN

    Checks model card syntax.

19. Subroutine MODIN

    Sets up the correct model parameters for each step.

20. Subroutine MODNN

    Identifies model card parameters.

21. Subroutine MODST

    Processes the model card statements, determines the model type and number, and extracts the model parameters which are then placed in the correct model array.

22. Subroutine MOVE

    Moves a source string to a destination string.

23. Subroutine OPSYN

    Checks the optimize card for syntax errors.

24. Subroutine OPTST

    Processes the OPTM statements, extracts the parameters, and determines the parameter location (initial, lower, or upper values).

25. Subroutine OXIDI

    Calculates parameters for the oxidation, diffusion, and gaseous predeposition step.

26. Subroutine OXINT

    Initializes the array passed as OXARY with the default oxide growth parameters determined by the oxidizing ambient and the orientation.

27. Subroutine PARSE

    Reads in an input line and determines the parameter values specified in the input line.

28. Subroutine PARTP

    Identifies the OPTM card parameter names.

29. Subroutine PLSYN

    Checks the syntax of the PLOT statement cards.

30. Subroutine PLTST

    Processes the PLOT statement cards and updates the common PLOT variables.

31. Subroutine PRNST

    Processes the PRINT statement cards and updates the common PRINT variables.

32. Subroutine PRSYN

    Checks the syntax of the PRINT statement cards.

33. Subroutine RSLSY

    Checks the RESULT card syntax.

34. Subroutine RSLTS

    Processes the RESULT statement, extracts the parameters and determines the parameter location, the target value, and the tolerance.

35. Subroutine SBSYN

    Checks the syntax of a SUBSTRATE card.

36. Subroutine SLSYN

    Checks the SAVE and LOAD card syntax.

37. Subroutine STOPS

    Processes the STOP statement.

38. Subroutine STOSY

    Checks the STOP statement syntax.

39. Subroutine STPRC

    Calls the correct process model according to the value of NSTEP.

40. Subroutine STPST

    Processes the STEP statements. Extracts the parameters and determines the step type, initializing the correct model parameters and returns to the MAIN controller for the actual STEP processing.

41. Subroutine STPSV

    Saves the STEP statement in the optimization step file.

42. Subroutine STPTP

    Identifies the STEP card type.

43. Subroutine STSYN

    Checks the syntax of the STEP statements.

44. Subroutine SUBS

    Processes any SUBSTRATE input card. The card can specify a crystalline orientation and/or a uniform impurity concentration of a particular element.

45. Subroutine SUERR

Outputs error messages if syntax errors were detected after the syntax checking segment.

46. Subroutine SUINP

Inputs segments for SUPREM and processes all input information.

47. Subroutine SUSYN

Reads and outputs without checking the input deck. Then it reads the input deck again but this time checks the syntax and legality of each line.

48. Subroutine SVLDF

Saves or restores the impurity distributions.

49. Subroutine TGSIN

Processes the TITLE, initial SUBSTRATE, and GRID cards. It also initializes the common areas to their default values.

3.0 USER'S GUIDE TO INPUT DATA

The input data format for MEMBRE is very much the same as that of SUPREM II. Our philosophy is to allow present SUPREM users to switch over to the MEMBRE program with very little effort. Because of time limitations, some of the features in SUPREM have not been implemented into MEMBRE, but it is our goal to improve and to add new features in the future. In this section, we shall not discuss the input coding of SUPREM, but only identify what has been modified and/or added as MEMBRE inputs. Users unfamiliar with SUPREM inputs are referred to Technical Report No. 5019-2, "SUPREM II – A Program for IC Process Modeling and Simulation," by D.A. Antoniadis, S.E. Hansen, and R.W. Dutton of Stanford University, dated June 1978, under Army Research Office Contract DAAG-29-77-C-006. Examples of utilization of this data format for MEMBRE input are provided in Section 4.

INITIALIZATION CARDS

TITLE CARD – this is not changed; same as SUPREM.

SUBSTRATE CARD – this is not changed; same as SUPREM.

END CARD – this is not changed; same as SUPREM.

GRID CARD – Entries DELY and YLMX are added.

```
GRID (DYSI = <N>) [,DPTH = <N>] (,YMAX = <N>) (,DELY = <N>)
+ (,YLMX = <N>)
```

DYSI(DELX): Space between grid points in ξ direction.

DPTH: Not used in MEMBRE; entry is ignored.

YMAX (L_o): Maximum thickness in microns in ξ direction.

DELY: Space between grid points in Y direction.

YLMX: Maximum length in microns in Y direction.

Note: MEMBRE only allows one GRID CARD; subsequent grid cards will be ignored. At present, it can handle a problem size of approximately $NX * NY \leq 5001$, where $NX = L_0/DELX + 1$ and $NY = YLMX/DELY + 1$. The dimension statements in MEMBRE assume $\max(NX,NY) \leq 101$. It is *not* required that DELX and DELY be the same. Unlike SUPREM, this program cannot allow for variable grids.

INPUT/OUTPUT CARDS

PRINT CARDS

PLOT CARDS

SAVE CARD

LOAD CARD

Note: The above options have not been changed and are temporarily disabled. MEMBRE is presently printing out results at the end of every processing step and saves the values for plotting. The plotting is done on the CDC Cyber 176 system because a 3-D plotting software package is not available on the IBM system at Rockwell.

PROCESS/MODEL CARDS

STEP CARD (Ion Implantation) – Entries YWIN and YDEV are added.

```
STEP TYPE = IMPL, ELEM = <E>, DOSE = <N>, (AKEV = <N>) or
+                (RANG = <N>, STDV = <N>, YDEV = <N>)
+                ,YWIN = <N> [,MODL = <M>]
```

YDEV: Value of lateral standard deviation (σ_L). The values of this parameter for different implant energies (keV) have been incorporated into the SUPREM lookup table for boron, arsenic, phosphorus, and antimony in silicon. This YDEV (σ_L) value, as well as those for STDV(σ_p) and RANG(R_p) which are also obtained from the SUPREM lookup table, can be replaced or set to other values by the user through use of the second option. These values are used in equation (1).

YWIN: Value of one-half the size of window opening; also, it denotes the position of the impenetrable mask edge. YWIN = a in equation (1).

STEP CARD (Etch) – this is not changed; same as SUPREM.

STEP CARD (Low Temperature Oxide Deposition) – this is not changed; same as SUPREM.

STEP CARD (Oxidation and Drive-in) – Entries YPEN and RATO are added.

```
STEP TYPE = OXID, TIME = <N> [,TEMP = <N>] [,TRTE = <N>]
                 [,MODL = <M>] [,YPEN = <N>, RATO = <N>]
```

RATO: Value of the constant, K_0, in equations (11) and (17). Also, it represents the ratio of lateral to vertical oxide growths.

YPEN: Value of this empirical parameter denotes the amount of penetration the bird's beak makes under the Si_3N_4 mask. The symmetry position, $\eta_0 = y_0$, of the complementary error function and the mask edge position, YWIN $= y_a$, are related to YPEN as follows: $y_0 = y_a + \text{YPEN}$. This expression states: (a) if YPEN $= 0$ then $y_0 = y_a$ and the penetration of the bird's beak under the mask is 50%; (b) if YPEN > 0, then $y_0 > y_a$, and the penetration of the bird's beak under the mask is > 50%; and (c) if YPEN < 0, then $y_0 < y_a$, and the penetration of the bird's beak under the mask is < 50%. See equations (11) and (17) to learn how $\eta_0 = y_0$ is used in the code.

Note: If the entries RATO and YPEN are absent from the STEP CARD, then the oxide growth model for the uniform moving boundary case is assumed.

STEP CARD (Epitaxial Growth) — Not implemented in MEMBRE. The program will print out a message saying that it cannot model epitaxial growth, and the program stops completely.

ELEMENT AND OXIDATION MODELS

Beta and alpha are coefficients whose values are determined by the parameters of the oxidation model card, while the parameters of the elemental model card determine the values of the coefficients D_I and K_I. These coefficients are converted from SUPREM for use in MEMBRE as follows:

PARABOLIC GROWTH RATE (BETA)

SUPREM: B_{OXI} = PRTE * EXP (-PREA/KT) * PRES

MEMBRE: BETA = B_{OXI} * 60 [CONVERTED TO UNIT/HR FROM UNIT/MIN.]

LINEAR GROWTH RATE (ALPHA)

SUPREM: A_{OXI} = LRTE * EXP (-LREA/KT) * PRES

MEMBRE: ALPHA = B_{OXI}/A_{OXI}

SEGREGATION COEFFICIENT (KI) – denoted by k in equations (3) and (9)

```
     SUPREM:  AM = SEGO * EXP (-SEGE/KT)

     MEMBRE:  KI = 1./AM
```

DIFFUSION COEFFICIENT (DI) – denoted by D_0 in equations (4)

```
     SUPREM:  DSN = DSXN * EXP (-ESXN/KT)
              DSW = DSXW * EXP (-ESXW/KT)
              DSD = DSXD * EXP (-ESXD/KT)

     MEMBRE:  DI = DSN * 60. UNIT/HRS  For nitrogen
              DI = (DSN + DSD) * 60.   For boron in dry oxygen
              DI = (DSN + DSW) * 60.   For boron in wet oxygen
              DI = DSD * 60.           For other elements in dry oxygen
              DI = DSW * 60.           For other elements in wet oxygen
```

4.0 EXAMPLES

In this section, three cases of input code simulated by MEMBRE are presented.

CASE 1. REDISTRIBUTION OF BORON FIELD IMPLANT

This problem involves 15 processing steps including the implant step. The complete code for this problem is shown in Figure 2.

The first card of the input deck is a title card. As shown on line 10, after the key word "TITL", a string of characters describing the problem is entered.

The next card is the GRID card shown on line 20. Both DELX and DELY have a value of 0.05. Recall that the entry name for DELX is DYSI. The silicon thickness in the ξ-direction is 2 μm, using the entry name YMAX. YLMX is the entry for maximum length of silicon in the η-direction (3 μm). As was mentioned in Section 3, the parameter DPTH is not used by the MEMBRE program and has been set equal to the value 0.05. The input data for this case gives a spatial grid size of 41×61 points ($2/.05 + 1 = 41$ and $3/.05 + 1 = 61$).

N_b (boron background) is simulated by the SUBS card of line 30, using standard SUPREM format.

For the initial oxide thickness of 0.005 μm, a STEP card with "TYPE = DEPO" is used. Line 50 of the input code shows the complete entry. Lines 60 and 65 are unnecessary because print and plot files are automatic in MEMBRE.

A 150 keV boron implant of dose $2.5 \times 10^{12}\,\text{cm}^{-2}$ is simulated by the STEP card shown on line 80. "a" which is 1 μm is coded as YWIN = 1. The entry YDEV (lateral standard deviation, σ_L) is normally not necessary when AKEV is entered because the RANGE (R_p), standard deviation (σ_p), and σ_L are all provided by the lookup table which is incorporated in the

```
00010 TITL  BORON FIELD IMPLANT, E-NMOS
00020 GRID  DYSI=0.05,DPTH=0.05,YMAX=2,DELY=0.05,YLMX=3
00030 SUBS  ORNT=100,ELEM=+,CONC=5E14
00040 COMM  STARTING OXIDE THICKNESS OF 0.005 UM
00050 STEP  TYPE=DEPO,TIME=1,GRTE=0.005
00060 PLOT  TOTL=Y
00065 PRINT TOTL=Y,HEAD=Y
00070 COMM  150KEV BORON IMPLANT
00080 STEP  TYPE=IMPL,ELEM=B,DOSE=2.5E12,AKEV=150,YDEV=0.148,YWIN=1.
00090 COMM  .....STOP PLOTTING, STARTING PRINTING....
00100 PLOT  TOTL=N
00120 COMM  FIELD OXIDATION
00130 STEP  TYPE=OXID,TEMP=1000,TIME=20,MODL=NIT0
00140 STEP  TYPE=OXID,TEMP=1000,TIME=160,MODL=WET0,RATO=0.3333,YPEN=0.075
00150 STEP  TYPE=ETCH,TIME=1,ERTE=0.05
00155 COMM  SACRIFICIAL OXIDE
00160 STEP  TYPE=OXID,TEMP=875,TIME=10,MODL=DRY0
00170 STEP  TYPE=OXID,TEMP=875,TIME=45,MODL=WET0
00180 STEP  TYPE=OXID,TEMP=875,TIME=10,MODL=NIT0
00190 STEP  TYPE=ETCH,TIME=1,ERTE=0.07
00195 COMM  GATE OXIDATION
00200 STEP  TYPE=OXID,TEMP=1000,TIME=65,MODL=DRY0
00210 STEP  TYPE=OXID,TEMP=1000,TIME=10,MODL=NIT0
00215 COMM  PHOS DEP + POLY OXIDATION
00220 STEP  TYPE=OXID,TEMP=950,TIME=60,MODL=NIT0
00225 COMM  ARGON ANNEAL
00230 STEP  TYPE=OXID,TEMP=1000,TIME=40,MODL=NIT0
00235 COMM  BARRIER OXIDATION
00240 STEP  TYPE=OXID,TEMP=950,TIME=140,MODL=DRY0
00245 COMM  SILOX DENSIFICATION + REFLOW
00250 STEP  TYPE=OXID,TEMP=950,TIME=50,MODL=NIT0
00260 END
```

Figure 2. Input Data for Case 1 — Boron Field Implant

program. However, the code on line 80 allows the user to supersede the given value of σ_L from the table with $\sigma_L = .148$.

Cycle 1 of this problem is simulated by the STEP card shown on line 130. The following consecutive cycles of this problem are given by the consecutive STEP cards starting from line 140 of the input code.

Attention should be paid to the input code of line 140. This STEP card simulates the nonuniformly moving boundary model by entering the parameters RATO and YPEN. The formulation for the model has been mentioned previously in the description of the subroutine UFCT. All other steps involve the use of the uniformly moving boundary model which is also described in the UFCT subroutine. It should be noted that the parameters RATO and YPEN are absent from the STEP card when the oxidation model for the uniformly moving boundary case is being used.

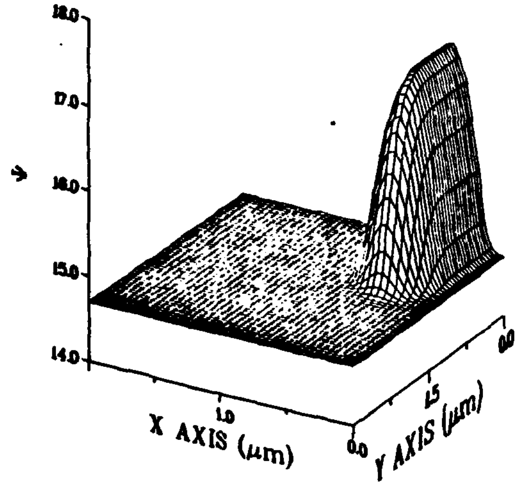
There are two etch steps for this example. They are simulated by STEP cards using TYPE = ETCH shown on lines 150 and 190.

As stated earlier, MEMBRE writes a plot file (on TAPE 20), but no plotting routines are supplied with the code because of the non-standardization of software packages at various computing centers. At Rockwell International, we employ the DISSPLA package (Display Integrated Software System and Plotting Language), which is available for many computers but has only been leased by Rockwell for the CDC machine. The tape is transferred from the IBM to the CDC machine with little difficulty. Some of these plots are illustrated below for Case 1.

Figure 3 shows the two-dimensional initial implant distribution and corresponding equi-density contours as calculated analytically using equation (1). The redistributed profile at the end of the fourth process step, which models the growth of the field oxide, is illustrated in Figure 4. The equi-density contours (Figure 4b) reveal a field oxide thickness which is uniform in the region of the window opening and

2D DISTRIBUTION (time = 0.00 hrs)
$\Psi = \mathrm{LOG}_{10}|N+N_b|$
N — IMPLANT(boron) DISTRIBUTION
N_b — UNIFORM(boron) BACKGROUND

150 keV boron implant (dose $2.5 \times 10^{12} cm^{-2}$)

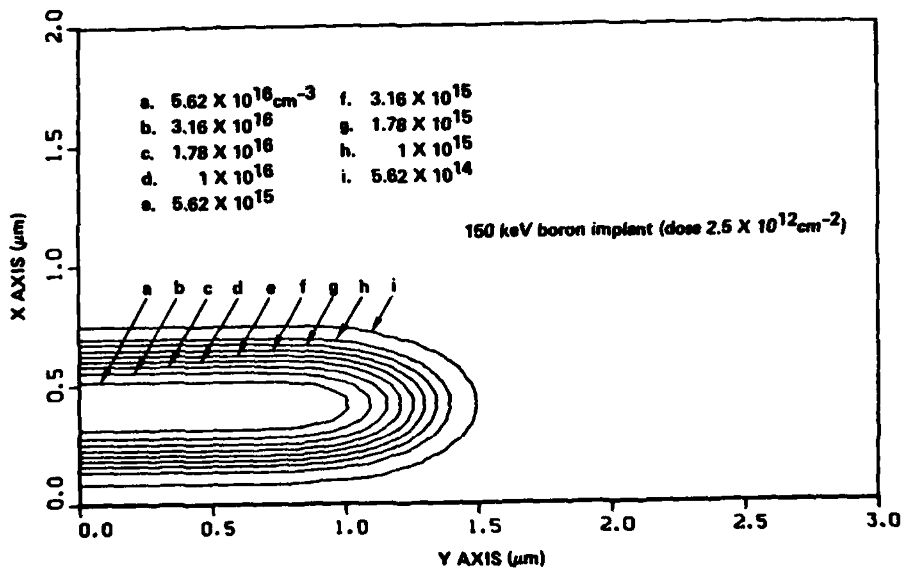


(a)

EQUI-DENSITY CONTOURS (0.00 hrs)

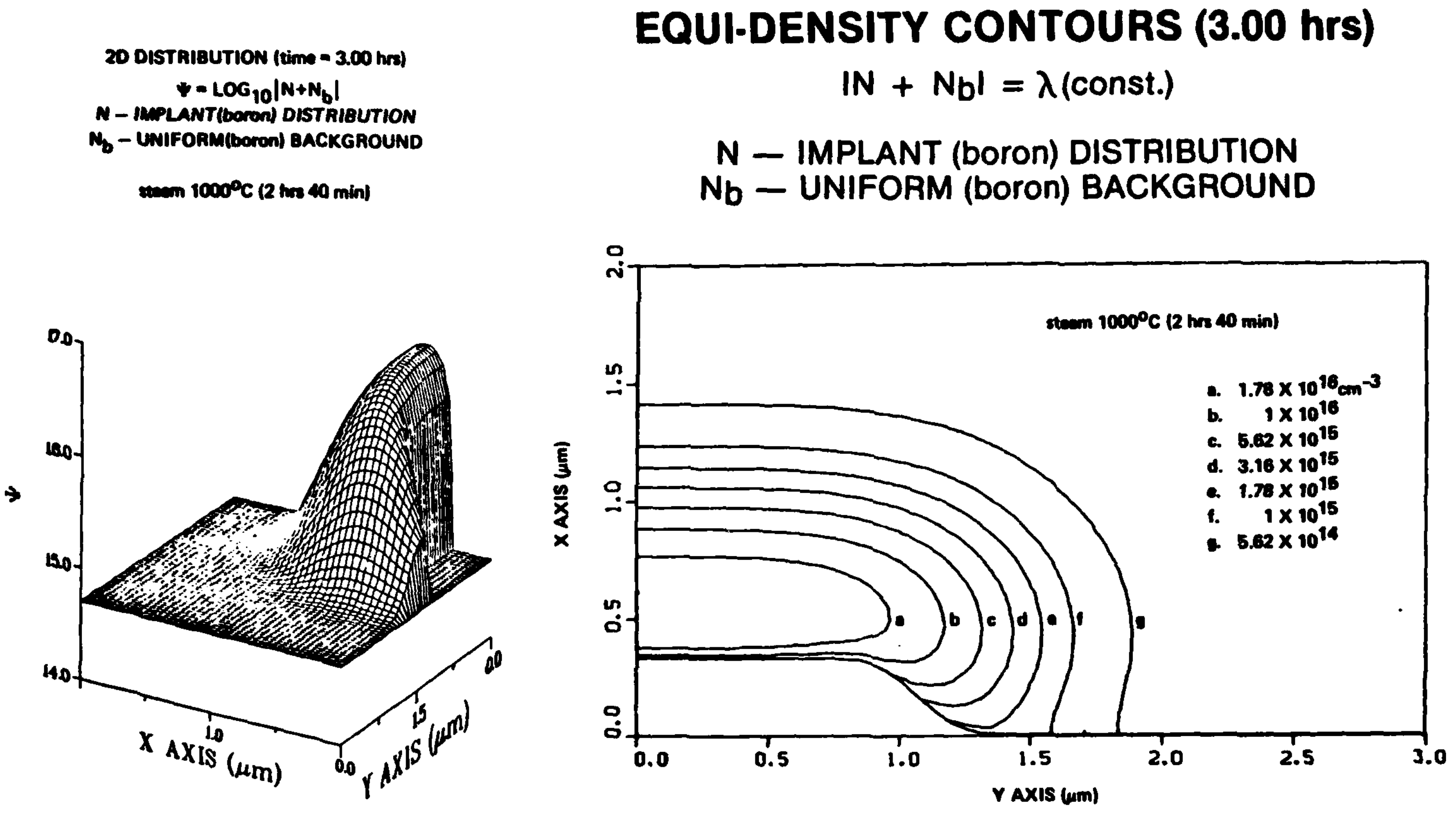
$|N + N_b| = \lambda$(const.)

N — IMPLANT (boron) DISTRIBUTION
N_b — UNIFORM (boron) BACKGROUND



(b)

Figure 3. Case 1: Boron Field Implant at t = 0



(a) (b)

Figure 4. Case 1: Redistribution of Boron Field Implant at t = 3

decreasing in value in the neighborhood of the window's edge to the mask, with significant penetration of this transition region (bird's beak) under the Si_3N_4 mask. Also, in the window and transition regions where the oxide-silicon interface is moving, there is considerable segregation of boron from the silicon to the oxide, as is evident from Figure 4. Figure 5 illustrates the profile at the end of the seventh process step, which involves the interaction of an oxidizing ambient of steam at 875°C for 45 minutes after removal of the Si_3N_4 mask. Again, boron is being segregated from the silicon to the oxide at the moving interface. Figure 6 shows the redistributed profile at the end of the final annealing step, which would be used for the determination of electrical properties (capacitance, breakdown voltage, etc.) in the neighborhood of the bird's beak. CPU time for this case was 6.87 minutes on the IBM 3033 and would be about 2.75 minutes on the CDC 176. Much smaller computer times would result if a coarser spatial grid was selected. Here our interest was in accuracy and not fast execution time.

CASE 2. REDISTRIBUTION OF BORON CHANNEL IMPLANTS

The MEMBRE input code for this redistribution problem is shown in Figure 7. In this example, a double implant through two STEP cards is considered. The implant steps are shown on lines 80 and 100. By observing the GRID card, one notices that in this example the spatial grid size is 61×41 points ($NX = 1.5/.025 + 1 = 61$ and $NY = 2.0/.05 + 1 = 41$). This example has a total of nine steps including the implant steps. CPU time for this case was 1.98 minutes on the IBM 3033 and would have been about 0.79 minute on the CDC 176. Some characteristic plots for this case can be found below.

Figure 8 contains the two dimensional distribution and corresponding equi-density contours for the superposition of the 50 keV (shallow) and

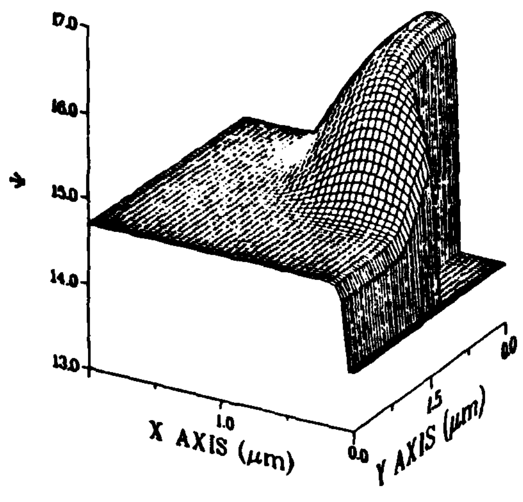
2D DISTRIBUTION (time = 3.92 hrs)
$\Psi = \mathrm{LOG}_{10}|N+N_b|$
N – IMPLANT(boron) DISTRIBUTION
N_b – UNIFORM(boron) BACKGROUND

steam 875°C (45 min)

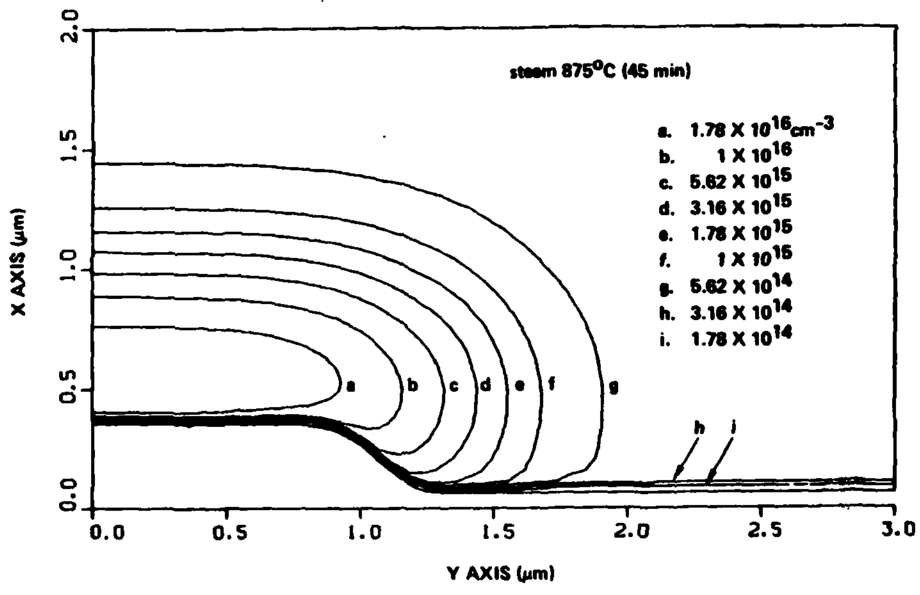
EQUI-DENSITY CONTOURS (3.92 hrs)

$|N + N_b| = \lambda$(const.)

N — IMPLANT (boron) DISTRIBUTION
N_b — UNIFORM (boron) BACKGROUND



(a)

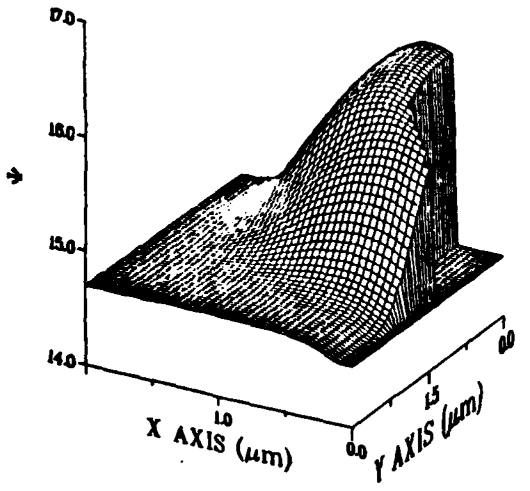


(b)

Figure 5. Case 1: Redistribution of Boron Field Implant at t = 3.92

2D DISTRIBUTION (time = 10.17 hrs)
$\Psi = \mathrm{LOG}_{10} |N + N_b|$
N – IMPLANT(boron) DISTRIBUTION
N_b – UNIFORM(boron) BACKGROUND

argon 950°C (50 min)

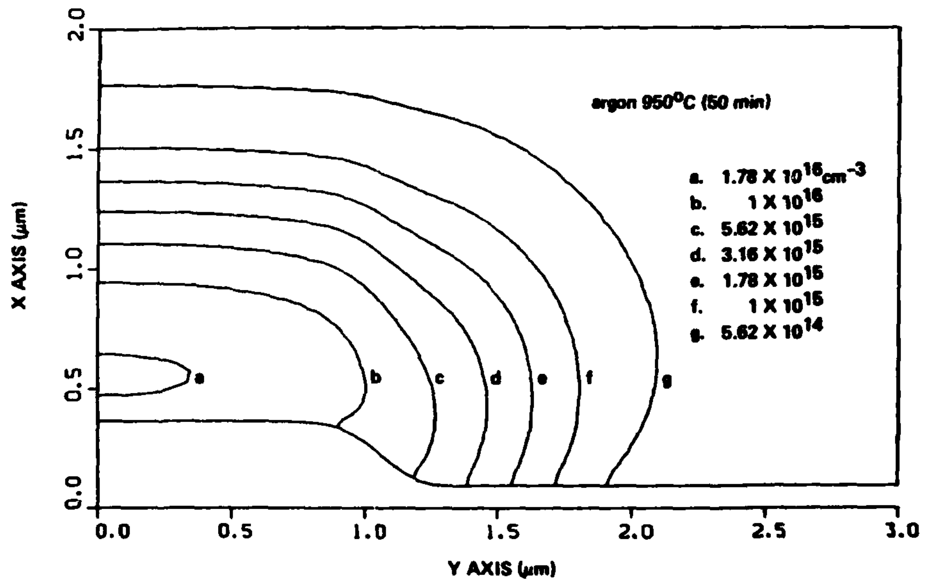


(a)

EQUI-DENSITY CONTOURS (10.17 hrs)

$|N + N_b| = \lambda(\text{const.})$

N — IMPLANT (boron) DISTRIBUTION
N_b — UNIFORM (boron) BACKGROUND

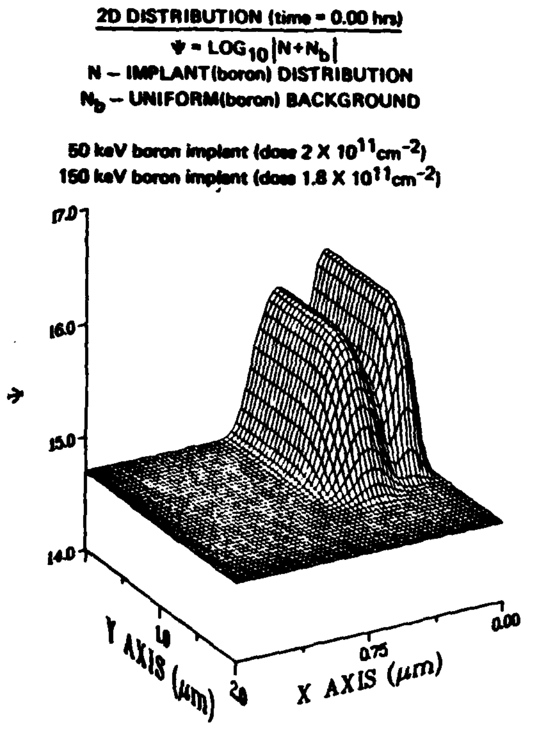


(b)

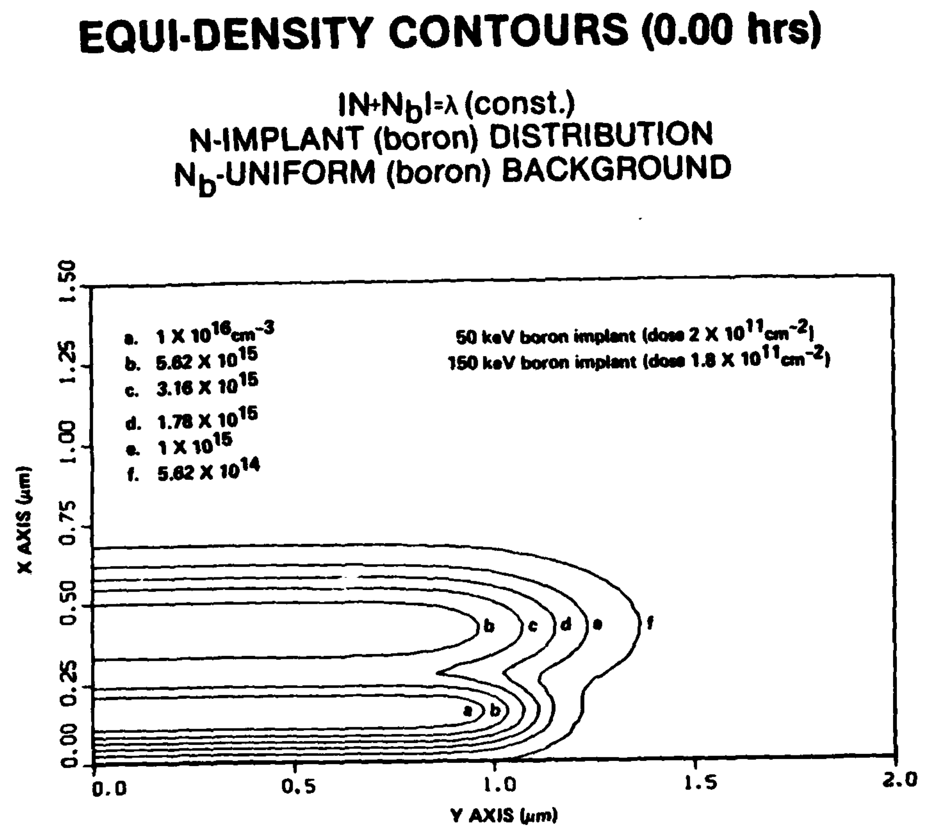
Figure 6. Case 1: Redistribution of Boron Field Implant at t = 10.17

```
00010 TITL BORON CHANNEL IMPLANTS, E-NMOS
00020 GRID DYSI=0.025,DPTH=0.025,YMAX=1.5,DELY=0.05,YLMX=2.0
00030 SUBS ORNT=100,ELEM=+,CONC=5E14
00040 STEP TYPE=DEPO,TIME=1,GRTE=0.005
00050 PLOT TOTL=Y,CMIN=14,NDEC=4,WIND=2
00060 PRINT HEAD=Y,TOTL=Y
00070 COMM 150KEV BORON IMPLANT
00080 STEP TYPE=IMPL,ELEM=B,DOSE=1.8E11,AKEV=150,YWIN=1,YDEV=0.148
00090 COMM 50 KEV BORON IMPLANT
00100 STEP TYPE=IMPL,ELEM=B,DOSE=2.0E11,AKEV=50,YWIN=1,YDEV=0.0793
00110 COMM GATE OXIDATION
00130 STEP TYPE=OXID,TEMP=1000,TIME=10,MODL=WETO
00140 STEP TYPE=OXID,TEMP=900,TIME=30,MODL=NITO
00145 COMM PHOS DEP + POLY OXIDATION (CAPPED)
00150 STEP TYPE=OXID,TEMP=950,TIME=60,MODL=NITO
00155 COMM ARGON ANNEAL
00160 STEP TYPE=OXID,TEMP=1000,TIME=40,MODL=NITO
00165 COMM BARRIER OXIDATION (CAPPED)
00170 STEP TYPE=OXID,TEMP=950,TIME=140,MODL=NITO
00175 COMM SILOX DENSIFICATION + REFLOW
00177 STEP TYPE=OXID,TEMP=950,TIME=50,MODL=NITO
00180 END
```

Figure 7. Inout Data for Case 2 — Boron Channel Implants



(a)



(b)

Figure 8. Case 2: Boron Channel Implants at t = 0

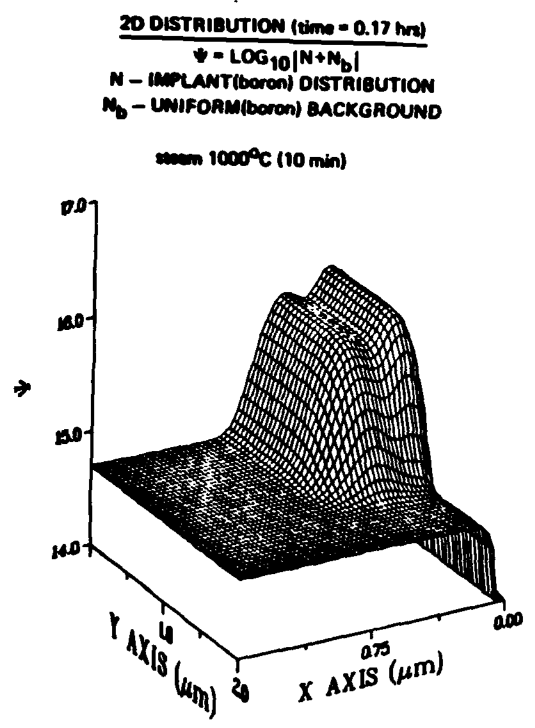
the 150 keV (deep) boron implant, which are the channel implants for an enhancement mode device. Although lateral penetration under the mask edge is evident for both implants, the deeper one has much more due to its greater scattering potential. At the end of the fourth process step (Figure 9), one observes segregation at the uniformly moving silicon-oxide interface because the entire surface is exposed to the oxidizing ambient. The final profile (Figure 10) plays an important role in determining the device I-V characteristics, threshold voltage, punchthrough voltage, etc.

CASE 3. REDISTRIBUTION OF ARSENIC SOURCE/DRAIN IMPLANT

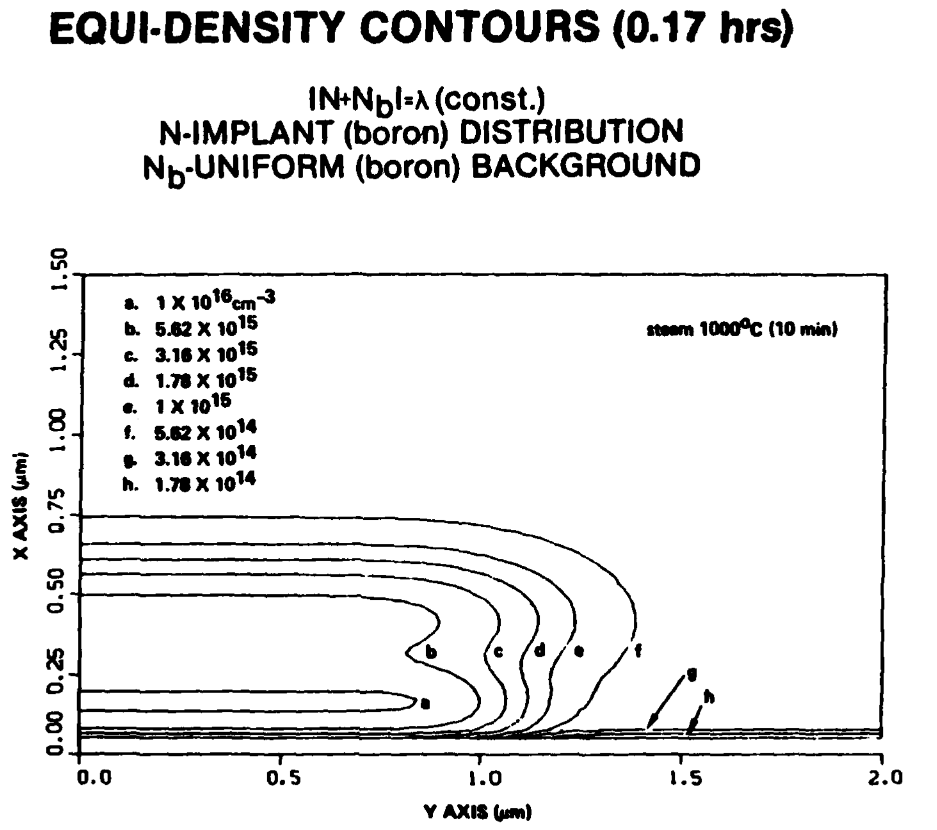
The thermal redistribution of an arsenic implant is demonstrated in this example, involving a total of four steps. The input code is shown in Figure 11. In the implant STEP card, the values for RANG (R_p), STDV (σ_p), and YDEV (σ_L) are initialized in lines 110 and 120. By inputting the parameters in this manner, the default values of the lookup table are bypassed. Three oxidation steps follow for 40, 140, and 50 minutes.

In general, arsenic problems require fine grids and consume more computer time. In this example, a 51×61 spatial grid was employed, and 10.37 minutes of CPU time on the IBM 3033 were required (about 4.15 minutes on a CDC 176).

Initial conditions using equation (1) are plotted in Figure 12 for this 40 keV arsenic implant. Note that there is only a slight penetration under the mask edge. After 40 minutes (second process step), Figure 13 reveals an explosive type diffusion in both vertical and lateral directions which gives rise to a flat plateau profile in the high concentration region and a rapid drop into the boron background. Such diffusion is characteristic of the highly nonlinear nature of this boundary value problem and is responsible for the large CPU time.



(a)



(b)

Figure 9. Case 2: Redistribution of Boron Channel Implants at t = .17

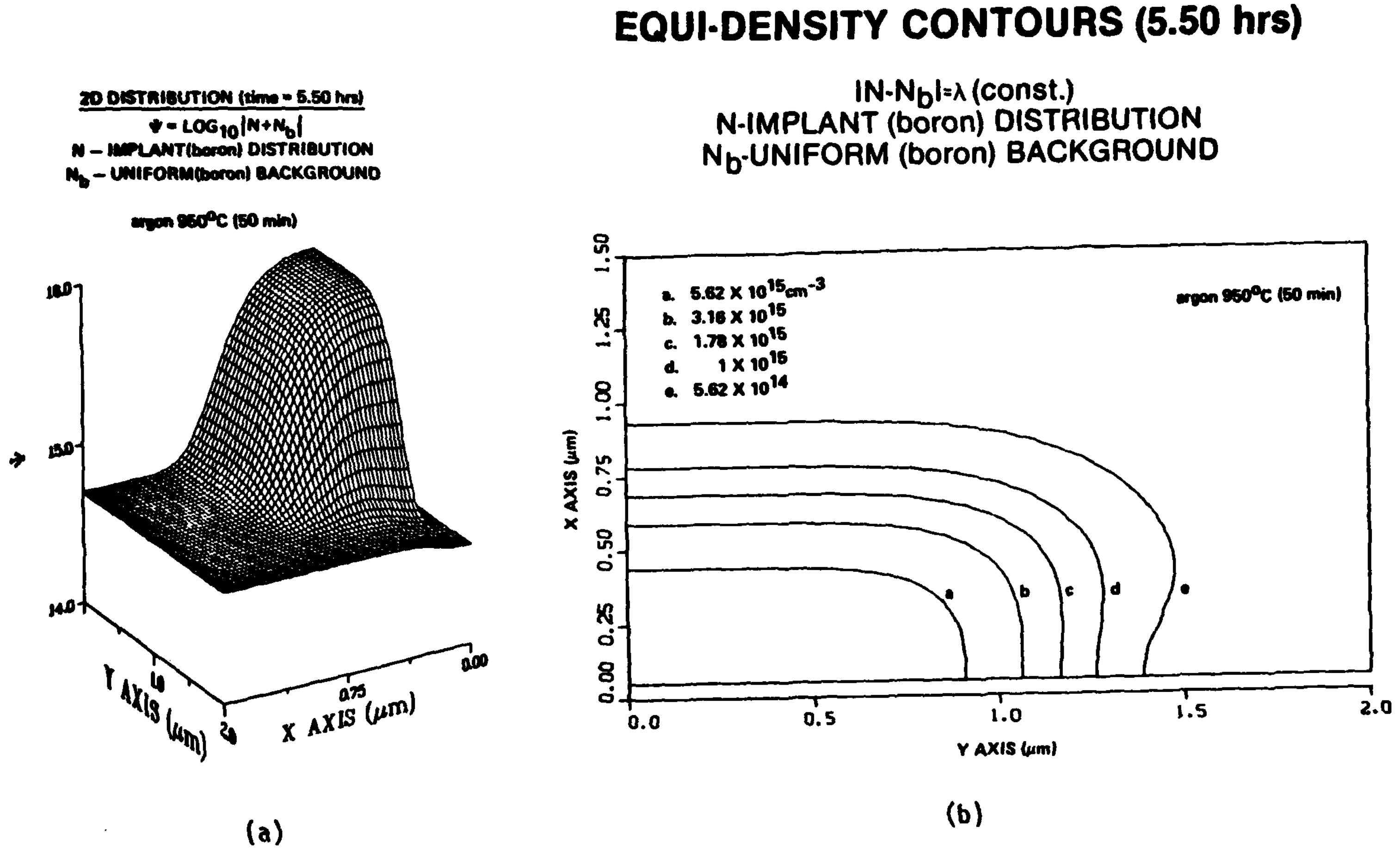
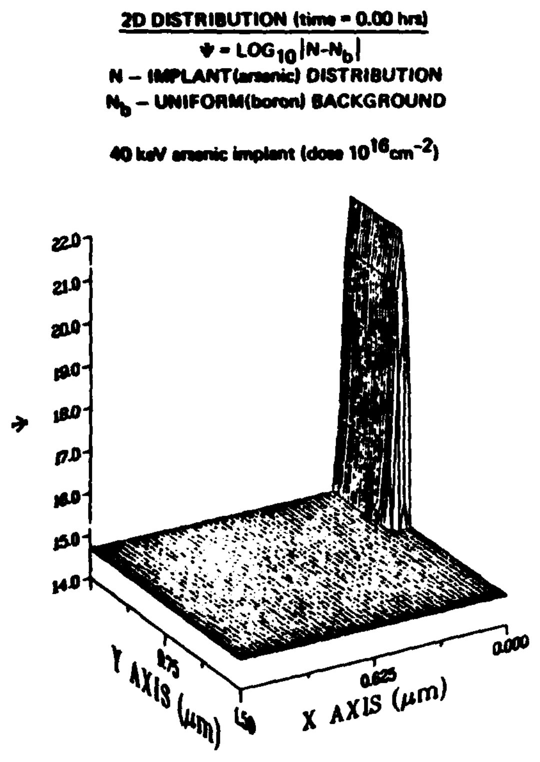


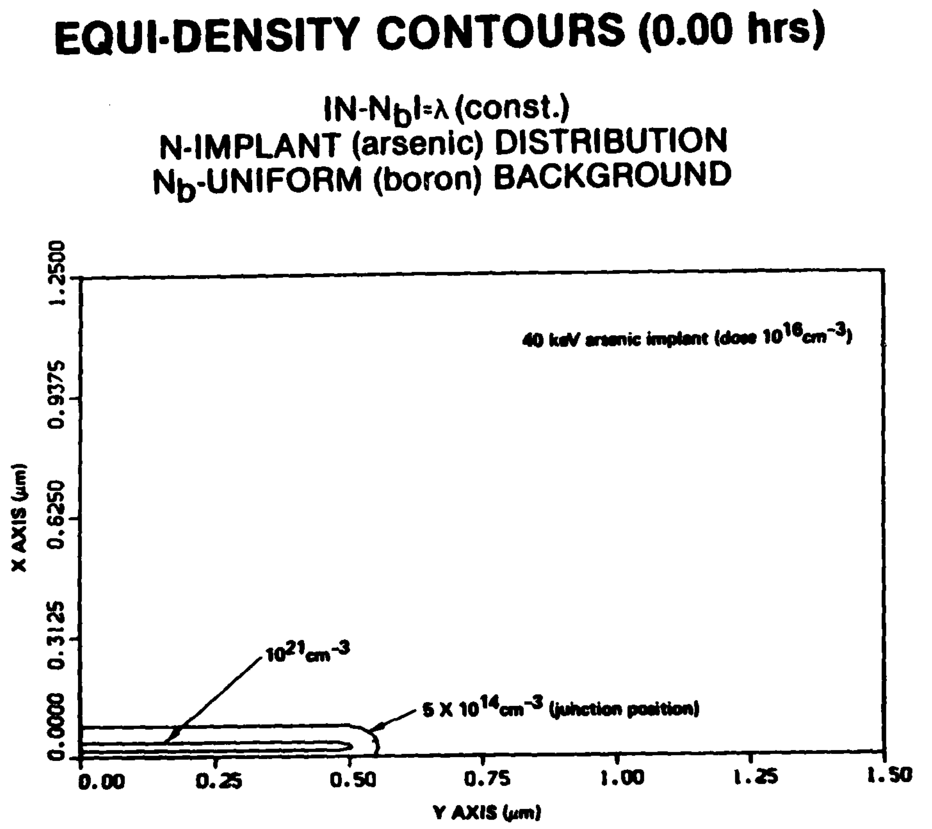
Figure 13. Case 2: Redistribution of Boron Channel Implants at t = 5.5

```
00050 TITL ARSENIC SOURCE/DRAIN IMPLANT, E-NMOS
00060 GRID DYSI=0.025,DPTH=0.025,YMAX=1.25,DELY=0.025,YLMX=1.5
00070 SUBS ORNT=100,ELEM=+,CONC=5E14
00080 PLOT TOTL=Y,CMIN=14,NDEC=4,WIND=2
00090 PRINT TOTL=Y,HEAD=Y
00100 COMM 40KEV ARSENIC IMPLANT
00110 STEP TYPE=IMPL,ELEM=AS,DOSE=1E16,RANG=0.0265,STDV=0.0099,
00120 +                          YWIN=0.5,YDEV=0.0103
00130 COMM ARGON ANNEAL
00140 STEP TYPE=OXID,TEMP=1000,TIME=40,MODL=NIT0
00150 COMM BARRIER OXIDATION
00160 STEP TYPE=OXID,TEMP=950,TIME=140,MODL=DRY0
00170 COMM SILOX DENSIFICATION + REFLOW
00180 STEP TYPE=OXID,TEMP=950,TIME=50,MODL=NIT0
00190 END
```

Figure 11. Input Data for Case 3 – Arsenic Source/Drain Implant

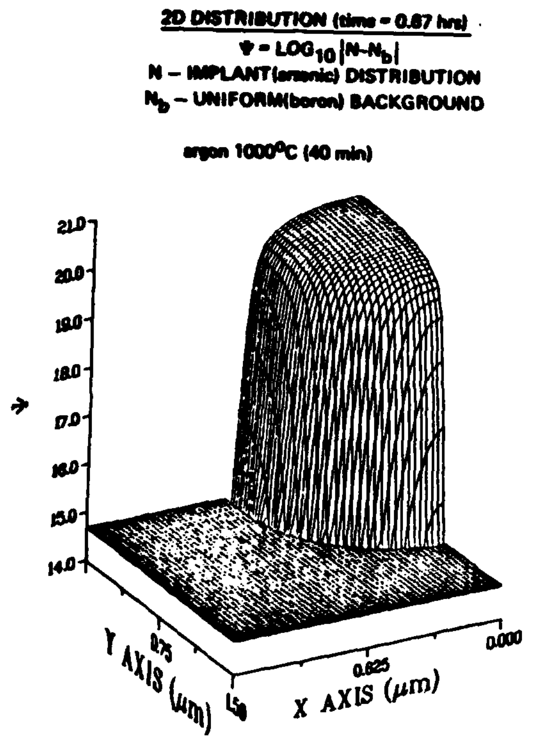


(a)



(b)

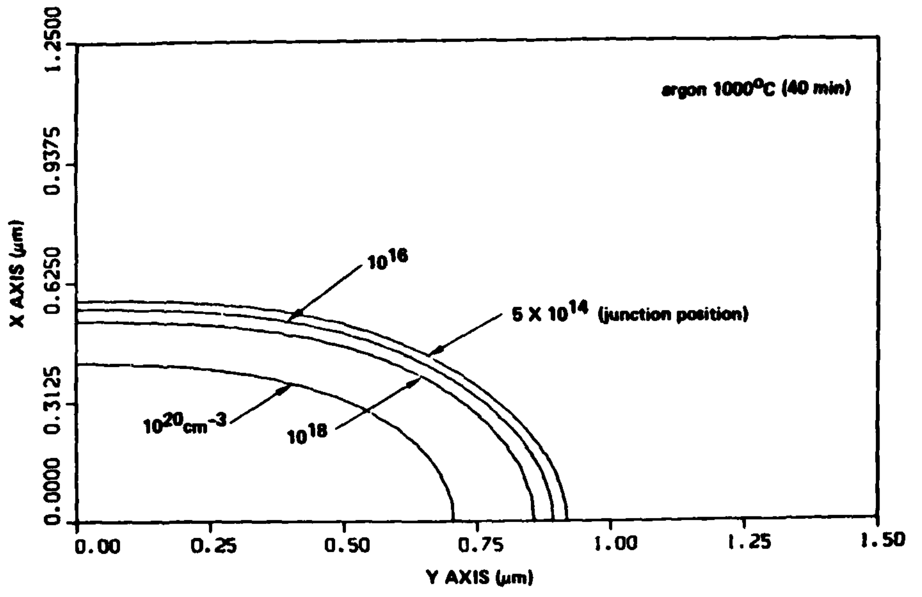
Figure 12. Case 3: Arsenic Source/Drain Implant at t = 0



(a)

EQUI-DENSITY CONTOURS (0.67 hrs)

$|N-N_b|=\lambda$ (const.)
N-IMPLANT (arsenic) DISTRIBUTION
N_b-UNIFORM (boron) BACKGROUND



(b)

Figure 13. Case 3: Redistribution of Arsenic Source/Drain Implant at t = .67

Figure 14 illustrates the final redistributed profile which contains information about junction depth and lateral penetration of the junction into the channel region.

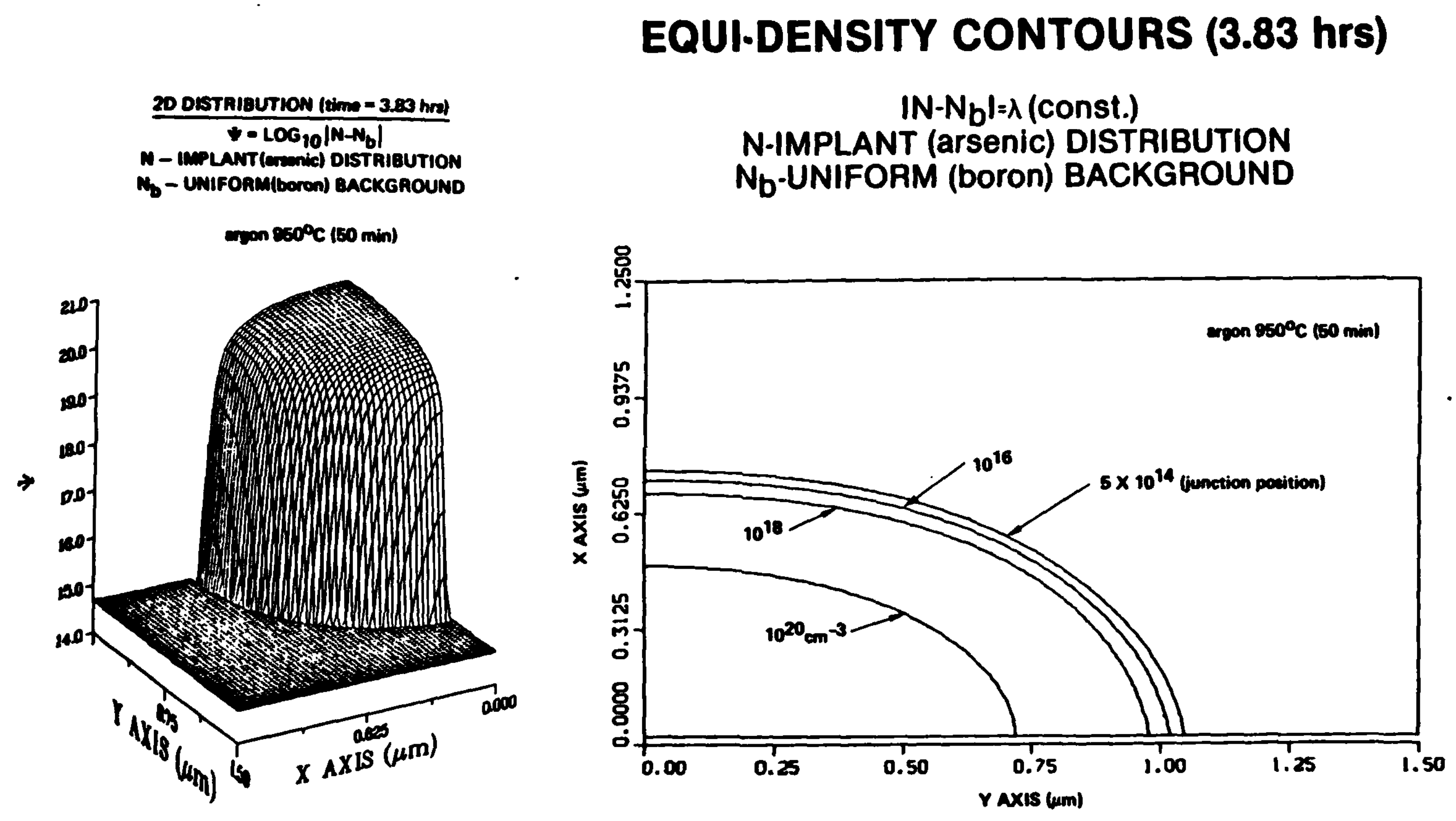
In order to give the reader some further experience on the usage of MEMBRE, we illustrate below some of our earlier one cycle results using the CDC Cyber 176 computer. IBM 3033 or IBM 370 (Mod 168) execution times should be approximately two-and-one-half times longer.

MEMBRE was utilized to predict the redistribution of an 80 keV boron field implant of dose $3 \times 10^{13} cm^{-2}$ during the growth of a field oxide using an oxidizing ambient of steam at 1000°C for 2-1/3 hours. The resultant equi-concentration contours are shown in Figure 15. For the special cases corresponding to the redistribution of source/drain and channel implants, see Figures 16 and 17, respectively. Figure 16 is for the redistribution of a channel profile which is composed of a shallow 40 keV boron implant (dose $2 \times 10^{11} cm^{-2}$) superimposed on a deep 120 keV boron implant (dose $1.5 \times 10^{11} cm^{-2}$), during the growth of a gate oxide by an oxidizing ambient of steam at 1000°C for 1/4 hour. In Figure 17, the redistribution is for a high dose $5 \times 10^{15} cm^{-2}$ (source/drain) 40 keV arsenic implant which is being driven-in by a nonoxidizing ambient of argon at 1000°C for 1 hour.

In Figure 18, we show results for an 80 keV boron implant of $3 \times 10^{13} cm^{-2}$ dose, oxidized in steam at 1000°C for half an hour. This problem was solved on a 31×51 grid in 12.5 seconds on the CDC 176.

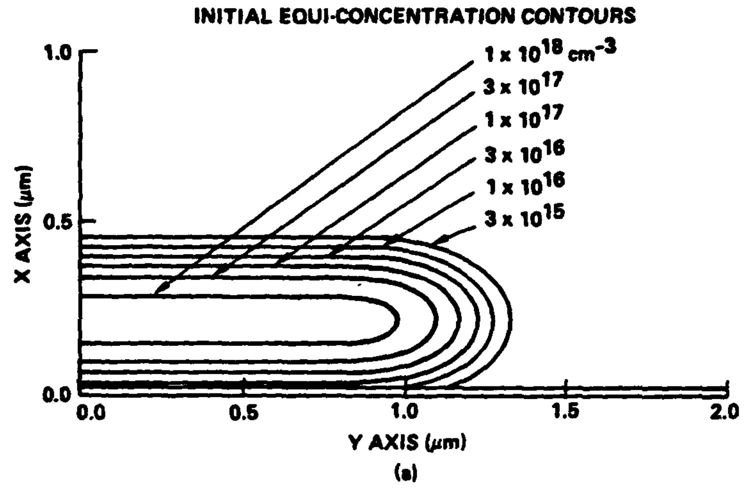
A typical drive-in problem without oxidation is illustrated in Figure 19. Here, a high-dose 40 keV boron implant is redistributed over 2 hours at 1000°C. This problem required 22.5 seconds CPU time for a 21×31 grid.

To illustrate the effect of different grid sizes and diffusivities (DI), we list in Table 1 several runs that were recently made. In



(a) (b)

Figure 14. Case 3: Redistribution of Arsenic Source/Drain Implant at t = 3.83



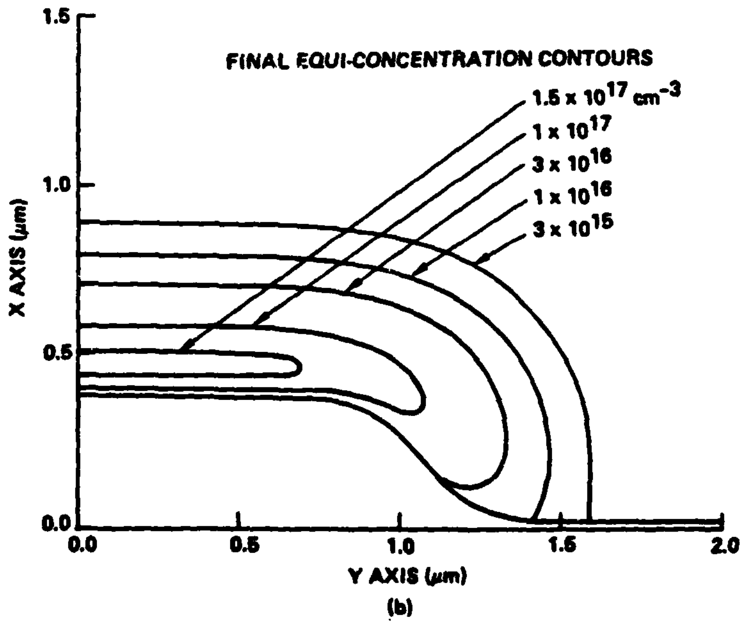


Figure 15. Initial and Final Equi-Boron Concentration Contours for the Redistribution of a Field Implant are Shown in (a) and (b), Respectively

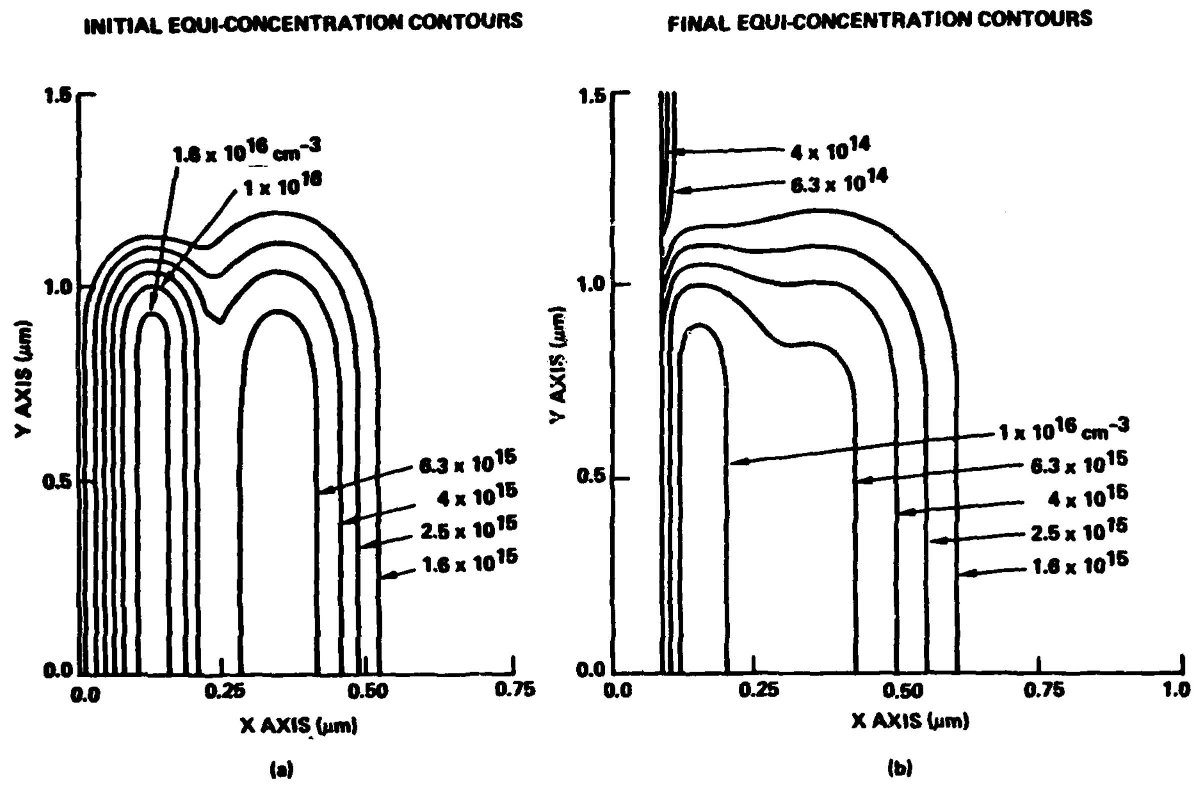


Figure 16. Initial and Final Equi-Boron Concentration Contours for the Redistribution of a Channel Enhancement Mode Profile are Shown in (a) and (b), Respectively

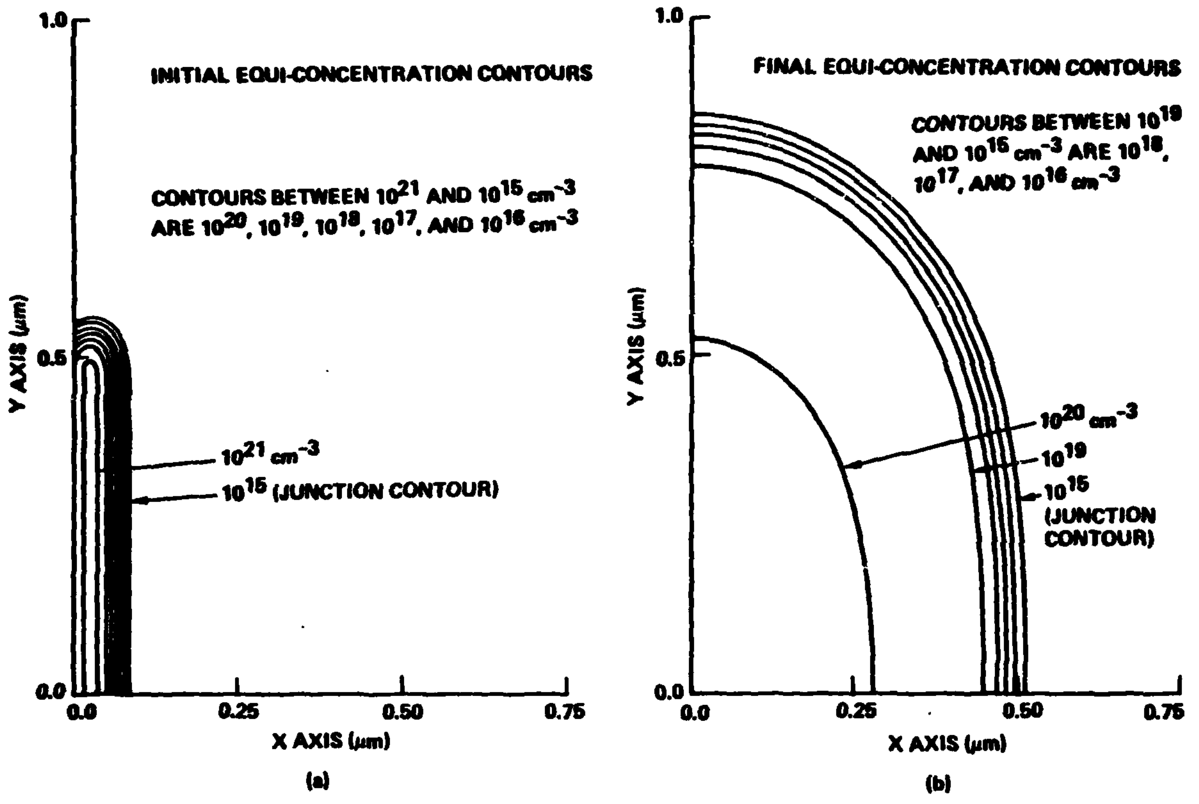


Figure 17. Initial and Final Equi-Arsenic Concentration Contours for the Redistribution of a Source/Drain Implant are Shown in (a) and (b), Respectively

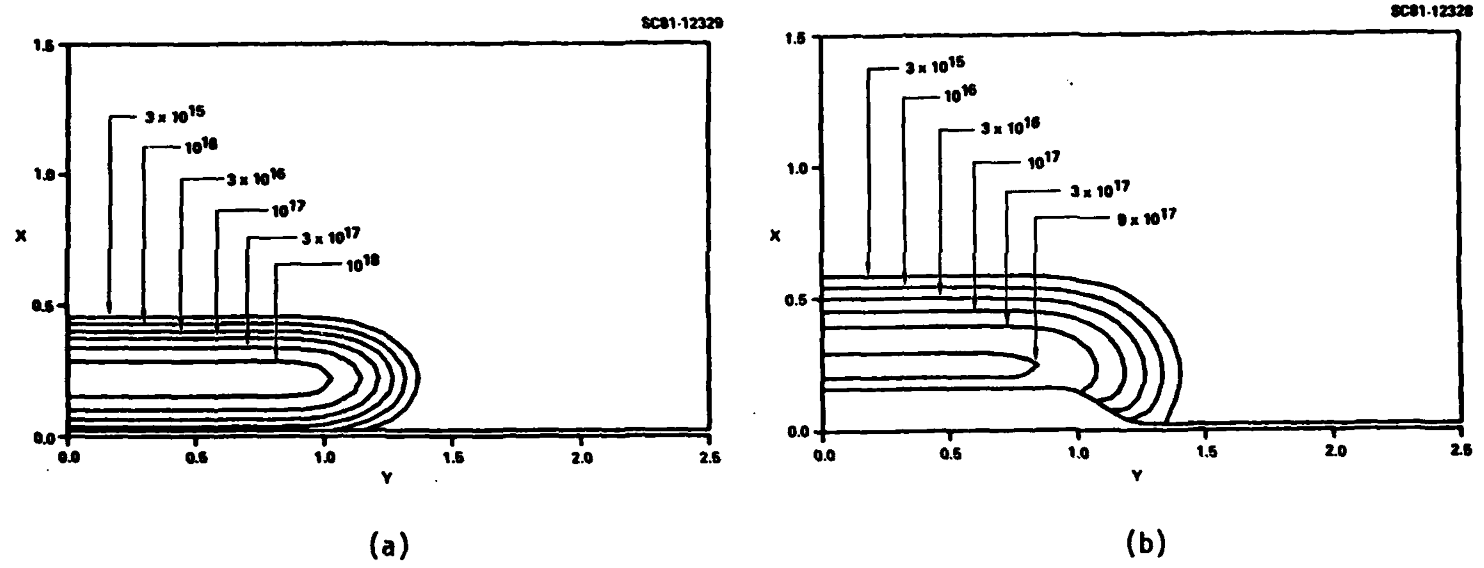


Figure 18. Contours of Constant Dopant Concentration. (a) Initial Conditions after Furukawa, et al. (b) After 30 Minutes in Steam at 1000°C. Note Motion of Oxide Boundary.

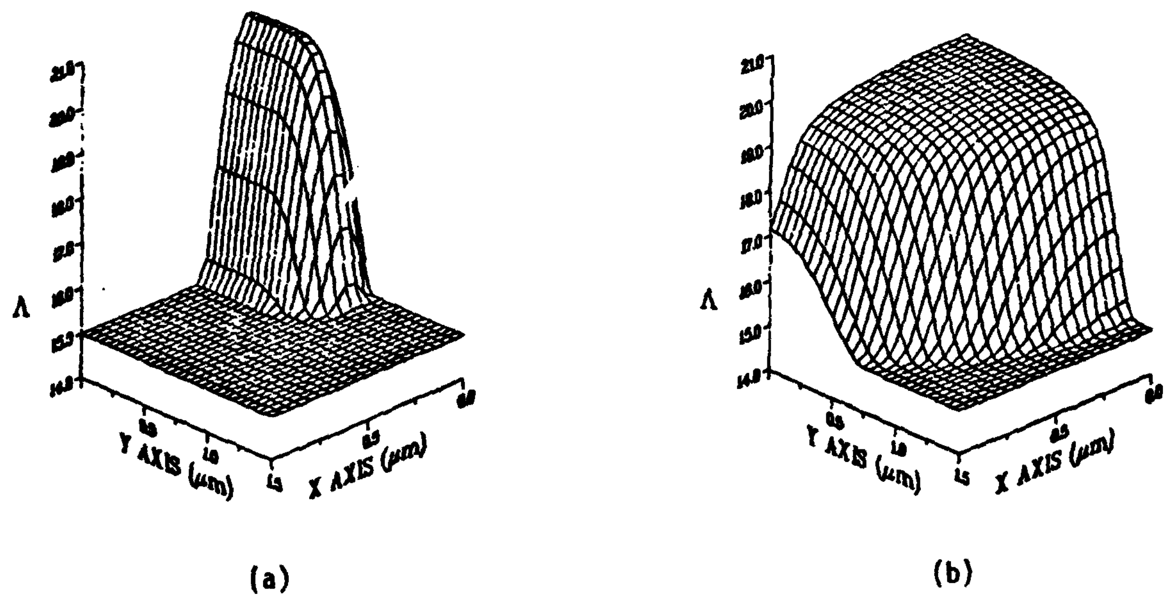


Figure 19. Highly Nonlinear Diffusion of Boron in Silicon. Dose: $5 \times 10^{15}\ \text{cm}^{-2}$. Energy: 40 keV. $\Lambda = \text{Log}_{10} |N + N_b|$. Temperature = 1000°C.
(a) Initial Concentration Profile (Time = 0). (b) Dopant Profile after 1 Hour.

Table 1. Some Case Studies ($\varepsilon = 10^{-4}$) on the CDC 176

Grid	DI	Time (hr)	NSTEPS	CPU (sec)
41 × 81 = 3321	7.6×10^{-3}	0.5	52	32.56
31 × 51 = 1581	7.6×10^{-3}	2.333	65	22.50
31 × 51	7.6×10^{-3}	2	62	18.18
31 × 51	7.6×10^{-3}	0.5	36	13.13
21 × 31 = 651	5.5×10^{-3}	2	239	22.56
61 × 81 = 4941	5.5×10^{-3}	1	163	137.89
41 × 61 = 2501	5.5×10^{-3}	1	164	59.70
21 × 31	5.5×10^{-3}	2	85	8.38
16 × 26 = 416	5.5×10^{-3}	0.25	91	6.43
21 × 41 = 861	1.07×10^{-5}	2	122	14.52
21 × 41	1.07×10^{-5}	2	28	3.59
21 × 41	6.15×10^{-4}	2	86	10.48
21 × 41	6.15×10^{-4}	2	351	42.39
21 × 31	1.85×10^{-4}	2	103	10.17
21 × 31	1.85×10^{-4}	2	29	3.29
21 × 31	6.15×10^{-4}	2	126	11.08

particular, note the "roughly linear" CPU times. That is, for the 61×81 case we would guess that the CPU time should be approximately $(4941/2501)59.70 = 117.9$ seconds (in contrast to the actual value of 137.89). This "roughly linear" behavior is due to the fact that most of the computations occur when solving the linear system (16).

In the same way, we can predict how much CPU time would be required to solve a three-dimensional problem. For example, consider the first case in Table 1. Assume that in the z-direction we introduce 21 points for a grid of $41 \times 81 \times 21$. Then the CPU time for an integration of 1/2 hour using this three-dimensional grid would be approximately $21 \times 32.56 = 683.76$ seconds = 11.4 minutes.

For a coupled system of partial differential equations containing several species, the CPU time would be estimated by multiplying the number of partial differential equations by the CPU time given in Table 1.

Some of the above examples were also presented in Refs. [10] and [12].

5.0 MEMBRE DIMENSION CHANGES

The program dimension statements assume that the solution vector is less than or equal to 5001 and the number of spatial points in the ξ and η directions are each less than or equal to 101. However, these dimensions can easily be changed to accommodate larger problems or to reduce storage space. We employ the following definitions:

NX = number of uniform points in the ξ direction = L_0/DELX + 1.

NY = number of uniform points in the η (or y) direction = YLMX/DELY + 1.

N = total number of spatial grid points (or size of solution vector) = NX * NY.

N4 = 4 * N.

NXP1 = NX + 1.

NYP1 = NY + 1.

NXY = max(NXP1,NYP1).

We now list the exact size of each array in every subroutine that may require a change.

MAIN: AJ(N), XIPT(NX), ETA(NY), XILIM(NX), VEC(N), YO(N)

ASET: UYT(NY), UOX(NY), UTOX(NY), UYOX(NY), UYYOX(NY), DSAVE(NXY), DUSAVE(NXY), UXI(NX), UXI1(NY), UXI2(NY), DVEC(NY)

DIFFUN: UYT(NY), UOX(NY), UTOX(NY), UYOX(NY), UYYOX(NY), UXI(NX), DER(NY,NX), DSAVE(NXY), UXI1(NY), DVEC(NY)

DRIVBI: Y(N,6), YMAX(N), ERROR(N), SAVE1(N), SAVE2(N), DD(N), AA(N4), IPIV(N)

INIT: SAVE(NX)

LDNXI1: UYT(NY), UOX(NY), UTOX(NY), UYOX(NY), UYYOX(NY)

```
LODNYX:  DER(NY,NX)

MESH:  UYT(NY), U(NY), UT(NY), UY(NY), UYY(NY)

TWOD:  VEC(N), YO(N), UYT(NY), UUU(NY)

UFCT:  UYT(NY)

STPRC:  VEC(N), YO(N)
```

6.0 SUGGESTIONS REGARDING CODE USAGE

The computer execution time can be reduced by reducing the number of spatial grid points (N = NX * NY). The CPU time is "roughly" linear with N since most of the computation time is used in solving the linear system (16). One should experiment with various grid spacings (DELX and DELY) to form a good compromise between accuracy and CPU time. In addition, the parameter EPS in subroutine TWOD has been set to 10^{-4} and could be easily changed to 10^{-3} if the user is willing to accept a relative error tolerance in the time integration (DRIVBI) of three significant figures instead of four. This change will have a major effect in reducing computer cost.

The parameters in the common block /GEAR9/ can give useful information about how well the integrator is working. These variables have the following meanings:

NSTEP – the number of integration steps taken up to this time point

NFE – the number of functional evaluations (calls to DIFFUN) up to this time point

NJE – the number of Jacobian evaluations up to this time point

NII – the number of inner (block-SOR) iterations up to this time point

NQUSED – the order (1 to 5) of the time integration last used

HUSED – the time step last used .

Tabulated below is a list of these parameters for an 80 keV boron implant of $3 \times 10^{13}\,\mathrm{cm}^{-2}$ dose, oxidized in steam at 1000°C for half an hour using a 31×51 grid.

Table 2. Integration Parameter Study

T	NSTEP	NFE	NJE	NII	NQUSED	HUSED	Seconds (CPU) (IBM)	Seconds (CPU) (CDC)
.05	14	22	3	39	3	.82(-2)		
.10	20	34	3	63	4	.12(-1)		
.15	24	40	4	75	3	.18(-1)		
.20	27	46	4	87	3	.18(-1)		
.25	29	50	4	96	4	.26(-1)	26.3	10.7
.30	31	54	4	105	4	.26(-1)		
.35	33	57	4	111	4	.26(-1)		
.40	35	59	5	115	3	.38(-1)		
.45	36	60	5	118	3	.38(-1)		
.50	37	61	5	122	3	.38(-1)	30.6	12.5

Typically, NFE should be about 150% of NSTEP and NJE should be less than 20% of NSTEP. If NJE takes on values much larger than this, the integrator is encountering some difficulty and the grid spacing should be changed. Usually, the problem becomes stiffer as the grid is refined, and consequently, NJE may grow. As the solution diffuses, the value of HUSED should increase since the initial large gradients will disappear. Although the order (NQUSED) of the difference scheme begins with one at $t=0$, the value should increase with diffusion. However, small values of NQUSED (1 or 2) are acceptable if HUSED is moderately large compared to its early values. When the integrator encounters discontinuities (possibly between cycles), the order will temporarily be reduced. This difficulty could be avoided by forcing the integrator to exactly integrate to the end of a cycle. Then the integration is restarted. See DRIVBI routine for details on how this may be done.

Occasionally, when the nonuniform growth model is being employed, a few negative impurity concentration values may appear in region 2. This is caused by too coarse an approximation in equation (12e). Refining the grid will alleviate this difficulty.

Generally, the redistribution of an arsenic implant will require more CPU time than a boron one because the initial gradients are much steeper and the diffusivity is smaller for the former implant. In addition, the arsenic profile will require a finer spatial grid to maintain accuracy.

7.0 SUBMITTING A BATCH JOB

The submission of a batch job over the counter requires the following standard job control language with allocation provided for TAPE20 (plotting) and TAPE13 (debugging):

```
// USER=XXXXX,PASSWORD=YYYYYY,
// TIME=(2,30), REGION=1500K
//C EXEC @FORTG
//C.SYSIN DD *
      [USER'S MODIFIED SUBROUTINE OR FUNCTION SOURCE CODES,
       SUCH AS SUBROUTINE UFCT, ETC.; IT CAN BE NO SOURCE.]
/*
//L EXEC @FLINK
//L.SYSIN DD DSN=$SSRH9.MEM1.OBJ.DISP=SHR
//G.FT20F001 DD DSN=$SSRH9.BORON.TAPE.DATA,DISP=SHR  [SEE NOTE]
//G.FT13F001 DD UNIT=SYSDA,SPACE=(TRK,(2,2))
//G.SYSIN DD *
      [MEMBRE INPUT DATA DECK]
/*
```

Note: TAPE20 (FT20F001) saves the results at the end of each processing step for plotting. Users should allocate a data set under their own TSO ID numbers and data set names. The data set name shown on this card (DSN=$SSRH9.BORON.TAPE.DATA) is only an example. TAPE13 is for debugging printout and is also required.

REFERENCES

1. S. Furukawa, H. Matsumura, and H. Ishiwara, Japan J. Appl. Phys., Vol. II, pp. 134-142, 1972.

2. D.D. Warner and C.L. Wilson, Bell Sys. Tech. J., Vol. 59, pp. 1-41, 1980.

3. B.E. Deal and A.S. Grove, J. Appl. Phys., Vol. 36, No. 12, pp. 3770-3778.

4. B.R. Penumalli, "Lateral Oxidation and Redistribution of Dopants," Numerical Analysis of Semiconductor Devices and Integrated Circuits, Proceedings of the NASECODE II Conference, Trinity College, Dublin, June 17-19, 1981, Boole Press, Dublin (1981).

5. C.W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1971).

6. L.A. Hageman, "The Estimation of Acceleration Parameters for the Chebyshev Polynomial and the Successive Overrelaxation Iteration Methods," Bettis Atomic Power Laboratory Report WAPD-TM-1038, 1972.

7. G.E. Forsythe and W.R. Wasow, Finite-Difference Methods for Partial Differential Equations, John Wiley & Sons, New York, 1960.

8. A.C. Hindmarsh, "Preliminary Documentation of GEARBI: Solution of ODE Systems with Block-Iterative Treatment of the Jacobian," Lawrence Livermore Laboratory Report UCID-30149, Livermore, CA, 1976.

9. W.D. Murphy, "Efficient Software for Systems of Nonlinear Parabolic Partial Differential Equations in Two and Three Dimensions," Science Center Technical Report, SCTR-81-3, 1981.

10. W.D. Murphy, W.F. Hall, and C.D. Maldonado, "Efficient Numerical Solution of Two-Dimensional Nonlinear Diffusion Equations with Nonuniformly Moving Boundaries: A Versatile Tool for VLSI Process Modeling," Numerical Analysis of Semiconductor Devices and Integrated Circuits, Proceedings of the NASECODE II Conference, Trinity College, Dublin, June 17-19, 1981, Boole Press, Dublin (1981).

11. A.C. Hindmarsh, et al., "DEC/SOL: Solution of Dense Systems of Linear Algebraic Equations," Lawrence Livermore Laboratory Report UCID-30137, June 1976.

12. C.D. Maldonado, W.F. Hall, W.D. Murphy, and S.A. Louie, "2-D Process Modeling and Simulation for VLSI Design," 1981 Symposium on VLSI Technology Digest of Technical Papers, September 9-11, 1981, Maui, Hawaii.