EFFICIENT TWO-DIMENSIONAL MONTE CARLO SIMULATION OF ION IMPLANTATION

G. Hobler, S. Selberherr
Institut für Allgemeine Elektrotechnik und Elektronik
Technical University of Vienna
Gußhausstraße 27-29, A-1040 Vienna, AUSTRIA

Abstract — We have developed a Monte Carlo code for 2D-simulations of ion implantation which allows fairly arbitrary geometries. To alleviate the problem of large computer times, we apply two methods. First, in the case of complex geometries, most of the time is spent to detect, whether the ions cross boundaries. This extra time—as compared with implantations in infinite targets—may be almost eliminated by putting a grid over each region of the simulation area, and giving each grid element the information whether an ion located inside this element may cross a boundary during the next free flight path. Secondly, we use a precomputed table to evaluate scattering angles θ. Tabulating $\cot\frac{\theta}{2}$ instead of θ, allows moderate table dimensions and small interpolation errors. In a typical example presented in this paper, the execution time could be reduced by a factor of 4 using these methods.

1. Introduction

The Monte Carlo method is the most powerful tool for the simulation of ion implantation. For instance, it poses no problem to treat ions correctly, which leave a mask edge laterally and re-enter the target. This may significantly increase the dopant concentration near the mask edge as compared with what is expected from the commonly used method of superposing point responses. Another typical application of MC-simulations is a trench implantation, where the dopant concentration on the shady side results from ions which have crossed the trench after being reflected at the opposite wall.

The price one has to pay for Monte Carlo simulations is the large amount of computer time required. Therefore it is worthwhile to pay particular attention to the efficiency of MC-codes. In this paper we present two features of our code reducing computer time by about 60–80%. They have been implemented in a code for amorphous targets, but are applicable to crystalline targets as well. They will be described in Chapters 2 and 3. In Chapter 4 a typical example will be presented which demonstrates the time savings.

This work has been substantially supported by the research laboratories of SIEMENS AG at Munich, by DIGITAL EQUIPMENT CORP. at Hudson, USA, and by the "Fonds zur Förderung der wissenschaftlichen Forschung", project S43/10. Helpful discussions and critical reading of the manuscript by Prof. H. Pötzl are gratefully acknowledged.

2. Grid

The key idea of the Monte Carlo method is to simulate a large number of ion trajectories. One of them is shown in Fig. 1a, together with a rectangular simulation area. The computational ion trajectory consists of many small straight lines, which represent the free flight paths between subsequent collisions. It is essential for 2D-simulations that at least some of the ions may leave the present simulation area and enter an adjacent region. For instance, the area of Fig. 1a could represent a mask, adjoining a Si-region at the bottom and a vacuum region at the top and at the right-hand side. Consequently, it must be checked before every collision, if the ion has crossed a boundary during the preceding free flight path.

In the case of a rectangular area, this is a rather simple task: One just has to compare the coordinates of the ion with the coordinates of the boundary lines. This may be generalized for convex regions (bounded by polygons) to checking whether the ion is located in proper half-planes. The situation is getting more complicated when concave areas are allowed. We have decided to detect the crossing of boundaries by checking if the free flight path intersects a boundary line.

In the simple example of Fig. 1a, 50% of the execution time is spent for this purpose. For more complex geometries this extra computer time—as compared with implantations in unbounded targets—increases linearly with the number of boundary lines. However, it may be nearly eliminated by putting a grid over each region of the simulation area. The idea shall be demonstrated by the simple grid of Fig. 1b. The ion starts in grid element 2. Now, if the ion is located in element 2, it is clear that it may not cross any boundary but the top boundary within a single free flight path (note that the free flight path L is much smaller than the grid spacings). Therefore 3 of 4 checks may be saved. When the ion reaches element 5, no checks need to be performed at all, because the distance between any location inside element 5 and the nearest boundary is larger than L.

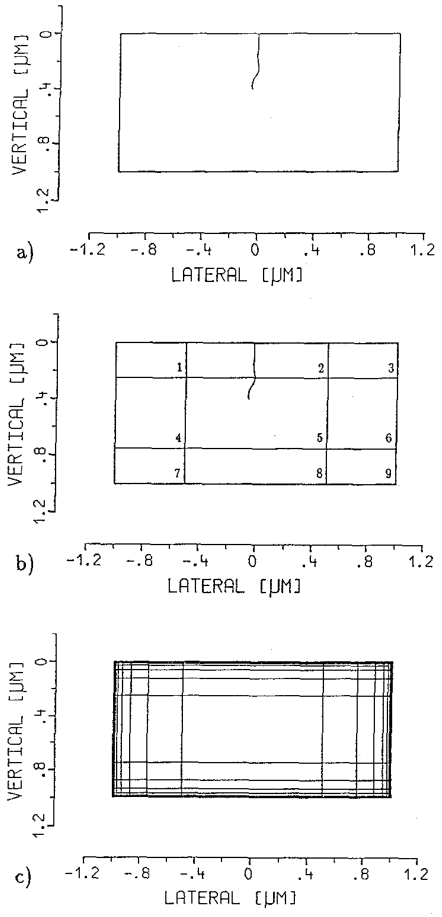

Fig. 1: a) Ion trajectory within a rectangular simulation area.
b) Simple grid.
c) Refined grid.

Some improvement may be achieved by refining the grid at the boundaries. The grid, as produced by our code, is shown in Fig. 1c. The execution time is now nearly identical with that for infinite targets and depends only slightly on the number of boundary lines.

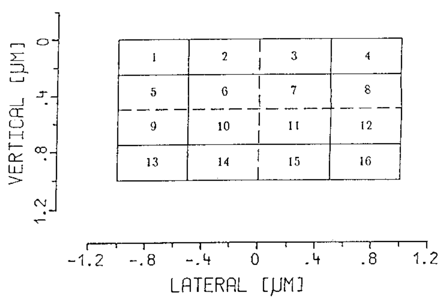

Fig. 2: Grid generation, starting from a 4x4 grid (see text for explanation).

The grid should resolve the boundaries as well as possible, whereas large mesh elements are desired far from the boundaries. The algorithm for the grid generation is as follows: We start with an equidistant grid (in our simple example it is a 4x4 grid, see Fig. 2), and try to remove proper lines. For this purpose we assign to every grid element the information whether the minimum distance between any location inside the element and the boundary is greater than the smaller one of the horizontal and vertical grid spacing. Now we take the mesh lines one by one. We look at all pairs of grid elements, which are lined up along the mesh line under consideration, each of them 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 a line, the line is removed, otherwise it is maintained. In Fig. 2 the dashed lines may be removed. For instance, elements 2/3 and 14/15 are "near" to the boundaries and 6/7 and 10/11 are "far" from the boundaries, so the vertical dashed line may be removed. In contrast, the vertical line on the left may not be removed, because element 5 is "near" to the boundaries whereas element 6 is "far" from the boundaries.*

Finally we assign to every grid element 1) the minimum distance d_{min} from the boundaries and 2) which boundary lines are within the maximum free flight path L_{max}. The latter is binary coded in an integer word. During the simulation of the ion motion it will be checked, 1) whether the next free flight path L is greater than d_{min}, and if this is the case, 2) which boundary lines might be crossed. In this way the calculation of many intersection points may be avoided as discussed above.

3. Table for the Scattering Angle

Using the grid described in the previous chapter, or for implantations in infinite targets, the major part of the simulation time is consumed by computing deflection angles ψ and energy loss ΔE_n in nuclear collisions. The calculation of both ψ and ΔE_n may be reduced to the evaluation of the "scattering integral"

$$\theta(\epsilon, P) = \pi - 2\,P \cdot \int_{R_0}^{\infty} \frac{dR}{R^2 \cdot \sqrt{1 - \frac{\Phi(R)}{\epsilon \cdot R} - \frac{P^2}{R^2}}} \tag{1}$$

which yields the scattering angle in the center-of-mass coordinate system as a function of reduced

* The grid produced by our code is actually a superposition of all grids constructed from equidistant 2^nx2^n-grids as described above, $n = 1 \ldots n_{max}$. A discussion must be omitted for lack of space.

energy ϵ and reduced impact parameter P. R_0 is the zero of the denominator. (For a complete description of the physics of the ion-target interaction see, e.g., Ref. [1].) $\Phi(R)$ is the screening function. We use the Ziegler-Biersack screening function

$$\Phi(R) = 0.1818 \cdot e^{-3.2 \cdot R} + 0.5099 \cdot e^{-0.9423 \cdot R} + 0.2802 \cdot e^{-0.4029 \cdot R} + 0.02817 \cdot e^{-0.2016 \cdot R} \qquad (2)$$

which is reported to agree best with measurements [2].

Numerical integration of Eq. 1 would be extremely time consuming. Biersack has introduced an approximative analytical formula for $\theta(\epsilon, P)$, known as "Magic Formula", which reduces execution times drastically [3]. However, involving several function evaluations (exp, sqrt, ...) and the solution of a nonlinear equation, nuclear scattering still requires by far the major part of the computer time. A further improvement has, in principle, been achieved by Scanlon [4], who reports the use of bicubic spline interpolation in a table for θ as a function of $\log \epsilon$ and $\log P$ with an interpolation error of less than 1%. However, Scanlon's tables cover only the range of $10^{-3} \leq \epsilon, P \leq 10$, so their applicability is restricted.

In our approach, we have tabulated $\cot\frac{\theta}{2}$, $\frac{\partial}{\partial \epsilon}(\cot\frac{\theta}{2})$, and $\frac{\partial}{\partial P}(\cot\frac{\theta}{2})$ as a function of ϵ and P. The knot points for ϵ and P have been carefully adjusted to equidistribute the error of the bicubic interpolation. With the following knot points, a maximum interpolation error of $5 \cdot 10^{-4}$ rad ($\approx 0.03°$) could be achieved in the range of $2 \cdot 10^{-6} \leq \epsilon \leq 10^7$, $0 \leq P \leq 16$:

$$\epsilon = 2^i, \qquad i = -19, -18, \ldots, 5, 7, 12, 24$$
$$P = 0,\ 0.03,\ 0.1,\ 0.35,\ 1,\ 2,\ 3.2,\ 4.5,\ 6,\ 8,\ 10,\ 12,\ 14,\ 16$$

The range of ϵ covers by far all possible values for implantations in semiconductors. The range of P should also cover most applications in semiconductors. To provide reasonable results up to $P = 40$, we have added 3 further P-values (20, 25, 35).

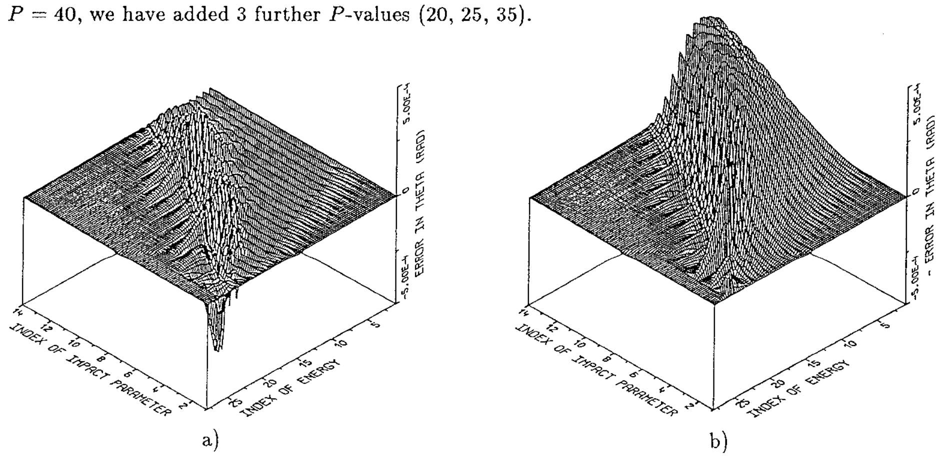

Fig. 3: Absolute error of θ induced by the interpolation in the table for $\cot\frac{\theta}{2}$.
a) θ (interpolated) $-$ θ (exact)
b) θ (exact) $-$ θ (interpolated)

The interpolation error is depicted in Fig. 3. Fig. 3a and Fig. 3b are identical except that in Fig. 3b the representation is reflected at the (error=0)-plane in order to visualize negative values of the error. ϵ- and P-values have been transformed in such a way that the knot points are equidistant on the ϵ- and P-axis. Every integer value on either of the axes represents a knot point. Remarkably small interpolation errors are found between $\epsilon = 2^7$ and $\epsilon = 2^{24}$ (indices 26 and 28). This is because $\cot\frac{\theta}{2}$ is linear in ϵ for high energies and linear in P for small impact parameters. That may be shown by means of Eq. 1 and is true for all reasonable $\Phi(R)$. The asymptotic behaviour of $\cot\frac{\theta}{2}$ is the reason why we have tabulated $\cot\frac{\theta}{2}$ instead of θ. In the next chapter it will be demonstrated that the table approach is nearly 50% faster than using the Magic Formula.

4. Example

In Fig. 4 the structure of a typical source-drain implantation of a LDD-MOSFET is shown. 80 keV As^+-ions have been implanted perpendicular to the surface of the wafer. The contour lines represent the logarithm of the dopant concentration divided by the implantation dose (units $[cm^{-1}]$). The sidewall spacer has been resolved by a 10-point polygon. The spacer forms one homogenous region (SiO_2) for the simulation, the poly-gate and the bulk form two others (Si). Fig. 5 shows the grid which has been generated for the spacer. The grids for the rectangular regions look like the grid in Fig. 1c.

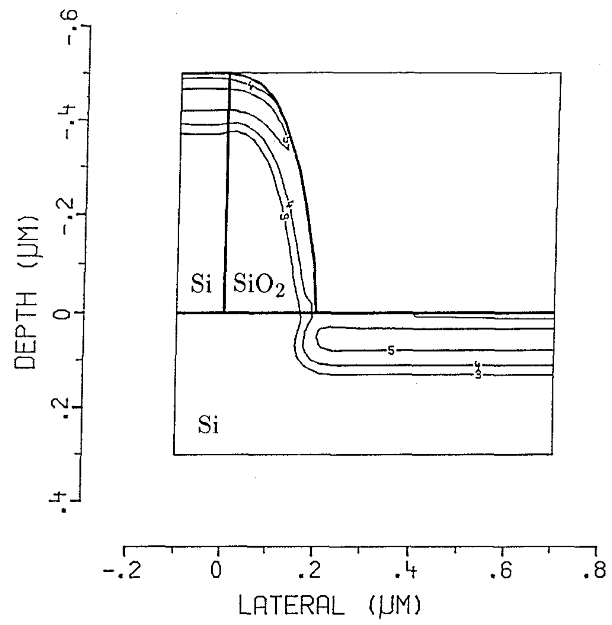

Fig. 4: Structure for the source-drain implantation of a LDD-MOSFET. The gate oxide is not shown. The contour lines represent the logarithm of the dopant concentration devided by the dose $[cm^{-1}]$ as obtained by the Monte Carlo simulation (As^+-ions, 80 keV).

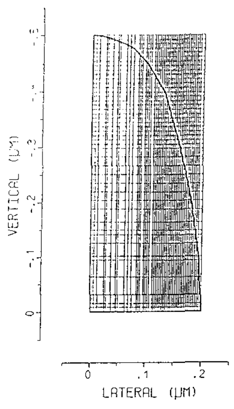

Fig. 5: Grid for the spacer

To investigate the impact of the newly developed methods, we have performed 4 test runs.

1) Without mesh, using Biersack's Magic Formula (= the "conventional" simulation),
2) without mesh, but using the table for the scattering angles,
3) with mesh and Magic Formula, and finally
4) with mesh and scattering table (= the new method).

The results for 1000 ions each on a NAS 9160 computer are shown in the table below.

From the table it can be seen that starting with the conventional method, one may save nearly 50% by introducing the grid and again almost 50% by using the scattering table instead of Magic Formula. On the other hand, if one introduces first the scattering table and then the mesh, one saves 25% and 60%, respectively. It can be concluded that the grid is more important in this context than the table, although both methods contribute substantially to the overall 71% time saving. The efficiency gain would be even greater for more detailed geometries. As mentioned previously, the CPU-time does practically not depend on the number of boundary lines. Therefore the user is relieved from weighing the degree of refinement of the geometry against computer time.

	Magic Formula	Scattering Table
without mesh	42.0 sec.	31.0 sec.
with mesh	22.2 sec.	12.2 sec.

5. References

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

[2] D.J. O'Connor, J.P. Biersack:
"Comparison of Theoretical and Empirical Interatomic Potentials"
Nucl. Instr. Meth., Vol. B15, pp. 14–19, 1986.

[3] J.P. Biersack, 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.

[4] P.J. Scanlon, P.M. Boucher, B. Castel:
"A Fast Numerical Evaluation of Heavy-Ion Scattering Variables"
Nucl. Instr. Meth., Vol. B16, pp. 301–309, 1986.