Monte Carlo Simulation of Ion Implantation into Two- and Three-Dimensional Structures

GERHARD HOBLER AND SIEGFRIED SELBERHERR, SENIOR MEMBER, IEEE

Abstract—**Until now, rigorous Monte Carlo simulations of ion implantation into 3-D structures have been prohibited by the large amount of computer time required. Also 2-D simulations have been restricted to simple structures like linear mask edges or rectangular trenches. In this work methods are presented which make 2-D simulations with arbitrary geometries feasible as well as 3-D simulations with simple geometries. First, an auxiliary grid is used to reduce the time required to check whether an ion crosses a boundary. Second, each ion trajectory is used several times to determine the history of ions entering the target at different positions. The methods are demonstrated by 3 examples: implantation into a rectangular 2-D trench, implantation into a 2-D trench with non-planar sidewalls, and implantation into a 3-D trench with quadratic cross section.**

I. Introduction

IN MOST process simulators, 2-D simulation of ion implantation is based on the point-response approach [1], [2]. In this model the response to a punctiform beam is calculated, assuming an infinite or semi-infinite target. The point response is then moved along the surface and superposed to obtain the total dopant distribution. Multilayer targets are treated by calculating point responses in infinite targets of any of the materials, and using these responses to construct the point response at a given point along the surface depending on the local thickness of the layers [3].

Problems arise when steep surface contours have to be handled. Also the models for multilayer targets are only approximate, especially if the difference between the stopping powers in the materials is large [4], or if the interfaces between the layers are not nearly perpendicular to the initial beam direction.

The problems with steep surfaces shall be illustrated by two examples. In the first example (Fig. 1) a boron implantation at 100 keV is performed around a vertical mask edge. The results of the point-response approach are shown in Fig. 1(a), the results of a full Monte Carlo simulation in Fig. 1(b). The dopant distribution differs at shallow depths near the mask edge. The additional concentration in Fig. 1(b) originates from ions which enter the oxide near the edge, leave the mask laterally and reenter the target in the bulk region.

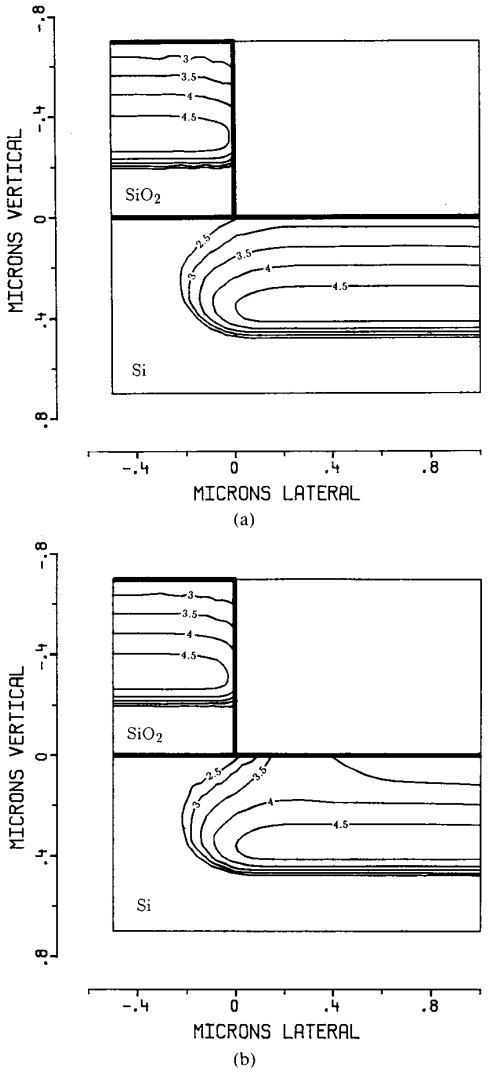

Fig. 1. Boron implantation (100 keV) by a vertical mask edge. The contour lines represent the logarithm of the dopant concentration divided by the dose (cm^{-1}). (a) Simple point response approach. (b) Full Monte Carlo simulation.

A second example is shown in Fig. 2: 25 keV boron ions are implanted at a tilt angle of 7° into a rectangular

Manuscript received June 30, 1988; revised October 28, 1988. This work was supported by DIGITAL Equipment Corporation, Hudson, MA, by the research laboratories of Siemens, AG, Munich, Germany, and by the "Fonds zur Forderung der wissenschaftlichen Forschung" under Project S43/10. The review of this paper was arranged by Guest Editor M. Pinto.

G. Hobler is with the Institut für Allgemeine Elektrotechnik und Elektronik, Technical University of Vienna, A-1040 Vienna, Austria.

S. Selberherr is with the Institut für Mikroelektronik, Technical University of Vienna, A-1040 Vienna, Austria.

IEEE Log Number 8826353.

0278-0070/89/0500-0450$01.00 © 1989 IEEE

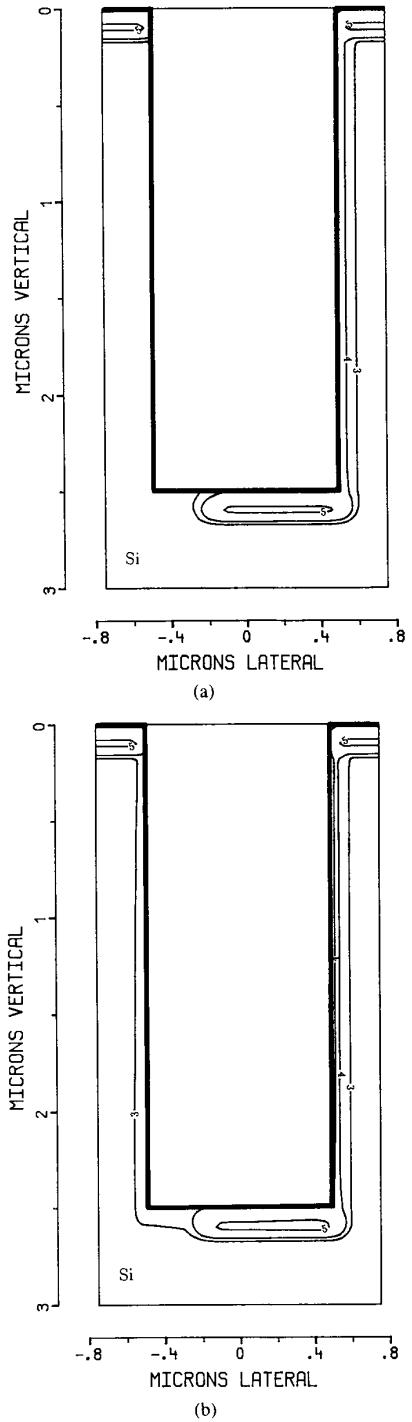

Fig. 2. Boron implantation (25 keV, tilt angle 7°) into an ideal trench. The contour lines represent the logarithm of the dopant concentration divided by the dose (cm^{-1}). (a) Simple point response approach. (b) Full Monte Carlo simulation.

trench. According to the point-response approach (Fig. 2(a)) no ions are found on the shady (left) sidewall of the trench, whereas a considerable dopant concentration occurs there according to the full Monte Carlo results (Fig. 2(b)). This extra amount of dopants on the shady side originates mainly from ions which hit the sunny sidewall, are reflected there, and cross the trench to enter the shady side. The history of these ions is substantially different from the history of ions in an infinite target. Therefore the simple point-response approach cannot produce correct results. Full 2-D simulations can be performed by either solving a Boltzmann Transport Equation [5] or using the Monte Carlo method [6], [7]. The Boltzmann Transport approach requires the discretization of a 5-D space. If discretization is not critical, the Boltzmann Transport method offers a computation speed advantage over the Monte Carlo method. The Monte Carlo method, on the other hand, is considered the most rigorous approach to ion implantation modeling, but it suffers from large computation times. A major part of the CPU-time is due to the evaluation of scattering angles, whereas the other part is proportional to the complexity of the geometry. The latter might be the reason why so far only results on simple geometries have been published [6]. 3-D Monte Carlo or Boltzmann Transport simulations have not been reported in the literature.

The purpose of this paper is to present a 2-D simulator for targets with arbitrary geometries as well as a 3-D simulator for simple geometries. Basic features of these simulators are described in Section II, methods which reduce computation times in Section III. The discussion will concentrate on the 2-D case, as the extension of all methods to 3-D is straightforward. Some simulation results on the sidewall doping of trenches are presented in Section IV.

II. Basic Features

2.1. Ion-Target Interaction

In our simulations we assume amorphous targets. Deviations from actual implantation profiles in crystalline targets are known [8]. However, apart from difficulties in accurately modeling ion implantation in crystalline targets, simulation times would increase by orders of magnitude.

In the Monte Carlo approach individual ion trajectories are simulated. Each ion is followed as it interacts with target atoms and electrons. Interaction with target atoms is modeled as binary collisions separated by free flight paths. The length of the free flight paths is a function of the ion energy and is evaluated by an analytical formula [9]. Scattering angle and energy loss in each nuclear collision are evaluated by a table look-up procedure [7] which is considerably faster than the analytical formula given by Biersack [10]. To describe the ion-target atom interaction, the Ziegler–Biersack ("Universal") potential is used, which is reported to agree well with experimental data [11]. Electronic stopping is considered as a dragging force along the free flight paths. The electronic stopping power is determined by interpolation in a table which is computed based on the modified Brandt–Kitagawa theory

[9]. All features described above, except for the table look-up procedure for nuclear scattering, agree with [9].

2.2. Geometry

Simulations that are 2- or 3-D require that at least two regions with different physical properties be present, one of them possibly free space (vacuum). When an ion is followed during slowing down, it is always necessary to know in which region the ion is located. The material will influence nuclear scattering and electronic energy loss as well as the length of the free flight path. A special treatment is required for free space: The free flight path has to be extended to the next point of entrance into the target or to the point where the simulation area is left, without any change of energy.

In order to determine in which region an ion is located, we distinguish two cases: convex and arbitrary (i.e., possibly concave) regions. We restrict ourselves to regions bounded by polygons (plane faces in 3-D). Convex polygonal areas can be considered as intersection of proper half-planes (Fig. 3). Therefore, in case of convex regions, we check after every free flight path if the ion is inside all of the half-planes:

$$a_i \cdot x + b_i \cdot z + c_i \begin{cases} > 0 \\ \text{or} \\ < 0 \end{cases}, \quad i = 1 \cdots n \quad (1)$$

where x and z are the coordinates of the ion and n the number of boundary lines. a_i, b_i, c_i, and the ">" or "<" sign determine the half-planes. The computational expense is two floating point operations and one comparison for each boundary line.

Concave regions could be subdivided into convex regions. But this would introduce new boundary lines and would increase the number of regions to be handled. We want to avoid this mainly because we use an auxiliary grid for every region (see Section 3.1) and the memory required increases with the number of grids.

There are a number of methods for concave regions. We prefer to intersect the free flight path under consideration with all boundary lines of the region where the last collision has taken place. If no intersection point is found, the ion remains in the same region. In the other case we have to determine the new region. For that purpose we have initialized an array which assigns each boundary line of each region the adjoining neighbour region. Thus if we know which boundary line is crossed we immediately know the new region.

Intersecting two straight lines (free flight path and boundary line) mathematically means to solve a system of two linear equations:

$$\vec{x}_j + \lambda \cdot (\vec{x}_{j+1} - \vec{x}_j) = \vec{p}_i + \mu \cdot (\vec{p}_{i+1} - \vec{p}_i). \quad (2)$$

$\vec{x}_j$, $\vec{x}_{j+1}$ delimit the free flight path and $\vec{p}_i$, $\vec{p}_{i+1}$ the boundary line. In addition it must be checked if

$$0 \le \lambda < 1; \; 0 \le \mu < 1. \quad (3)$$

This procedure requires between five and eight floating-point operations and one to four comparisons. Special care must be taken to avoid rounding errors because grazing angles cannot be ruled out and a crossing event not detected will in general lead to a completely wrong ion trajectory. So we need up to six other floating-point operations and up to two other comparisons. Like in the case of convex regions the expense is proportional to the number of boundary lines, but the expense per line is larger.

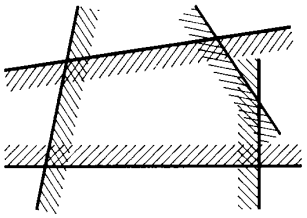

Fig. 3. Convex polygonal region considered as intersection of half-planes.

From the description above it should be plausible that for complicated geometries with many boundary lines the computation time is dominated by geometry checks. This is the motivation for the method presented in Section 3.1.

2.3. Number of Particles

To perform an actual simulation, we have to equidistribute all particles over the width of the simulation area. Two options are possible: First, we can randomly choose the initial lateral position of each ion. This method is correct from a probabilistic point of view. But it introduces additional fluctuations in the results due to fluctuations in the distribution of the initial points. Secondly, we can let the ions start at equidistant lateral positions. As we simulate a large number of particles, the distance between neighbouring ions will be much smaller than any detail of the geometry. Therefore we do not introduce considerable correlations between trajectories (this would be the case, e.g., if the surface had a periodicity corresponding to the distance between neighboring ions). We use this method.

The number of particles to be simulated is a major concern as the simulation time will be proportional to this quantity. It will depend on the desired accuracy, on the concentration range of interest (i.e., the ratio between maximum concentration and lowest concentration of interest), and on the desired spatial resolution. The spatial resolution is determined by the size of the histogram boxes which, in turn, has to be correlated with a characteristic length of profile variation. Such a characteristic length is the lateral standard deviation of a corresponding point response. Assuming that we need a fixed number of histogram boxes per lateral standard deviation, we may roughly say that the number of particles required is proportional to the number of lateral standard deviations which fit into the width of the simulation area. Correspondingly, in 3-D the number of particles has to be proportional to the number of squares with an edge length of the lateral standard deviation which fit into the surface area exposed to the computational beam.

For reasonable results, about 10^4–10^5 particles per lateral standard deviation are required. In the trench exam-

ple of Fig. 2, 50 standard deviations fit into the simulation width. Thus, taking 40000 ions per lateral standard deviation, a total of $50 \cdot 40000 = 2 \times 10^6$ particles is required. For a corresponding 3-D trench with quadratic cross section $2500 \cdot 40000 = 10^8$ particles would be required. On a VAX 8800 this would take about 1 CPU-day and 2 CPU-months, respectively, which is obviously impractical.

III. Advanced Features

3.1. Auxiliary Grid

The idea behind the auxiliary grid shall be demonstrated by Fig. 4. An ion is implanted into a rectangular region, starting in the middle of the top boundary line. Note that the length of all free flight paths is much smaller than any detail which can be seen. It is modeled as a unique function of the ion energy and decreases when the ion slows down. We have to ask after every free flight path, if the ion has left the region. According to the second method of Section 2.2, we have to check if the free flight path intersects any of the 4 boundary lines. However, using the simple grid shown in Fig. 4, most of these checks can be avoided. For instance, if the ion is located inside grid element 2, we know that it cannot cross any boundary but the top boundary line within a single free flight path, because any point inside grid element 2 has a larger distance from all other boundary lines than the free flight path. Analogously, if we know that the ion is located in element 5, we need not do any checks at all, because the distance of element 5 from all boundaries is larger than the free flight path.

To implement the auxiliary grid in the program, the following has to be done: During the initialization phase (1) the grid has to be generated. Details are discussed below. (2) for every grid element (indexes i_x, i_z) the minimum distance $d(i_x, i_z)$ of any point inside the element from the boundaries is calculated. (3) for every grid element those boundary lines are determined which lie within a distance $L_{\max}$ from the element, where $L_{\max}$ is an upper limit to all possible free flight paths. As the length of the free flight paths always decreases during slowing down, $L_{\max}$ can be set to the length of the first free flight path. We obtain an array of logical values, each indicating if a particular boundary line is within a distance $L_{\max}$ from the grid element. As many computers do not store logical variables efficiently, we encode the information in one or more integer words. These are denoted by *near* (i_x, i_z).

During the simulation we have to determine, (1) in which grid element (i_x, i_z) the ion is located, (2) whether the length of the free flight path L is larger than $d(i_x, i_z)$, and if this is the case, (3) which boundary lines are within a distance $L_{\max}$ from the grid element, by decoding *near* (i_x, i_z). Then we have to check these boundary lines on possible intersection points with the free flight path. On the other hand, if L is smaller than $d(i_x, i_z)$, we know that the ion will not cross a boundary during the next $d(i_x, i_z)/L$ free flight paths. Therefore we may skip all geometry checks on the next $d(i_x, i_z)/L$ free flight paths.

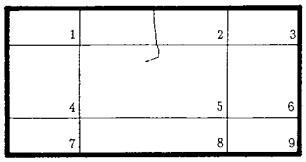

Fig. 4. Implantation of an ion into a rectangular simulation area. A simple grid with 9 mesh elements is used.

In general, simple grids already reduce CPU-times considerably. Further improvement can be achieved when the grid is refined at the boundaries. Then in case of small free flight paths the ion will earlier enter a grid element which has a larger distance from the boundaries than L. In order not to waste computer memory, the grid should be coarse in the middle, i.e., it should be non-equidistant. At the same time we have to take care that the computation of the mesh index from the ion coordinates remains simple, as the method is not reasonable unless the computation of the mesh index is much faster than the geometry checks.

For easy index computation we demand that the grid be part of an equidistant grid, i.e., the grid can be constructed from an equidistant grid by leaving lines away. Then we can first calculate the indices of the equidistant grid j_x, j_z from the ion coordinates x, z (compare Fig. 5)

$$j_x = \frac{x - x_0}{\Delta x}; \qquad j_z = \frac{z - z_0}{\Delta z} \tag{4}$$

and then look in a table, which indices of the actual grid i_x, i_z correspond to j_x, j_z:

$$i_x = i_x(j_x); \qquad i_z = i_z(j_z). \tag{5}$$

The table for the indexes in x-direction of Fig. 5 would read

$$\begin{aligned} i_x(1) &= 1 \\ i_x(2) &= 2 \\ i_x(3) &= 2 \\ i_x(4) &= 3. \end{aligned} \tag{6}$$

The following algorithm is used to decide which lines of the equidistant grid be removed: First every grid element is assigned the information whether the minimum distance between any point inside the element and the boundaries is larger than the smaller one of the horizontal and vertical grid spacing, min $(\Delta x, \Delta z)$. In Fig. 6(a) these elements are hatched. Then we consider the grid lines one by one. We look at all pairs of grid elements which are lined up along the grid line, each having one element on the one side and one element on the other side of the line. For each of these pairs we compare the information assigned to the two elements. If we find the same information in either element, these elements may merge. If this is the case for all pairs along the line, it is removed, otherwise it is maintained. The result of this procedure is shown in Fig. 6(b).

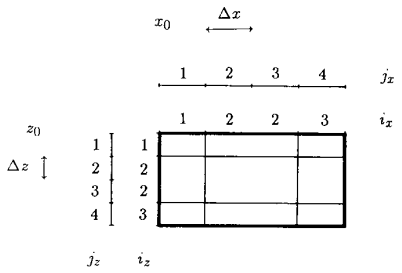

Fig. 5. Mesh indices for a simple grid.

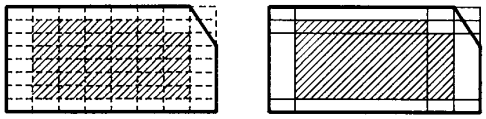

Fig. 6. Design of an auxiliary grid (b) from an equidistant grid with given grid spacings (a).

This grid is best for free flight paths L slightly smaller than min $(\Delta x, \Delta z)$. If L is larger than min $(\Delta x, \Delta z)$, many of the inner elements are not farther from the boundaries than L so that geometry checks are necessary for ions inside these elements. On the other hand, if L is much smaller than min $(\Delta x, \Delta z)$, it will take the ion a large number of collisions after entering the region to enter an inner element where no checks have to be performed. For that reason we compose our actual grid from grids with different mesh spacings of the underlying equidistant grid. It is convenient to take $2^n \times 2^n$ grids, $n = 2 \cdots n_{\max}$. This is shown for the example of Fig. 6 with $n_{\max} = 4$ in Fig. 7. This procedure has the side-effect that for small free flight paths the inner grid elements are many free flight paths away from the boundaries so that geometry checks (including index computation) need only rarely be carried out for ions inside these elements.

Typical maximum dimensions are 1024×1024 for the equidistant grid and a total of 8000 mesh elements for the actual grid. The code generates grids from $2^n \times 2^n$ equidistant grids, starting with $n = 2$, until one of these limits is reached. Some examples of grids, using these maximum dimensions, are shown in Fig. 8.

3.2. *Superposition Method*

The auxiliary grid described in the previous section eliminates most of the CPU-time spent on geometry checks. In this section we try to reduce the time required for the evaluation of the ion-target interaction. The idea is to use each ion trajectory several times to determine the history of ions entering the target at different points. In this way quantities like scattering angles need only be calculated once for a set of trajectories. To justify the method, we make use of the superposition law.

The superposition law holds if the history of all ions are independent from each other. This is the case for amorphous targets. One formulation of the superposition

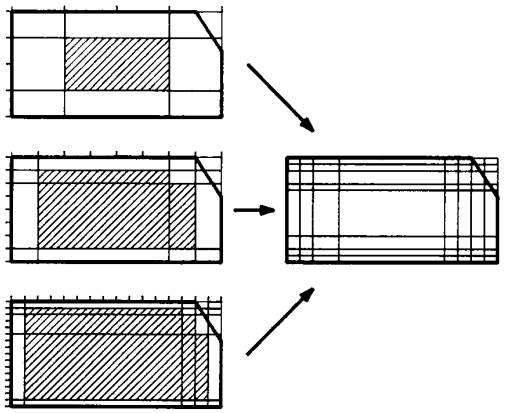

Fig. 7. Superposition of grids based on $2^n \times 2^n$ equidistant grids, $n = 2, 3, 4$.

law says that we may subdivide the area exposed to the ion beam into subwindows, calculate the results of implantations through each subwindow, and superpose the concentrations. This can be written in 2-D (compare Fig. 9):

$$C(x, z) = \sum_i c(x, z; \xi_i, \Delta\xi_i) \tag{7}$$

x, z denote the coordinates of the point where the concentration C is to be calculated, ξ_i and $\Delta\xi_i$ are the lateral position and the width of the subwindows, respectively, and c is the concentration resulting from the implantation through a subwindow. In the following we assume that all $\Delta\xi_i$ be equal (''$\Delta\xi$''). If we let $\Delta\xi \to 0$, we obtain an equivalent formulation with point responses. In this formulation the point response depends on the point of entrance of the punctiform beam into the target. If this dependence is neglected and the point response of an implantation into a semi-infinite target is used, the method is obtained which has been critically discussed in the introduction of this paper.

Equation (7) says that if we have correct results of implantations through subwindows, $c(x, z; \xi_i, \Delta\xi)$, we obtain the correct concentration $C(x, z)$ by superposing all $c(x, z; \xi_i, \Delta\xi)$. Note that we need less particles to calculate $c(x, z; \xi_i, \Delta\xi)$ with the same accuracy as $C(x, z)$, because the particles scatter over a smaller range and are distributed in a smaller number of histogram boxes. This seems to be outweighed by the fact that we have to calculate several responses $c(x, z; \xi_i, \Delta\xi)$. But (7) does not say how the $c(x, z; \xi_i, \Delta\xi)$ be obtained. Therefore, it is not necessary that particles used to calculate one $c(x, z; \xi_i, \Delta\xi)$ are independent from the particles used to calculate another $c(x, z; \xi_j, \Delta\xi)$. In most cases this fact can be used to save a large part of the CPU-time.

The following method is justified by the superposition law: (1) Subdivide the width of the simulation area into N_W subwindows. A convenient value of N_W is such that the width of the subwindows is about the lateral standard deviation. (2) In order to simulate the kth particle, calculate a ''physical'' ion trajectory in an infinite target. (3)

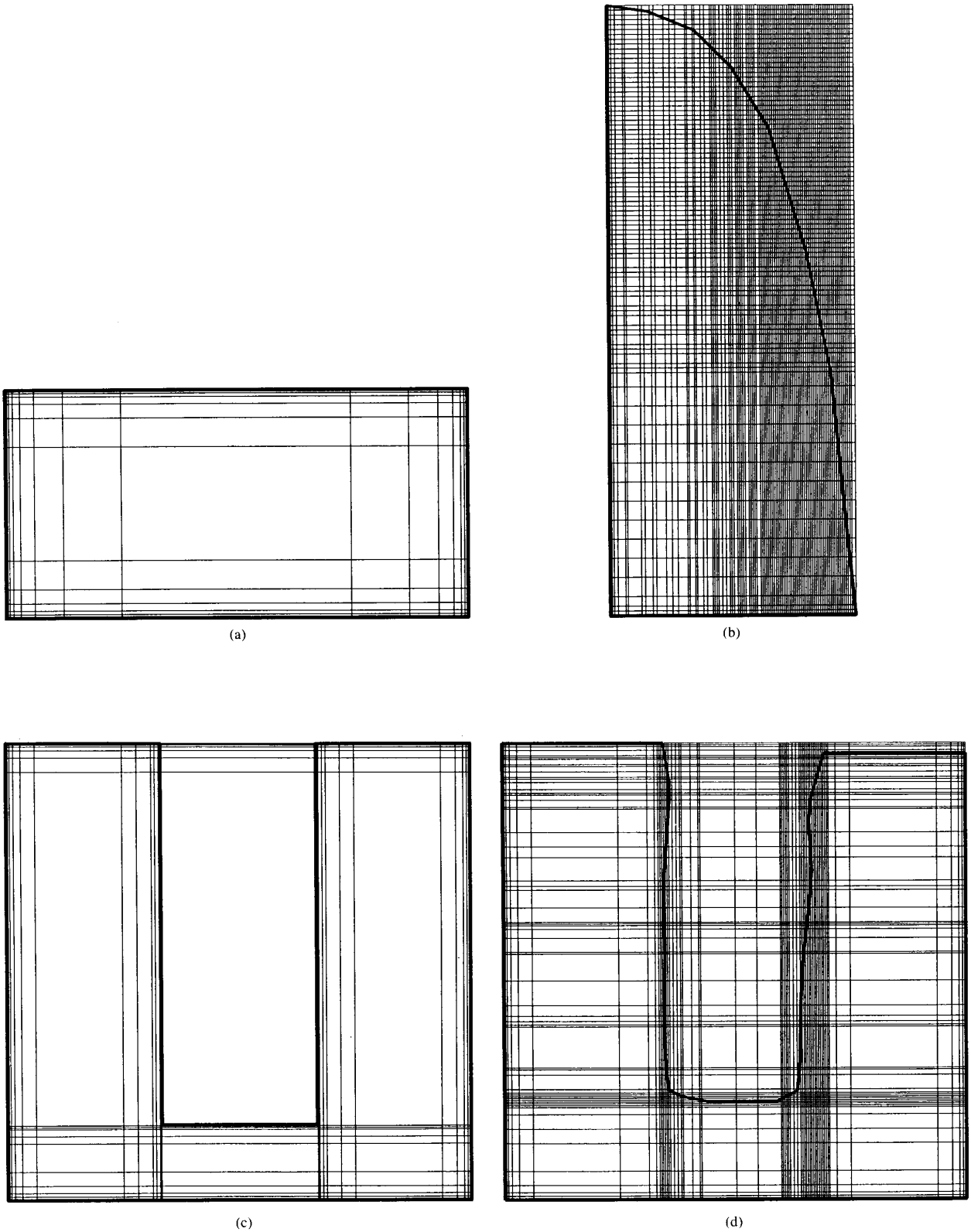

Fig. 8. Examples of auxiliary grids. (a) Rectangular region. (b) Spacer of an LDD-MOSFET. (c) Ideal trench as discussed in Section 4.1. (d) Non-ideal trench as discussed in Section 4.2.

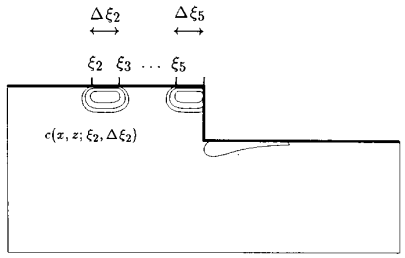

Fig. 9. Response of implantations through subwindows. Only subwindows 2 and 5 are shown.

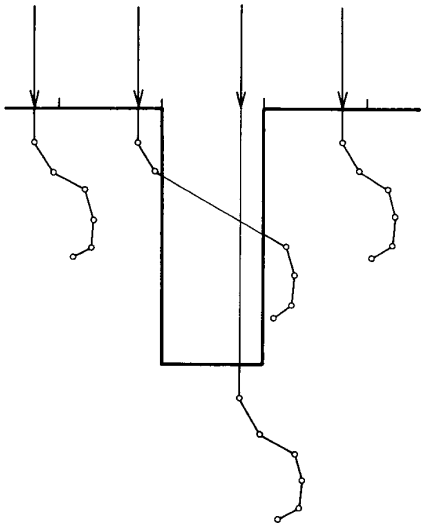

Fig. 10. Construction of trajectories from one "physical" ion trajectory. The width of the subwindows is equal to the width of the trench. Open circles denote the location of collisions. Note that actual free flight paths are much smaller than depicted.

Make copies of this trajectory and move them to corresponding points of each subwindow. The ions have to pass the subwindow at lateral positions $\xi_i + (k - (1/2))/N \cdot \Delta\xi$. If N distinct particles are simulated, all particles will finally be equidistributed over the width of the simulation area. The simulation will be approximately equivalent to a conventional simulation with $N \cdot N_W$ particles. Obviously, the trajectories must be moved vertically so that they start at the surface of the target. These copies may be modified in the next step. (4) Follow each trajectory copy and check if any boundary is crossed. In Fig. 10 the leftmost trajectory never crosses a boundary. Therefore, its endpoint is valid without further action. The second trajectory crosses a boundary after the second collision. If a boundary is crossed, we have to distinguish two cases: (a) the ion enters free space (as in this example). Then we have to follow the current direction to see whether the target is hit again. If so, we have to check, whether the material is the same as the material assumed in the first part of the trajectory. If the same material is found, we can move the rest of the trajectory to the point of entrance. Otherwise we have to simulate the rest of the trajectory conventionally, evaluating the ion-target interaction. (b) the ion enters a region with another material. Then we also have to simulate the rest of the trajectory conventionally. Another situation can occur, when more than one material is encountered at the surface. This can be simply handled by calculating "physical" trajectories for each material. Of course, this will increase the CPU-time.

The method described is most efficient if the target consists of only one material. Then the ion-target interaction need only be evaluated for one "physical" trajectory in order to obtain N_W actual trajectories. In the other case we have to evaluate the ion-target interaction for parts of some or all of the N_W actual trajectories. The method is most inefficient for targets covered by a thin layer, e.g., a thin oxide. Then only a small part of the "physical" trajectory in the infinite target may be used each time. However, usually thin layers can be replaced by equivalent layers of the underlying material for ion implantation simulations.

IV. Example: Sidewall Doping of Trenches

Trenches are used in VLSI-circuits as isolation technique [12] or as capacitors [13]-[15]. In the latter case the sidewalls have to be doped in order to form one electrode of the capacitor. The doping process is performed by diffusion or, more recently, by ion implantation [15], [16]. The implantation has to be done with a tilt angle and the wafer is usually rotated four times by 90°, so that trench walls with any orientation are exposed to the beam. However, due to reflections ions may also hit shady walls.

4.1. Ideal 2-D Trench

Fig. 11 shows the results of a boron implantation (25 keV, tilt angle 7°) into the rectangular trench of Fig. 2. The upper left edge of the representation corresponds to the wafer surface. The trench extends in vertical direction from 0 to 2.5 μm and in lateral direction from -0.5 to 0.5 μm. A considerable concentration can be seen on the shady (left) sidewall, which is only about half an order of magnitude lower than the concentration on the sunny sidewall. Also it is not constant but increases with depth. This depth dependence is much more pronounced for heavy ions like arsenic.

Because of the simple structure, geometry checks can be made in a very efficient way, although the silicon region is concave: The ion is inside the target if ($z > 0$ and $x < -0.5$) or ($z > 2.5$) or ($z > 0$ and $x > 0.5$). We have performed simulations with and without the superposition method. In both cases a total of 2×10^6 particles is used. For the superposition method 50 subwindows are defined, the width of each approximately equal to the lateral standard deviation. Therefore the ion-target interaction is evaluated for $2 \times 10^6/50 = 40000$ trajectories. The results of the two simulations show no deviations at high concentrations and the expected statistical deviations at low concentrations where only few particles are contained in each histogram box. The CPU-times are shown in the first line of Table I. A speed-up factor of about 40

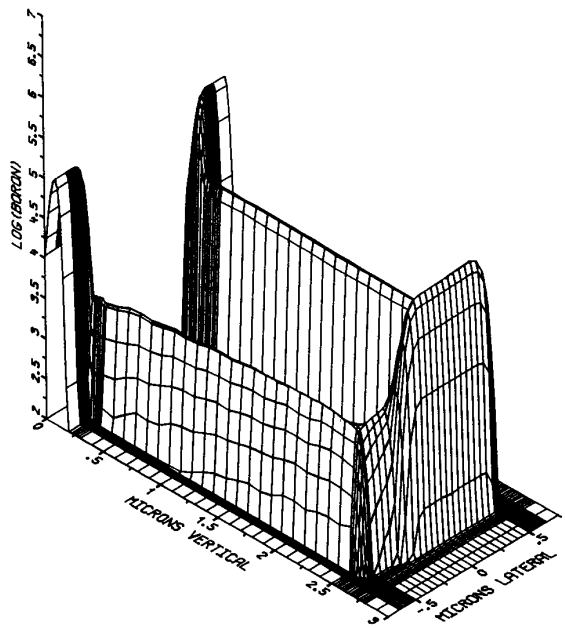

Fig. 11. Boron implantation (25 keV, tilt angle 7°) into an ideal trench. The upper left edge corresponds to the wafer surface. The quantity depicted is the logarithm of the dopant concentration divided by the dose (cm^{-1}).

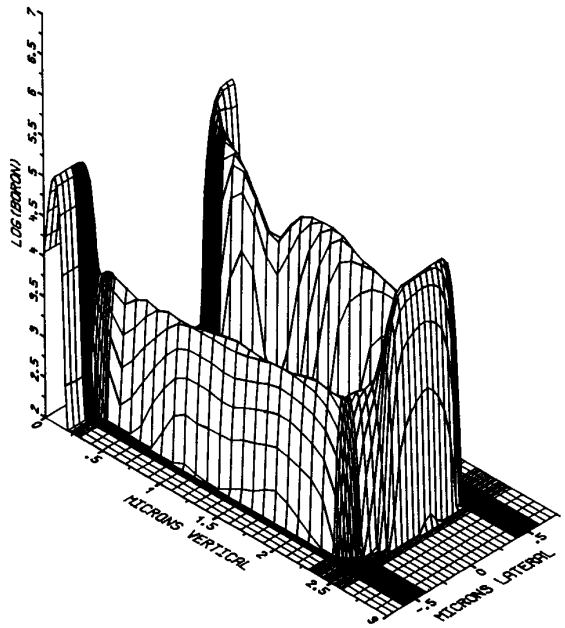

Fig. 12. Boron implantation (25 keV, tilt angle 7°) into a trench with nonplanar walls. The upper left edge corresponds to the wafer surface. The quantity depicted is the logarithm of the dopant concentration divided by the dose (cm^{-1}).

TABLE I
CPU-TIMES OF 2-D TRENCH SIMULATIONS ON A VAX 8800

		without superposition	with superposition
ideal trench	*	16.3 h	25 min
	without grid	25.9 h	8.8 h
	with grid	17.5 h	50 min
non-ideal trench	without grid	43.6 h	24.2 h
	with grid	18.6 h	70 min

* using the efficient geometry check method described in the text

is obtained by the superposition method. As the ion–target interaction is evaluated only once for every 50 trajectories, it can be concluded that the extra expense as compared with the conventional simulation of 40000 particles is only 25 percent.

We have also performed these simulations using the geometry check method for arbitrary regions. Simulation times are listed in lines 2 and 3 of Table I. The CPU-time is reduced by 66 percent using the superposition method alone, by 32 percent using the auxiliary grid alone, and by 97 percent using both methods together. Note that although each method separately saves large absolute values of the CPU-time, only both methods together may change the order of magnitude.

4.2. *Non-Ideal 2-D Trench*

In this section the effects of nonplanarities of the walls due to a nonideal etching process are studied. In trenches the ions hit the sidewalls at grazing angles. Under these conditions the amount of backscattered particles and, therefore, also the amount of particles remaining in the wall are strongly dependent on the angle of incidence. For this reason, nonplanarities lead to inhomogeneous doping profiles at the sunny sidewall.

The geometry of the simulated structure (together with the auxiliary grid) can be seen in Fig. 8(d). The results are shown in Fig. 12. At a depth of 0.8 μm on the sunny side a large local reduction of the concentration occurs. In contrast, the shady side is not severely affected.

Computation times are shown in the last two lines of Table I. Again it is observed that both methods—superposition method and auxiliary grid—are required to drastically reduce the CPU-time. Comparing this simulation with the simulation of the ideal trench using the geometry check method for arbitrary boundaries, we would like to emphasize one point: Employing both of our advanced features, in both simulations about 20 min are spent on the evaluation of the ion–target interaction. Thus in the ideal case 30 min are spent on geometry checks and in the nonideal case 50. This is an increase of 66 percent, mainly due to a worse resolution of the boundaries by the auxiliary grid (compare Figs. 8(c) and 8(d)). At the same time the number of boundary lines is increased from 8 to 25, i.e., by more than a factor 3. That means that using the auxiliary grid, the CPU-time is a strongly sublinear function of the number of boundary lines. Without auxiliary grid, the time spent on geometry checks would be exactly linear with the number of boundary lines.

4.3. *Ideal 3-D Trench*

The 2-D simulations of the previous sections correspond to infinitely long trenches (Fig. 13(a)). Now we

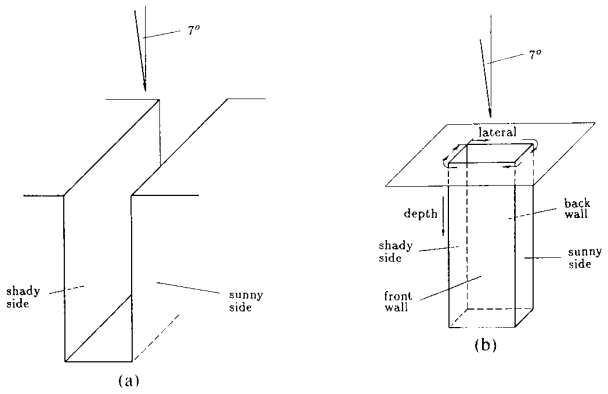

Fig. 13. (a) 2-D trench. (b) 3-D trench with quadratic cross section.

want to investigate a trench with quadratic cross section (Fig. 13(b)). Again, boron is implanted at 25 keV with a tilt angle of 7°, parallel to the front and back wall. We expect that some dopants will be found on the front and back wall, because ions which are reflected at the sunny sidewall usually get a velocity component toward the front or back wall. As these dopants do not reach the shady side, we expect that the concentration on the shady side will decrease.

To compare the 3-D with the 2-D results, we make a cross section in the symmetry plane between front and back wall. The concentration in this plane is depicted in Fig. 14. As compared with Fig. 11, no difference is found on the sunny (right) sidewall. On the shady side the concentration at larger depths is reduced by about a factor 2. Also the concentration is now almost constant with depth.

Finally, we would like to give a global representation of the doping along the sidewalls. For that purpose we have integrated for each point of the wall surfaces the concentration over the direction perpendicular to the wall. This quantity is depicted in Fig. 15. The axis labeled "lateral" covers the lateral coordinate of all four sidewalls as indicated in Fig. 13(b). As expected we find a large constant concentration on the sunny side and a lower, almost constant concentration on the shady side. On the front and back wall the doping is approximately exponentially decreasing toward the shady side of the trench. This is because ions reflected by the sunny sidewall hit the front or back wall near the shady side under more grazing angles than near the sunny side. Anomalous peaks are found near the surface (at a depth of about 0.3 μm). They originate from ions which have first entered the target outside the trench, then left the target through a trench sidewall, and finally hit another sidewall.

For the 3-D simulation, the ion–target interaction is again evaluated for 40000 independent trajectories. 2500 square subwindows are defined so that a total of 10^8 particles is simulated. The geometry checks are performed analogously to the "efficient" method of Section 4.1. The simulation time for this example is 5 h. Most of this time is spent on geometry checks, as these must be carried out

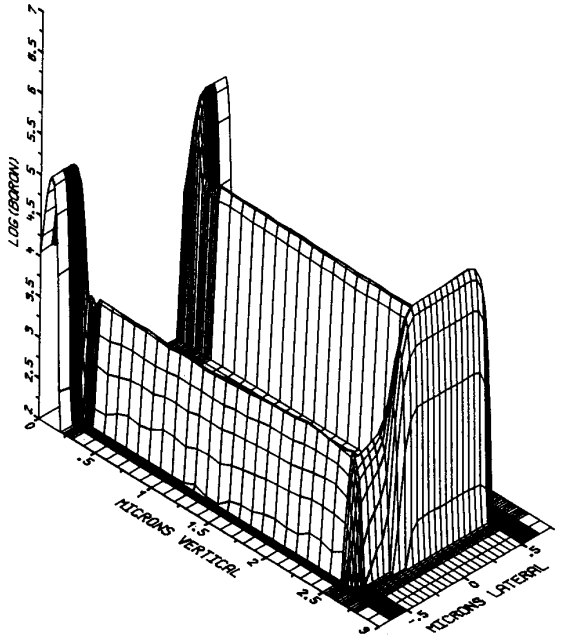

Fig. 14. Concentration in the symmetry plane between front wall and back wall of the trench shown in Fig. 13(b) for a boron implantation (25 keV, tilt angle 7°). The upper left edge corresponds to the wafer surface. The quantity depicted is the logarithm of the dopant concentration divided by the dose (cm^{-1}).

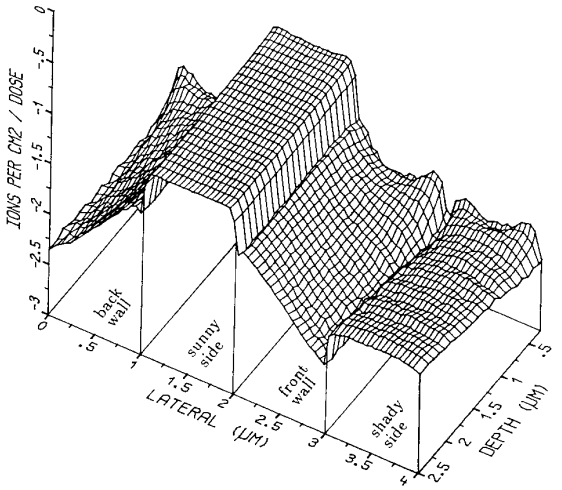

Fig. 15. Concentration integrated over the coordinate perpendicular to the wall surface. Depicted is the logarithm of this quantity divided by the dose. For an explanation of the "lateral" and "depth" axis refer to Fig. 13(b).

for all 10^8 particles whereas scattering angles and energy loss need only be calculated for the 40000 independent particles.

V. Conclusion

Monte Carlo simulation of ion implantation into 2-D structures can be performed at almost the same expense as the computation of point responses. A code allowing

general target structures has been developed and is now part of our 2-D process simulator PROMIS [17]. Typical simulation times are 1 h on a VAX 8800. For lower demands regarding statistical fluctuations these times can be reduced by a factor of about 10. This is especially justified if profiles are broadened and smoothed by subsequent diffusion simulations. 3-D simulations have been demonstrated to be feasible with certain restrictions on target composition and geometry.

Acknowledgement

The authors are indebted to Professor H. Pötzl for carefully reading the manuscript.

References

[1] S. Furukawa, H. Matsumura, and H. Ishiwara, "Theoretical considerations on lateral spread of implanted ions," *Jap. J. Appl. Phys.*, vol. 11, no. 2, pp. 134–142, 1972.

[2] H. Runge, "Distribution of implanted ions under arbitrarily shaped mask edges," *Phys. Stat. Sol.*, vol. A 39, pp. 595–599, 1977.

[3] H. Ryssel, J. Lorenz, and K. Hoffmann, "Models for implantation into multilayer targets," *Appl. Phys.*, vol. A 41, pp. 201–207, 1986.

[4] G. Hobler and S. Selberherr, "Verification of ion implantation models by Monte Carlo simulations," in *Proc. ESSDERC 87*, pp. 445–448, 1987.

[5] M. D. Giles, "Ion implantation calculations in two dimensions using the Boltzmann transport equation," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 679–684, 1986.

[6] H. Ryssel, J. Lorenz, and W. Krüger, "Ion implantation into nonplanar targets: Monte Carlo simulations and analytical models," *Nucl. Instr. Meth.*, vol. B 19/20, pp. 45–49, 1987.

[7] G. Hobler and S. Selberherr, "Efficient two-dimensional Monte Carlo simulation of ion implantation," in *Proc. NASECODE V*, pp. 225–230, 1987.

[8] H. Ryssel, G. Prinke, K. Haberger, K. Hoffmann, K. Müller, and R. Henkelmann, "Range parameters of boron implanted into silicon," *Appl. Phys.*, vol. 24, pp. 39–43, 1981.

[9] J. F. Ziegler, J. P. Biersack, and U. Littmark, *The Stopping and Range of Ions in Solids*. New York: Pergamon, 1985.

[10] J. P. Biersack and L. G. Haggmark, "A Monte Carlo computer program for the transport of energetic ions in amorphous targets," *Nucl. Instr. Meth.*, vol. 174, pp. 257–269, 1980.

[11] D. J. O'Connor and J. P. Biersack, "Comparison of theoretical and empirical interatomic potentials," *Nucl. Instr. Meth.*, vol. B 15, pp. 14–19, 1986.

[12] H. Goto and K. Inayoshi, "Trench isolation schemes for bipolar devices—Benefits and limiting aspects," in *Proc. ESSDERC 87*, pp. 369–372, 1987.

[13] P. Chatterjee *et al.*, "Trench and compact structures for DRAMs," in *Proc. IEDM 86*, pp. 128–131, 1986.

[14] K.-H. Küsters *et al.*, "A high density 4 Mbit DRAM process using a fully overlapping bitline contact (Fobic) trench cell," in *Proc. 1987 Symp. VLSI Technology*, pp. 93–94, 1987.

[15] K. Mashiko, M. Nagatomo, K. Arimoto, Y. Matsuda, K. Furutani, T. Matsukawa, M. Yamada, T. Yoshihara, and T. Nakano, "A 4-Mbit DRAM with folded-bit-line adaptive sidewall-isolated capacitor (FASIC) cell," *IEEE J. Solid State Circuits*, vol. SC-22, pp. 643–650, 1987.

[16] G. Fuse, H. Ogawa, Y. Naito, K. Tamura, K. Tateiwa, H. Iwasaki, "Doping of sub-half μm trench capacitors by grazing incident ion implantation," in *Proc. 1988 Symp. VLSI Technology*, pp. 75–76, 1988.

[17] P. Pichler, W. Jüngling, S. Selberherr, E. Guerrero, and H. Pötzl, "Simulation of critical IC-fabrication steps," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 384–397, 1985.

*

Gerhard Hobler received the Diplomingenieur degree in communications engineering from the Technical University of Vienna in 1985. His master's thesis dealt with Monte Carlo simulation of ion implantation. After graduation he joined the "Institut für Allgemeine Elektrotechnik und Elektronik" at the Technical University of Vienna to work towards his doctor's degree. He was appointed assistant professor in 1988. His dissertation will be on two-dimensional process modeling and process simulation.

*

Siegfried Selberherr (M'79–SM'84) received the degree of Diplomingenieur in control theory and industrial electronics from the Technical University of Vienna in 1978. Since that time he joined the Institut für Allgemeine Elektrotechnik und Elektronik (previously called the Institut für Physikalische Elektronik) at the Technical University of Vienna as professor. He finished his thesis on 'Two Dimensional MOS-Transistor Modeling' in 1981.

He has been holding the 'venia docendi' on 'Computer-Aided Design' since 1984. He is the head of the 'Institut für Mikroelektronik' since 1988. His current topics are modeling and simulation of problems for microelectronics engineering. He has authored and coauthored more than 100 publications in journals and conference proceedings. Furthermore, he wrote a book *Analysis and Simulation of Semiconductor Devices*. In 1983 Dr. Selberherr received the 'Dr. Ernst Fehrer' award; in 1985 he received the award of the 'Nachrichtentechnische Gesellscaft', in 1986 he has been honoured with the 'Dr. Herta Firnberg Staatspreis' and in 1987 he received the 'Heinz Zemanek' award. He is editor of *The Transactions of the Society for Computer Simulation*, of *Electrosoft*, of 'Mikroelektronik' and of the Springer-Verlag book series *Computational Microelectronics*.

Dr. Selberherr is a member of the Association for Computing Machinery, the Society of Industrial and Applied Mathematics, and the Verband deutscher Elektrotechniker.