PAPER Special Issue on 1993 VLSI Process and Device Modeling Workshop (VPAD93)

Monte Carlo Simulation of Ion Implantation for Three-Dimensional Structures Using an Octree

Hannes STIPPEL† and Siegfried SELBERHERR†, Nonmembers

SUMMARY A fully three-dimensional simulation tool for modeling the ion implantation in arbitrarily complex three-dimensional structures is described. The calculation is based on the Monte Carlo (MC) method. For MC simulations of realistic three-dimensional structures the key problem is the CPU-time consumption which is primarily caused by two facts. (1) A large number of ion trajectories (about 10^7) has to be simulated to get results with reasonable low statistical noise. (2) The point location problem is very complex in the three-dimensional space. Solutions for these problems are given in this paper. To reduce the CPU-time for calculating the numerous ion trajectories a superposition method is applied. For the point location (geometry checks) different possibilities are presented. Advantages and disadvantages of the conventional intersection method and a newly introduced octree method are discussed. The octree method was found to be suited best for three-dimensional simulation. Using the octree the CPU-time required for the simulation of one ion trajectory could be reduced so that it only needs approximately the same time as the intersection method in the two-dimensional case. Additionally, the data structure of the octree simplifies the coupling of this simulation tool with topography simulators based on a cellular method. Simulation results for a three-dimensional trench structure are presented.

key words: *ion implantation, Monte Carlo method, point location, octree*

1. Introduction

With the increasing number of devices per wafer area the geometric structures are becoming increasingly complex such that a two-dimensional contemplation is insufficient for modern semiconductor technologies. A tool for the simulation of ion implantation using the Monte Carlo method with the ability of modeling realistic, complex three-dimensional structures has been developed.

The most critical point in three-dimensional simulations is the CPU-time consumption. Several speed-up techniques have been developed to improve the performance of multidimensional Monte Carlo simulation tools for ion implantation to make them applicable for every day usage [1], [2]. Usually, these tools perform quasi-three-dimensional simulations. Even if ion trajectories are computed in a three-dimensional space, for geometry checks, i.e. the determination of the material in which a point is located, it is assumed that the target is infinite at least in the third direction. An approach for the simulation of specific three-dimensional geometries has been presented in [3], where implants into differently shaped trenches were calculated.

In this paper the following two speed-up techniques are discussed. First, the superposition [4] method which allows the simulation of a large number of ions in reasonable CPU-times. Second, an octree [5] was introduced to perform the geometry checks more efficiently. It is demonstrated how this geometry discretization method for the three-dimensional structure speeds up the geometry checks and different point location methods are compared.

2. Superposition Method

For the simulation of ion implantation with the Monte Carlo method a tremendous number of trajectories of ions in a solid have to be computed. An implanted particle looses energy by elastic binary collisions with the target atoms, and is continuously slowed down by the electrons of the lattice atoms. For every trajectory several hundred binary collision events have to be calculated, until the ion comes to rest. To reduce the computational expense the superposition method is used.

It is assumed that ions are only implanted in a certain area, i.e. they enter the target in a restricted field. This area is called the implantation window. For the superposition method the implantation window is divided into separate subwindows. The size of the subwindows is determined by the expected lateral standard deviation σ_{LAT} of the doping profile. A model trajectory is precomputed. For calculating this model trajectory an infinite target is assumed which is filled with a homogeneous material. This trajectory is then copied into every subwindow, so that out of one actually computed trajectory several effective trajectories are derived.

If different materials exist in the structure to be simulated, a model trajectory has to be precomputed for every material. When the particle crosses a boundary between different regions of the simulation area, the pre-computed trajectory is interchanged with the

Manuscript received July 29, 1993.
Manuscript revised October 4, 1993.
† The authors are with the Institute for Microelectronics, Technical University of Vienna, Gußhausstraße 27-29, A-1040 Vienna, Austria.

one for the new material. During the computation of one model trajectory with a certain initial energy other trajectories with a lower initial energy are automatically determined, too [4]. In the case of a reduced initial energy only this part of the model trajectory can be used where the energy lies below this lower energy. This means that on changing the model trajectory the collision event has to be determined where the energy is smaller than the current ion energy. From this point on the new model trajectory is used.

If the energy which is stored for the model trajectory and the actual energy of the ion differ too much it might be necessary that some collisions have to be computed conventionally. When a matching energy is found the model trajectory can be used. Additionally, it is often necessary to twist the model trajectory for two reasons. On one hand, the actual direction of the ion does not necessarily correspond with the direction of the ion in the new model trajectory, and on the other hand, the ion usually does not enter the region perpendicular to the surface which is assumed for the computation of the model trajectory. In either case the model trajectory is rotated accordingly.

3. Point Location Methods

The superposition method decreases the computational expense by reducing the number of collision events to be calculated, since they are only evaluated for the model trajectory. No more computations of collisions have to be performed on determining the effective trajectories by copying the model trajectory to the subwindows. But geometric checks have to be carried out for each effective trajectory. After every collision it has to be tested, if the ion leaves the current region. Consequently, in the three-dimensional case most of the simulation time is consumed to detect whether the particles cross a boundary between different materials. In this case the current model trajectory must be switched over to another one for the new material.

The geometries that have to be treated can be arbitrarily complex. There are no restrictions in any way — for instance to convex shapes — which would simplify the geometry checks; the geometric tests must be very universal. Without special considerations those checks would take tremendous CPU-time consumption and despite today's CPU-power full three-dimensional simulations would not be possible at all. In the following sections different methods for the geometry checks are discussed.

3. 1 Intersection Method

The intersection method [6] (see Fig. 1) is straight forward to apply and is widely used for two-dimensional simulations where the geometry checks are not the main time consumption problem. Nevertheless, some considerations are necessary to avoid numerical problems.

For this method a straight line starting at the current position of the ion and going to infinity is constructed. The number of intersections of this line with all faces which define the actual region is determined. If this count is odd then the point lies inside the region, if it is even then the ion is not in the tested region. In this case the next region has to be investigated. The problem of this method is the CPU-time consumption for the large number of required intersection computations.

The algorithm for the computation of the intersections of the line with the faces is as follows: First the intersection point of the constructed line with a plane

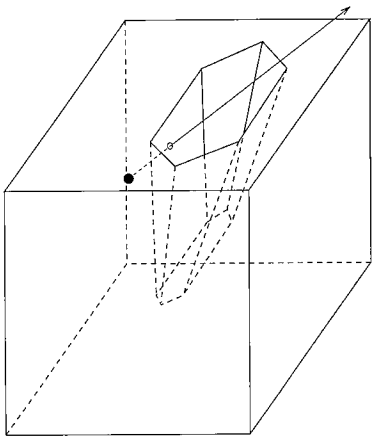


Fig. 1 Point location by intersection method.

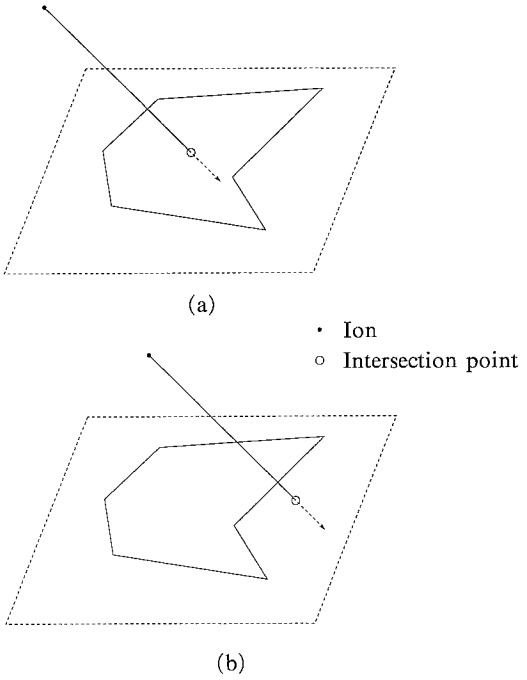


Fig. 2 Intersection point of plane and line.

containing the interesting face is computed. Such a point exists when the line is not parallel to the plane. On the plane this intersection point can be inside (Fig. 2(a)) or outside (Fig. 2(b)) the face. Consequently, it has to be examined further, whether this intersection point is inside the face. For this purpose all points of the face and the intersection point are transformed to the two-dimensional space. This can for instance be done by projecting those points to the x/y-plane, which simply means that the z-coordinate is droped. Then the same intersection counting algorithm can be used to determine if the previously computed intersection point is inside the face or not: Again a line from the current location of the ion to infinity is constructed, but this time in the two-dimensional space. The number of intersections of this line with lines bounding the face is determined. If this number is odd then the point lies inside the face.

To speed this method further up — if more than one region has to be tested — at first all faces that are intersected by this line are marked and then for every region only the marked faces have to be counted. In this manner every intersection is only performed once for a face.

Besides the CPU-time consumption problem, numerical problems can occur when the line starting at the current ion location into infinity lies in a plane of a face, if it goes through a corner point, or if it contains a line defining a face. Therefore careful considerations have to be undertaken to determine the direction of this line.

This method was used for the two-dimensional code. Because of the previously listed disadvantages, an alternative method had to be chosen for the three-dimensional simulator which will be discussed in the following section.

3. 2 Octree Method

As other techniques are too CPU-time expensive a so-called octree was introduced for the representation of the geometric data. This method originates from the graphical image processing (see e.g. [7], [8]). Although in the graphical image processing the contrary task, namely to combine areas with the same attributes, is desired this method can be adopted to be suitable for the ion implantation where one big area must be subdivided into smaller zones.

By use of the octree the simulation area is discretized into cubes. This is done in the following way: At first the whole geometry is included in one cube (the root cube). This cube is then subdivided into eight subcubes, if it is not composed of the same material entirely. This procedure is recursively continued for every subcube until either the desired accuracy of the discretization is reached — which is measured by the sidelength of the cube — or no intersection of this cube with the polygones of the geometry exists.

To clarify the concept of an octree, a triangle which is discretized by a quadtree — this is the two-dimensional equivalent to an octree — is shown in Fig. 3. The principle idea of the octree can be seen in Fig. 4. In the program, a tree-structure is used for the data representation of the octree. Every cube which is further subdivided is represented as a node in the tree, cubes which are not further subdivided are the leafs of the tree. This data representation simplifies the construction of the geometric data representation on one

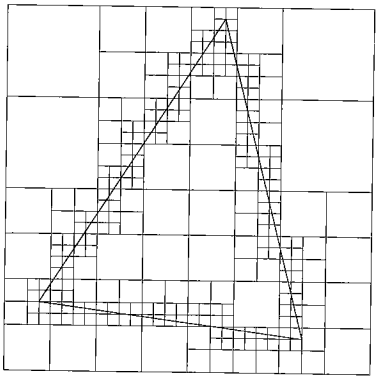


Fig. 3 Discretization of a triangle.

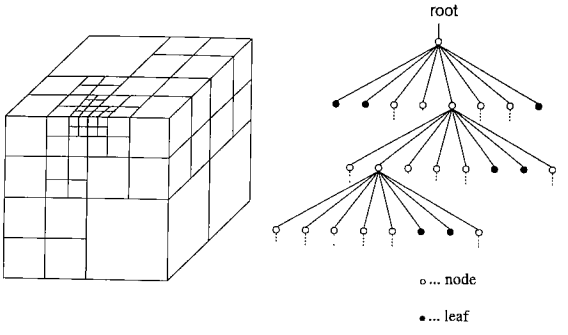


Fig. 4 Discretization using an octree.

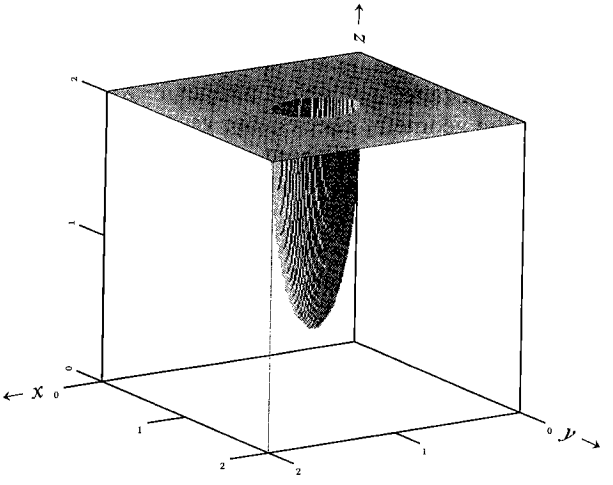


Fig. 5 Geometry derived from topography simulation.

hand and it allows an efficient data storage and fast data access on the other hand.

Once the octree has been created, for the simulation of ion trajectories simple comparisons can be used instead of very complicated and numerically sensitive computations of intersections. To determine the location of an ion, just a simple test of the coordinate against the related coordinates of the sidewalls of the cube is required, because the cube is aligned with the coordinate axes. This leads to a drastic decrease in computation time compared to other methods.

Beside the reduction of computing time, the octree also simplifies the coupling of this module with three-dimensional topography simulators which are based on cellular methods [9]. Such a program produces already a geometry which is discretized into cubes of the same sidelength as output. Those very tiny cubes have only to be combined to cubes that are as large as possible. In this case the structure can even be reproduced exactly. In Fig. 5 a trench derived from such a topography simulation is shown. In this example the geometry is discretized by $256\times256\times256$ cubes. Figure 6 shows the discretization of the trench by an octree resulting in 8667 nodes and 60670 leafs. For this discretization less than 1 second of execution time were

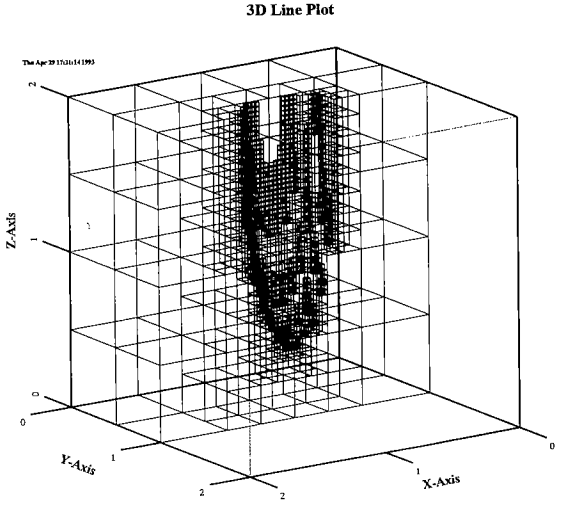


Fig. 6 Discretization of a trench.

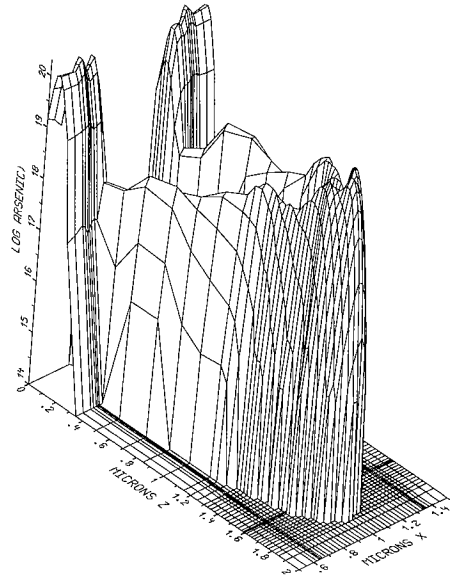


Fig. 7 Cross-Section for constant y coordinate ($y=1$ μm).

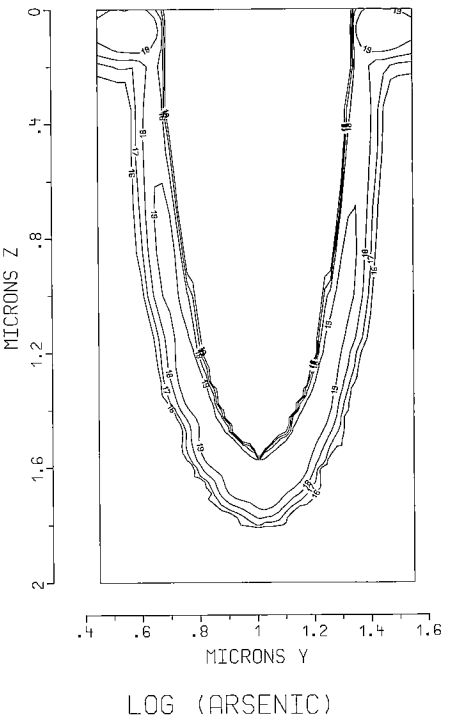


Fig. 8 Cross-Section for constant x coordinate ($x=1$ μm).

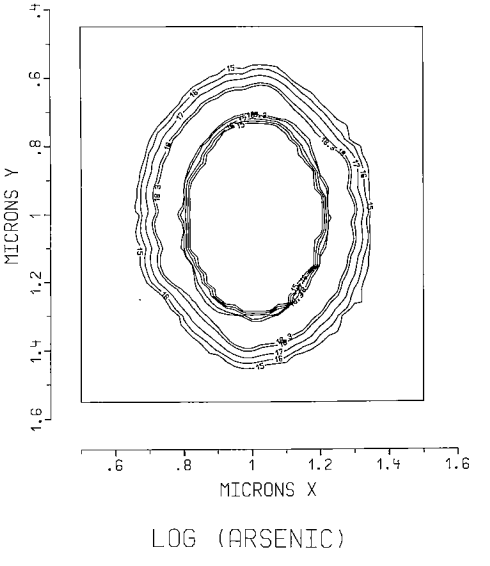


Fig. 9 Cross-Section for constant z coordinate ($z=1.15$ μm).

needed on an HP 9000/735.

The difference of this method to the previously listed method is that the geometry is discretized before the simulation itself is performed. In this manner, not the real but a much simpler geometry is used for the geometry checks. One could say that instead of checking intersections at every collision event, all those intersections are performed once at the initialization step. Moreover, the determination of the minimum distance of the ion's current location to the border of the leaf-cube is very simple. This minimum distance is also the minimum distance which the ion will remain in the same region. Therefore, it is possible to skip some checks at all when this minimum is several times the maximum free flight path of the ion. This gives an additional speed-up.

4. Example

As an example the simulation of an arsenic implantation step (As with 100 keV, no tilt angle) into a trench derived from anisotropic etching simulations is presented (Fig. 5). The discretization can be seen in Fig. 6. In Figs. 7-9 the results are shown. In Fig. 7 the cross-section in the middle of the trench parallel to the vertical y/z-plane can be seen, in Fig. 8 the (again vertical) cross-section for the x/z-plane. In both figures an increased concentration can be found on the bottom of the trench. This is due to the fact that in the steep sidewalls of the trench less ions stop, because the entrance angle is very flat and therefore the probability to exit the target again is higher, too. Due to the symmetric conditions (the symmetric trench, 0° tilt angle, continuous rotation) uniform profiles are obtained. This symmetry can especially be seen in Fig. 9 where the trench was intersected parallel to the x/y-plane. The computation for this example took less than half an hour of computation time on an HP 9000/735.

In Table 1 the CPU-time for this example is shown including a comparison of computations without the described speed-up techniques. From this comparison it can be clearly seen that three-dimensional simulations without such accelerating methods are not possible at all.

5. Conclusion

A superposition method was presented which makes simulations of ion implantation in three-dimensional structures possible. Moreover, the use of the octree to determine the region where an ion is located reduces the CPU-time requirements tremendously in comparison to other point location method for the three-dimensional simulation of ion implantation. Other methods like the intersection method are very CPU-time expensive as well as numerically problematic. Therefore the octree is the optimum solution in the three-dimensional case.

Table 1 Comparison of computation times for the example.

	no superposition	superposition
no octree	86.5 hours	8 hours
octree	4.5 hours	25 minutes

Acknowledgement

This project is supported by the laboratories of: AUSTRIAN INDUSTRIES-AMS at Unterpremstätten, Austria; DIGITAL EQUIPMENT Corporation at Hudson, USA and SIEMENS Corporation at Munich, Germany.

References

[1] Giles, M. D. and Gibbons, J. F., "Two-Dimensional Ion Implantation Profiles from One-Dimensional Projections," *J. Electrochem. Soc.*, vol. 132, no. 10, pp. 2476-1480, 1985.

[2] Van Schiele, E. and Middelhoek, J., "Two Methods to Improve the Performance of Monte Carlo Simulations of Ion Implantation in Amorphous Targets," *IEEE Trans. CAD*, vol. 8, no. 2, pp. 108-113, Feb. 1989.

[3] Stippel, H., Hobler, G. and Selberherr, S., "Three Dimensional Simulation of Ion Implantation," *Proc. ICSICT '92*, pp. 703-705, Oct. 1992.

[4] Hobler, G. and Selberherr, S., "Monte Carlo Simulation of Ion Implantation into Two- and Three-Dimensional Structures," *IEEE Trans. CAD*, vol. 8, no. 5, pp. 450-459, May 1989.

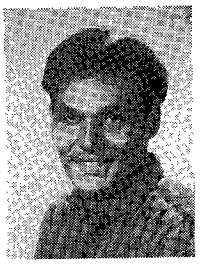
[5] Stippel, H. and Selberherr, S., "Three Dimensional Monte Carlo Simulation of Ion Implantation with Octree Based Point Location," *Proc. VPAD'93*, pp. 122-123, May 1993.

[6] Preparata, F. P., *Computational Geometry, An Introduction*, Springer Verlag, 1985.

[7] Fellner, W. D., *Computergrafik*, Wissenschaftsverlag, 1992.

[8] Mäntylä, M., *An Introduction To Solid Modeling*, Computer Science Press, 1988.

[9] Strasser, E., Wimmer, K. and Selberherr, S., "A New Method for Simulation of Etching and Deposition Processes," *Proc. VPAD '93*, pp. 54-55, Nara, May 1993.



Hannes Stippel was born in Vienna, Austria, in 1966. He received the Diplomingenieur in Control Theory and Industrial Electronics from the Technical University of Vienna in 1990. He joined the Institute for Microelectronics at the Technical University of Vienna in June of the same year. His work is focused on simulation of ion implantation with special emphasis on three-dimensional applications based on the Monte Carlo method.

Siegfried Selberherr was born in Klosterneuburg, Austria, in 1955. He 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. He finished his thesis on "Two Dimensional MOS Transistor Modeling" in 1981. Dr. Selberherr 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 authored and coauthored more than 200 publications in journals and conference proceedings. Furthermore, he wrote a book *Analysis and Simulation of Semiconductor Devices.* Dr. Selberherr is member of the Association for Computing Machinery (1979), the Society of Industrial and Applied Mathematics (1980) and fellow of the The Institute of Electrical and Electronics Engineers. Dr. Selberherr is editor of The Transactions of the Society for Computer Simulation, of Mikroelektronik and of the Springer-Verlag book series Computational Microelectronics.