Three-Dimensional Topography Simulation Using Advanced Level Set and Ray Tracing Methods

Otmar Ertl and Siegfried Selberherr
Institute for Microelectronics, TU Wien
Gußhausstraße 27-29/E360, A-1040 Wien, Austria
Email: {ertl|selberherr}@iue.tuwien.ac.at

***Abstract*—We present new techniques for three-dimensional topography simulation of processes for which ballistic transport can be assumed at feature-scale. The combination of algorithms and data structures lent from the area of computer graphics allows a fast and memory saving solution of various deposition and etching processes.**

I. Introduction

Topography simulation requires two essential ingredients: A method to track the surface and a method to determine the local surface velocities. Various combinations of methods to handle both tasks in three-dimensions are discussed in [1]. A new combination using the level set method for surface representation and a Monte Carlo method for flux calculation was recently reported [2], [3]. In the following we describe techniques which enable topography simulations of large three-dimensional geometries.

II. Surface evolution

A. Level set method

In three dimensions the level set method has become widely accepted for surface tracking [4]. The surface is implicitly described as zero level set of a function Φ

$$\mathcal{S} = \{\mathbf{x} \mid \Phi(\mathbf{x}) = 0\}. \quad (1)$$

The time evolution of the surface $\mathcal{S}$ can then be described by the level set equation

$$\frac{\partial\Phi}{\partial t} + V(\mathbf{x})\|\nabla\Phi\| = 0, \quad (2)$$

where $V(\mathbf{x})$ is the surface velocity field. The level set function is usually discretized on a regular grid. The original level set technique stores and integrates the level set values of all points in the grid over time, leading to a non-linear scaling of memory and computational costs with surface size. To reduce both down to linear order, we use the sparse-field level set method [5] in combination with the recently developed hierarchical run-length-encoded level set data structure [6].

B. Sparse field level set method

The sparse-field level set method is a further development of the narrow band method [4]. This method reduces the narrow band to just one layer of active grid points, namely all points for which

$$|\Phi(\mathbf{x})| \leq 0.5 \quad (3)$$

is fulfilled. Therefore, the calculation time is reduced to a minimum, since only the level set values of a minimum number of grid points have to be integrated over time. For the calculation of derivatives also the level set values of neighboring grid points have to be known. Therefore, additional layers of grid points are necessary. Their level set values are determined by the sparse field level set method using a simple update scheme, which is performed after each time integration step. A further advantage of the sparse-field level set method is that it does not require periodic re-initializations like the narrow band method. Moreover, the velocity field $V(\mathbf{x})$ has to be only calculated for all active grid points. If the surface velocity is determined directly for these points [7] the time consuming fast marching method for the velocity extension [4] can be avoided.

C. Hierarchical run-length encoding

To store a level set function, we use the hierarchical run-length encoding data structure [6]. It only stores the level set values at grid points which are near the surface. For all other grid points just the signs of their level set values are stored using run-length encoding. The memory requirements follow an optimal linear scaling with surface area. Sequential traversal is also optimal, while random access to grid points is of sub-logarithmic complexity. The availability of the sign of the level set function for all grid points makes this data structure especially convenient for multi-level-set methods, where boolean operations like union or intersection can be expressed as the minimum or maximum of two level set functions, respectively [8]. The computational costs of these operations using this data structure are of linear complexity.

D. Multiple materials

To represent regions of different materials the geometry is divided by level sets. One way is to describe each material region M_k by one enclosing level set function Φ_k [9]

$$\Phi_k(\mathbf{x}) \leq 0 \Leftrightarrow \mathbf{x} \in M_k. \quad (4)$$

However, with this representation very thin layers with thicknesses smaller than one grid spacing cannot be resolved. To circumvent this problem, we describe a stack of materials $M_1, M_2, \ldots, M_K$, where M_1 denotes the substrate, by choosing N level sets in such a way that

$$\Phi_k(\mathbf{x}) \leq 0 \Leftrightarrow \mathbf{x} \in \bigcup_{i=1}^{k} M_i. \quad (5)$$

978-1-4244-1753-7/08/$25.00 ©2008 IEEE

Only the top most level set function M_K is integrated over time. However, in case of etching processes the different etching rates are incorporated during time integration. All other level set functions are adjusted following

$$\Phi_k^{(t+\Delta t)}(\mathbf{x}) = \max(\Phi_k^{(t)}(\mathbf{x}), \Phi_K^{(t+\Delta t)}(\mathbf{x})). \tag{6}$$

III. Surface velocity calculation

To determine the surface velocities the transport and surface reaction equations have to be solved. We focus on processes which can be described by ballistic transport at feature-scale. The incoming arrival angle distribution $\Gamma_{src}(\mathbf{t})$ is assumed to be known at a certain plane $\mathcal{P}$ just above the surface. The flux distribution at the surface is given by

$$\Gamma(\mathbf{x},\mathbf{t})d\Omega = \begin{cases} \frac{-\mathbf{t}\cdot\mathbf{n}(\mathbf{x})}{\|\mathbf{x}-\mathbf{x}'\|^2}\Gamma_{src}(\mathbf{t})\,dA' & \text{if } \mathbf{x}' \in \mathcal{P} \\ \frac{-\mathbf{t}\cdot\mathbf{n}(\mathbf{x})}{\|\mathbf{x}-\mathbf{x}'\|^2}\Gamma_{re}(\mathbf{x}',\mathbf{t})\,dA' & \text{if } \mathbf{x}' \in \mathcal{S} \end{cases} \tag{7}$$

where $\mathbf{x}'$ is the origin of a ray with direction $\mathbf{t}$ hitting the surface at point $\mathbf{x}$. $\mathbf{n}(\mathbf{x})$ is the surface normal at point $\mathbf{x}$. The re-emission of particles is described by

$$\Gamma_{re}(\mathbf{x},\mathbf{t}) = \int Q(\mathbf{n}(\mathbf{x});\mathbf{t},\mathbf{t}')\Gamma(\mathbf{x},\mathbf{t}')\,d\Omega'. \tag{8}$$

Here Q denotes the transmission probability function. The surface velocity is assumed to be of the form

$$V(\mathbf{x}) := \int \Gamma(\mathbf{x},\mathbf{t})Y(\mathbf{n}(\mathbf{x});\mathbf{t})\,d\Omega, \tag{9}$$

where Y is the yield function.

A. Direct integration

A common approach for the solution of this system of equations (7) - (9) is direct integration. However, in its general form these equations require a discretization of the surface and also of the solid angle for each surface point. This would result in a huge system of linear equations, unfeasible for three-dimensional problems. Therefore a common simplification is to neglect the dependence of the re-emission on the incoming direction [10]

$$Q := Q(\mathbf{n}(\mathbf{x});\mathbf{t}). \tag{10}$$

Then the system of equations can be reduced to relations between the total incoming fluxes $F(\mathbf{x})$, which avoids the directional discretization

$$\begin{aligned} F(\mathbf{x}) = &\int_{\mathcal{P}} \mathrm{vis}(\mathbf{x},\mathbf{x}')\frac{-\mathbf{t}\cdot\mathbf{n}(\mathbf{x})}{\|\mathbf{x}-\mathbf{x}'\|^2}\Gamma_{src}(\mathbf{t})\,dA' + \\ &\int_{\mathcal{S}} \mathrm{vis}(\mathbf{x},\mathbf{x}')\frac{-\mathbf{t}\cdot\mathbf{n}(\mathbf{x})}{\|\mathbf{x}-\mathbf{x}'\|^2}F(\mathbf{x}')Q(\mathbf{n}(\mathbf{x}');\mathbf{t})\,dA'. \end{aligned} \tag{11}$$

Here $\mathrm{vis}(\mathbf{x},\mathbf{x}')$ is the visibility function which is either 1 or 0 dependent on whether the points $\mathbf{x}$ and $\mathbf{x}'$ are in line of sight or not. However, it is still a demanding task to solve this surface integral equation. Generally, each surface element contributes to each other surface element, if it is in line of sight, resulting in a dense system matrix. Therefore both, the memory requirements and the calculation time for solving this linear system of equations, are expected to follow $\mathcal{O}(N^2)$, N denoting the number of discretized entities. For setting up the equation matrix an even worse scaling can be expected due to the visibility check [11].

B. Monte Carlo method

Another way to calculate the surface velocities is ray tracing, a widely used technique in computer graphics to render three-dimensional scenes efficiently. There, millions of rays are calculated to get a realistic picture. Analogously we calculate a huge number of particle trajectories. Each time a particle hits the surface, it contributes to the local surface velocity. Then the particle is re-emitted following the directional distribution (8). A weight factor, which describes the probability of the particle, is adjusted after each re-emission according to the directional distribution. This factor is used to describe the statistics correctly and is incorporated, when the contribution of a particle to the surface velocity is calculated. The particles are tracked as long as they do not leave the simulation domain upwards or their weight factor goes below a certain limit.

The main computational task within this method is to calculate the intersection of a ray with the surface. Various algorithms and data structures were developed to reduce the calculation time [12]. We use spatial binary subdivision to reduce the effort of calculating one particle ray down to order $\mathcal{O}(\log N)$ [7].

To achieve a certain statistical accuracy the number of simulated particles has to increase with the surface area. Therefore, the whole algorithm scales like $\mathcal{O}(N \log N)$.

In our simulator ray tracing is directly applied to the implicit level set representation of the surface. Three-linear interpolation within one grid cell is used to calculate ray-surface intersections. Hence, no explicit surface representation is needed during the whole simulation, which results in savings of memory and computation time.

The surface velocity is directly determined for all active grid points. A disk of certain radius is defined for each active point $\mathbf{p}$ as shown in Figure 1. The disk is orientated normal to the gradient $\nabla\Phi(\mathbf{p})$ and its distance d is given by

$$d := \frac{\Phi(\mathbf{p})}{\|\nabla\Phi(\mathbf{p})\|}. \tag{12}$$

Thus, the center point of the disk is an approximation of the closest surface point, which guarantees that the disk positions are close to the surface $\mathcal{S}$.

All particles hitting the disk contribute to the surface velocity at $\mathbf{p}$ according to the yield function in (9). Particle trajectories are tracked for a certain distance after intersection with the surface $\mathcal{S}$ to calculate the surface velocity properly.

C. Parallelization

The most time consuming part in each time step of the simulation is the surface velocity calculation. However, by nature, since individual particle trajectories are completely

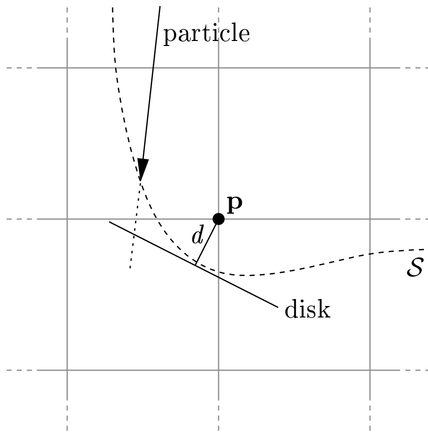

Fig. 1. For each active grid point $\mathbf{p}$ a disk is defined. All particles impinging on the disk contribute to the surface velocity of $\mathbf{p}$. Particle trajectories are tracked for a certain distance beneath the surface $\mathcal{S}$ to obtain a representative flux distribution on the disk.

independent from each other in the ballistic transport regime, the Monte Carlo method can be easily parallelized, especially on shared memory architectures. We used OpenMP [13] to distribute the surface velocity computation over multiple cores.

IV. Examples

In the following we demonstrate the capabilities of our simulator using the above described level set and ray tracing techniques on various examples. For all examples symmetric boundary conditions were assumed for the lateral directions. Since all our data structures are adaptive, the vertical direction is unbounded.

A. Simple deposition process

To prove that ray tracing is also convenient to determine the surface rates for large geometries we applied a deposition process to a test structure with a lateral resolution of 500×500. The result is shown in Figure 2. The process was modeled using a sticking probability of 0.5. The directional distributions of incoming and re-emitted particles were assumed to follow a cosine distribution. In this simulation all higher order re-emissions were incorporated. Due to the adaptiveness of all data structures the total memory consumption does not exceed 500 MB.

B. Reactive ion etching

Figure 3 shows the final profiles after the application of etching processes in SF_6/O_2-plasma with different amount of oxygen. Model and parameters were taken from [14]. The model is based on a Langmuir-Hinshelwood-type adsorption model and incorporates three kinds of species: ions, inhibitors, and etchants. Coverages are introduced for inhibitors and etchants to describe the surface kinetics. Specular reflexions and the energy distribution of ions are also taken into account. The dependence of the sputter yield on angle of incidence and ion energy is modeled as well. In contrast to direct integration methods, all these effects can be easily included using ray

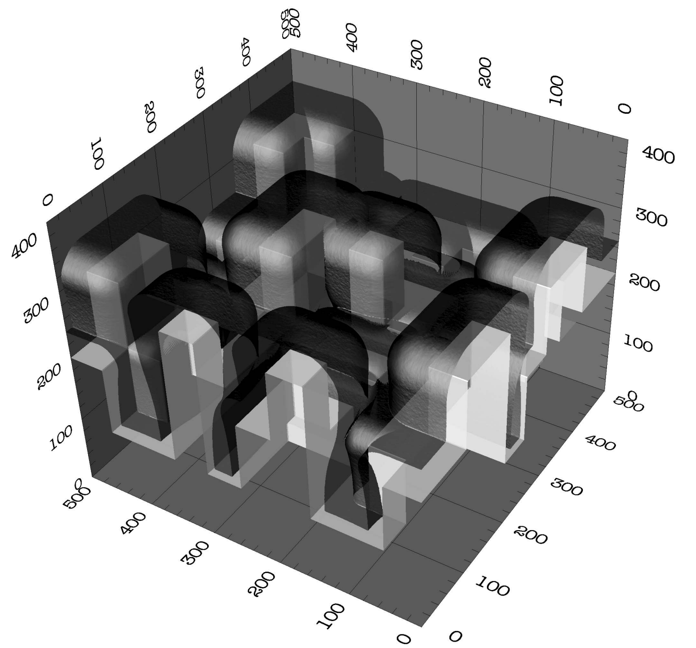

Fig. 2. Deposition process with sticking probability 0.5 applied to a test structure. Lengths are given in multiples of grid spacings.

tracing.
Within this model the sticking probabilities depend on the coverages, which again are obtained from the site balance equations under pseudo-steady-state assumptions. Therefore, the transmission probability function in (8) depends on the flux distribution itself, leading to a recursive problem. In our simulation we use the fluxes calculated in the previous time step to determine the coverages.

C. Bosch process

As demonstration of our multi-level-set framework we simulated a Bosch process using the model given in [15]. Figure 4 shows the final profile after 10 deposition and etching cycles, respectively. This alternation of process steps requires an accurate description of very thin layers as provided by our multi-level-set method. In this simulation three level set functions are used to describe the substrate, the mask, and the polymer layer. If the polymer layer is locally completely removed during a time step, the different etching rates are adequately taken into account during level set time integration to enhance accuracy.

V. Conclusion

We presented techniques for the efficient solution of topography processes, for which ballistic particle transport can be assumed. The application of modern level set and ray tracing algorithms results in an $\mathcal{O}(N\log N)$ scaling of the computational costs and an optimal $\mathcal{O}(N)$ scaling for the memory requirements, which allows the simulation of large three-dimensional structures. Furthermore, the Monte Carlo

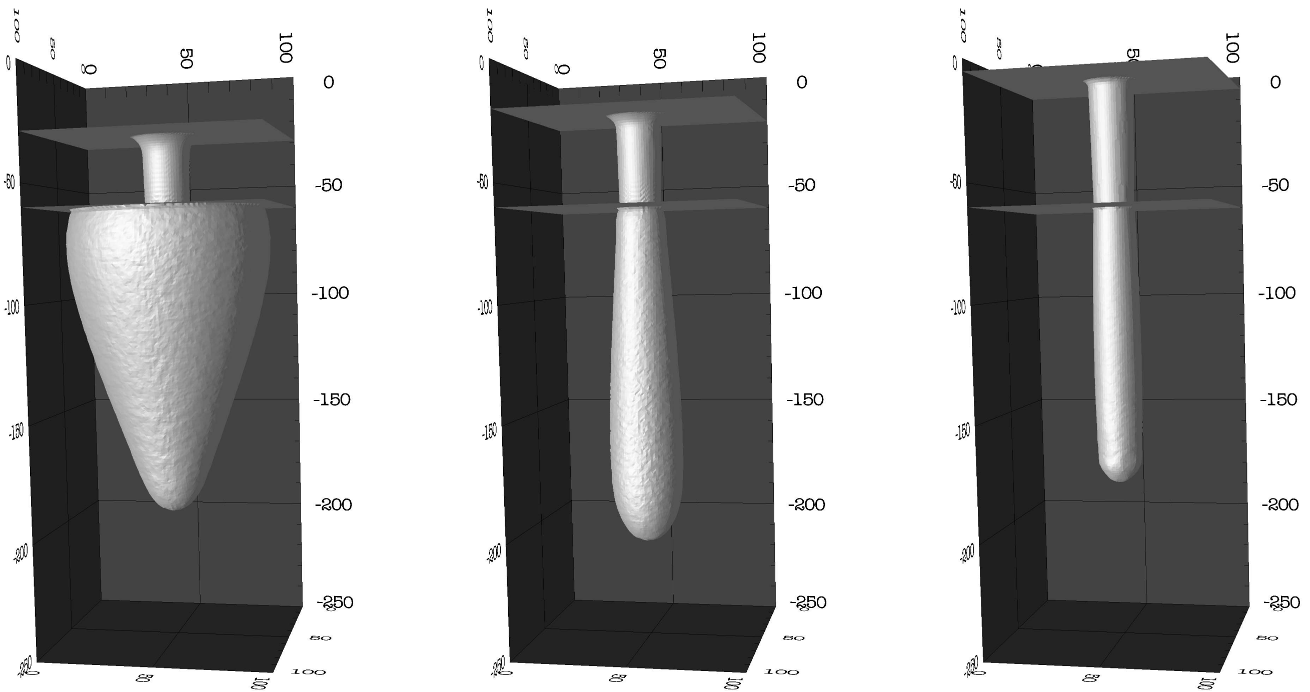

Fig. 3. Reactive ion etching of Si in SF_6/O_2-plasma with increasing amount of oxygen from left (without oxygen) to right, which leads to sidewall passivation and hence to more isotropic etching. Mask etching is also incorporated.

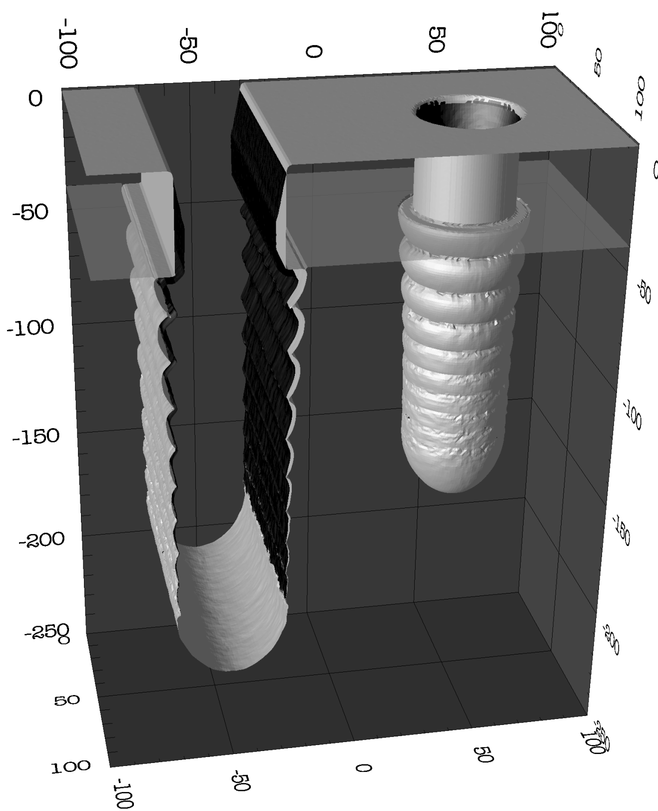

Fig. 4. The profile after 10 cycles of a Bosch process. The polymer surface is colored black. Three different level set functions are used to describe the geometry.

approach for surface velocity calculation supports the incorporation of more complex models, which account for example for specular reflexions or energy dependent effects.

References

[1] U.H. Kwon, W.J. Lee, Thin Solid Films 445/1, pp. 80-89, 2003.
[2] B. Radjenović, J.K. Lee, Proc. 17th ICPIG, Eindhoven, the Netherlands, 17, pp. 142-145, 2005.
[3] D. Kunder, E. Bär, Microelectron. Eng. 85, pp. 992-995, 2008.
[4] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.
[5] R.T. Whitaker, J. Comp. Vision 29/3, pp. 203-231, 1998.
[6] B. Houston, M.B. Nielsen, C. Batty, O. Nilsson, K. Museth, ACM Trans. Graph. 25/1, pp. 151-175, 2006.
[7] O. Ertl, C. Heitzinger, S. Selberherr, Simulation of Semiconductor Processes and Devices 2007, T. Grasser, S. Selberherr (eds.), Springer Wien New York, pp. 417-420, 2007.
[8] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, The Visual Computer 11/8, pp. 429-446, 1995.
[9] Z.-K. Hsiau, E.C. Kan, J.P. McVittie, R.W. Dutton, IEEE Trans. Electron Devices 44/9, pp. 1375-1385, 1997.
[10] T.S. Cale, G.B. Raupp, J. Vac. Sci. Technol. B 8/6, pp. 1242-1248, 1990.
[11] P.L. O'Sullivan, F.H. Baumann, G.H. Gilmer, J. Appl. Phys. 88/7, pp. 4061-4068, 2000.
[12] V. Havran, Heuristic Ray Shooting Algorithms, PhD thesis, Czech. Technical University, Prague, 2001.
[13] OpenMP C and C++ Application Program Interface. Available from http://www.openmp.org.
[14] R.J. Belen, S. Gomez, D. Cooperberg, M. Kiehlbauch, E.S. Aydil, J. Vac. Sci. Technol. A 23/5, pp. 1430-1439, 2005.
[15] G. Kokkoris, A. Tserepi, A.G. Boudouvis, E. Gogolides, J. Vac. Sci. Technol. A 22/4, pp. 1896-1902, 2004.