A Novel Surface Mesh Simplification Method for Flux-Dependent Topography Simulations of Semiconductor Fabrication Processes

Christoph Lenz, Alexander Scharinger, Paul Manstetten, Andreas Hössinger, and Josef Weinbub

Abstract In etching and deposition simulations of a semiconductor fabrication process the calculation of the surface rates of particles is an essential but also the computationally most demanding step. A promising approach is to preprocess the simulation domain by simplifying the surface. We thus propose a new surface mesh simplification method that takes advantage of geometric domain-specific surface properties that are prevalent in topography simulations. We compare our method to a suitable reference algorithm and show that our method maintains higher geometric accuracy and accordingly maintains the original geometry in great detail. Furthermore, the evaluation of the simplified meshes show an enhanced performance of the particle surface rate calculation.

1 Introduction

Process technology computer-aided design (TCAD) tools are used to simulate fabrication processes of semiconductor devices. One important branch of process TCAD is the evolution of the topography during etching and deposition processes. In each time step the three essential computational tasks are: (a) the calculation of the particle flux on the surface, which is used to (b) calculate the surface velocity according to a surface model and (c) the calculation of the new position of the surface using the surface velocities [1]. In Process TCAD the surface can be represented implicitly using the level-set method where the domain is discretized on

C. Lenz (✉) · A. Scharinger · P. Manstetten · J. Weinbub
Christian Doppler Laboratory for High Performance TCAD, Institute for Microelectronics, TU Wien, Wien, Austria
e-mail: lenz@iue.tuwien.ac.at; scharinger@iue.tuwien.ac.at; manstetten@iue.tuwien.ac.at; weinbub@iue.tuwien.ac.at

A. Hössinger
Silvaco Europe Ltd., St. Ives, UK
e-mail: andreas.hoessinger@silvaco.com

M. van Beurden et al. (eds.), *Scientific Computing in Electrical Engineering*, Mathematics in Industry 36, https://doi.org/10.1007/978-3-030-84238-3_8

a regular grid. This approach is attractive due to the robust handling of topographical changes in a level set framework [2]. The particle flux on the semiconductor surface denotes the number of particles interacting on the surface. One possible numerical method for calculating the surface flux is Monte Carlo ray tracing [3]. At practically relevant surface resolutions the flux calculation dominates the overall execution time of an etching or deposition simulation [1]. It is thus useful to investigate approaches that speed up the flux calculation. One promising approach is to use temporary explicit surface meshes as there exists a large body of knowledge about ray tracing on explicit surfaces. The *marching cubes algorithm* [4] is commonly used to extract an explicit surface from the level set. However, the resulting surface meshes typically contain very narrow and long triangles (needles) or small triangles in flat regions that contain no geometric variation. Therefore eliminating those surface elements reduces the total surface element count which speeds up the ray tracing tasks, further underlining the attractiveness of an explicit surface mesh approach.

There exist several algorithms that reduce the resolution of surface meshes with respect to a given metric; several metrics have been proposed in literature [5–8]. However, some of these algorithms try to simplify the geometry homogeneously [5, 6] or use computationally expensive metrics [7, 8]. The latter is particularly relevant when considering the entire etching or deposition workflow where the mesh simplification has to be conducted at every single time step. Mesh simplification, or more general domain simplification, is a commonly used approach in process TCAD simulations [9, 10]. In particular, in [11] the authors evaluate the flux on a mesh by sampling only a sparse set of surface elements to accelerate the simulation.

In this paper we introduce a flexible and computationally lightweight simplification method based on the local surface curvature. We evaluate the impact of our mesh simplification method on typical process TCAD topography simulations by using the high performance ray tracing library Embree [12] by conducting a ray tracing performance analysis. Specifically we compare the flux calculation time for surfaces obtained with the presented method, with the flux calculation time obtained for surfaces generated by the reference Lindstrom-Turk algorithm [5], by comparing the execution time of the simplification process and the performance of the flux calculation using Monte Carlo ray tracing.

2 Surface Mesh Simplification

The simplification method presented in this work is based on the Lindstrom-Turk algorithm [5]. This algorithm uses an *Edge Collapse* procedure to simplify the surface mesh. It offers a relatively low computational complexity and takes the quality of triangles into account: The latter is particularly important for process TCAD simulations, as the mesh quality directly influences subsequent procedures.

Our method uses the mean curvature of each vertex to partition the mesh into regions. This allows us to adjust the amount of simplification according to the local

geometric properties in each region. This simplification method has been designed to simplify regions of the mesh offering negligible geometric variation (e.g. flat areas) to a higher degree, thus allowing to maintain a higher resolution in regions of the mesh with high geometric variation. Furthermore, our method is not limited to the Lindstrom-Turk simplification algorithm, hence other simplification algorithms [6] can be used in combination with our method.

2.1 Feature Detection

The first step in our simplification method is the detection of geometric features in the mesh: We use the absolute mean curvature of each vertex and calculate it via a discrete approximation of the *Laplace-Beltrami operator* [13] in the vertex $\mathbf{x}_i$

$$|H(\mathbf{x}_i)| = \frac{\| \sum_{j \in N_1(i)} (\cot \alpha_{ij} - \cot \beta_{ij})(\mathbf{x}_i - \mathbf{x}_j) \|}{4A_{\text{avg}}}, \tag{1}$$

where $H(\mathbf{x}_i)$ denotes the mean curvature in the vertex $\mathbf{x}_i$ and $N_1(i)$ is the set of all vertices adjacent to $\mathbf{x}_i$. The angles α_{ij}, β_{ij} are the angles of the triangles that share the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$, which are opposite to this edge and A_{avg} is the average area of the triangles surrounding the vertex $\mathbf{x}_i$. The mean curvature is used to categorize each vertex to be either a *flat* or a *feature* vertex. In particular, an empirical threshold is used to identify vertices with small curvature (numerical artifacts), which are considered to be flat.

2.2 Mesh Partitioning and Movement of Regions

The Mesh is partitioned into the *feature regions* and the *transition regions* according to the metrics above. The feature region encompasses the triangles of the mesh with significant geometric variation. The transition region contains the triangles that do not hold information about the geometric variation. This partition of the mesh allows to simplify the transition region to a greater extent, which reduces the overall number of mesh elements without loosing information about the geometric variation. Furthermore, this approach allows to keep a high resolution in regions of the mesh with high geometric variation by simultaneously limiting the overall mesh size in terms of number of triangles. However, simplifying the flat region to a higher degree than the feature region leads to low quality triangles (e.g. needles).

To prevent the formation of low quality elements the transition region is simplified with linearly increasing parameters, thus creating a reasonable mesh grading. Figure 1 schematically depicts two steps of the discussed process. At first the whole mesh, including the feature region, is simplified until the smallest edge has an edge length of l_0. If the feature region should not be simplified l_0 is set

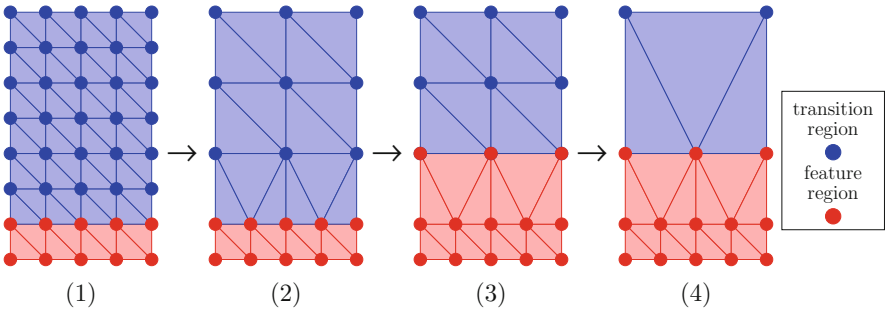


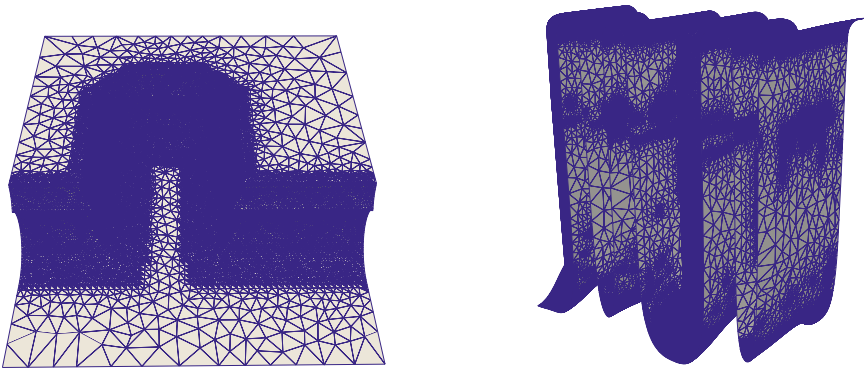
Fig. 1 Example of the simplification process: (1) shows the mesh after it has been divided into regions. (2) shows the simplification of the feature region. (3) shows the extension of the feature region. (4) shows again the simplification of the transition region with an increased edge length

to 0. After this initial simplification step the transition region is simplified until the smallest edge has an edge length of $l_1 = l_0 + sl$, where sl denotes the step length. Next, the feature region is expanded into the transition region. Afterwards the now smaller transition region is simplified until the smallest edge has an edge length of $l_{i+1} = l_i + sl$ with $i \in \{0, 1, \ldots, n \in \mathbb{N}\}$. These last two steps continue until the feature region cannot move any further into the transition region, and thus terminates the simplification process. To avoid unwanted side effects of the potentially large edge lengths produced by our iterative scheme, another parameter $l_{\max}$ is used to terminate the refinement once the edge length l_i in the transition region has reached $l_{\max}$.

The parameter for the simplification of the feature region l_0, when using the level set method, can be connected to the level-set and is chosen in concordance with the minimal grid size Δ_t. When using meshes not originating from a level-set, this parameter can be chosen by averaging the edge length of all feature vertices. We have empirically determined that the step length sl should be approximately the edge length of the feature region after the simplification with the parameter l_0 stops. A bigger step size increases the amount of edges that are removed. However, the bigger the difference between the edge length of the feature region and the step length, the worse the triangle quality of the mesh.

3 Results

The simplification method has been evaluated in the context of process TCAD in three ways: geometric distance to the original geometry, execution time of the simplification method, and the execution time of a subsequent surface flux calculation by ray tracing. In this study two example geometries have been analyzed and each example geometry has been simplified applying eight different degrees of simplification, resulting in a reduction of vertices from 20–90%. Figure 2 shows the



(a) (b)

Fig. 2 Process TCAD surface meshes simplified with our method. (**a**) Surface 1 with 78% of the vertices of the original mesh removed by our simplification method. (**b**) Surface 2 with 52% of the vertices of the original mesh removed by our simplification method

two surface meshes after they have been simplified with our method. The original surface meshes of Surface 1 and Surface 2 have 70,831 and 175,550 vertices, respectively. The performance benchmarks presented in the following are based on a serial C++ implementation of our method executed on a 64bit GNU/Linux platform equipped with an Intel Devil's Canyon CPU.

3.1 Distance to Original Geometry

Surface mesh simplification introduces geometric distortions into the simplified mesh. To measure the error introduced by the simplification process we use the *Hausdorff distance* [14] between the original and the simplified mesh. The Hausdorff distance is measured from each vertex of the original mesh to the simplified mesh. Figure 3 shows the results for one test case of our analysis. The distance to the original mesh is smaller when using our simplification method. On average our simplification method has 20–40% lower Hausdorff distance to the original geometry than using the Lindstrom-Turk algorithm. The reason for the significantly improved Hausdorff distance is our method which allows to use more vertices in areas of high geometric variation, allowing to represent the overall geometry better.

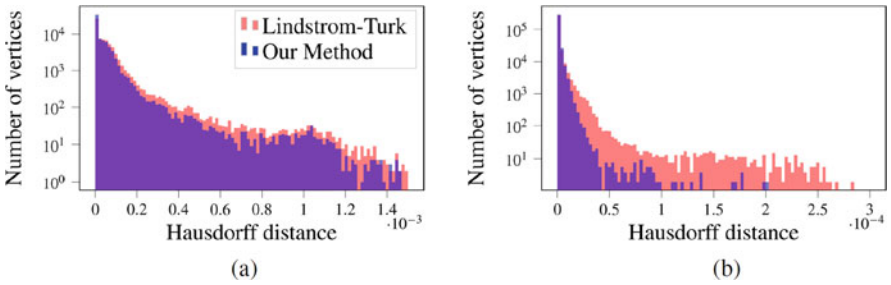


Fig. 3 Hausdorff distance from each vertex of the original mesh to a mesh simplified with our method and a mesh simplified using the Lindstrom-Turk algorithm. (**a**) Surface 1 with 78% of the vertices of the original mesh removed by our simplification method. (**b**) Surface 2 with 52% of the vertices of the original mesh removed by our simplification method

3.2 Time Spent on Simplification

The simplification method presented in this work introduces an overhead to the simplification process. This overhead consists primarily of the feature detection, at the start of the simplification process, and the movement of the feature regions. As can be seen in Fig. 4 our simplification method takes on average 17% longer than the Lindstrom-Turk algorithm.

3.3 Flux Calculation and Monte Carlo Ray Tracing

A common approach to compute the surface flux in a Process TCAD application is to use a Monte Carlo simulation [15]. This is a randomized procedure and the results of the Monte Carlo method are of stochastic nature. To compute the trajectories

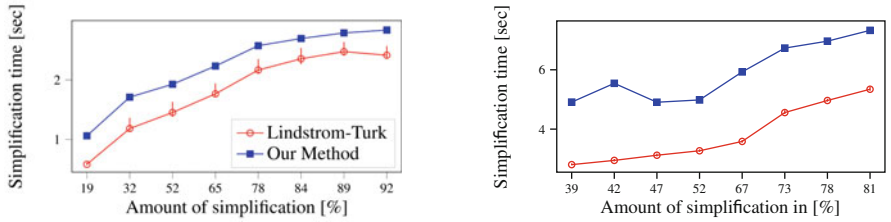


Fig. 4 Average simplification time of our method and the Lindstrom-Turk algorithm. The amount of simplification denotes the number of vertices which have been removed from the original mesh. (**a**) Surface 1. (**b**) Surface 2

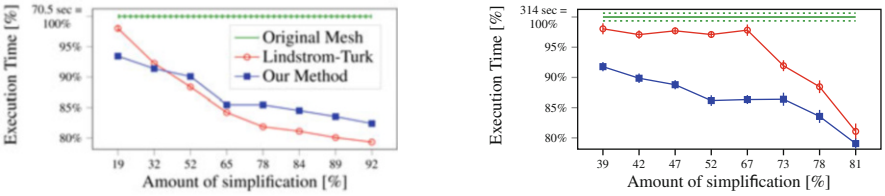


Fig. 5 Execution time of Monte Carlo ray tracing using 10^8 rays. The amount of simplification denotes the number of vertices which have been removed from the original mesh. (**a**) Surface 1. (**b**) Surface 2

on which particles move through the simulation domain (modeling the surface flux) we use the Embree ray tracing library [12]. In Embree a bounding volume hierarchy data structure [3, 12] is used to efficiently compute the paths on which the particles move through space. The internal structure of the bounding volume hierarchy depends eminently on the structure and the coarseness of the surface mesh.

Figure 5 shows the execution times measured to perform the Monte Carlo ray tracing on the meshes with different degrees of simplification. As the simplified meshes contain less triangles the bounding volume hierarchy data structure used for ray tracing will have less elements than the data structure for the original mesh. As the size of the data structure is decreased the memory footprint is reduced and this leads to faster flux calculations because less data has to be processed and the caches of the processor are used more effectively. Figure 5a and b show that the empirical speedup in flux calculation depends on the shape of the surface mesh. When tracing Surface 1, the meshes of both simplification methods perform approximately the same and are faster than the original mesh. When tracing Surface 2, the meshes generated by our simplification method clearly outperform the meshes simplified with the Lindstrom-Turk algorithm and the original geometry. Surface 2 contains deep trenches and the rays of the tracing algorithm need to travel towards the bottom of these trenches. As the walls of the trenches do not have high curvature the bounding volume hierarchy data structure created from the mesh simplified with our method will be less complex within the deep trenches and hence, the traces of the rays down the trench can be computed by performing less operations. Also, the rays which travel towards the bottom of the trench usually reflect off the surface many times, which makes the difference in computational effort for using a bounding volume hierarchy from a mesh simplified with our method even more evident. Figure 5b for Surface 2 shows a speedup of about 12% compared to the Lindstrom-Turk algorithm for simplification levels of 52 and 67%.

4 Summary

We introduce a new surface mesh simplification method that uses the curvature of the surface mesh to identify regions which can be simplified with different sets of parameters depending on the local surface properties. Our approach is well suited for meshes that are common in flux-dependent process TCAD simulations since such meshes often contain large flat regions with high resolutions from the originating regular grid. We have evaluated our method with respect to geometric distances and execution times for simplification and subsequent computations of flux estimates. The geometric distances in the experiments have improved in comparison to the reference algorithm. In particular, the average Hausdorff distance of the investigated geometries has improved by 20–40%. The ray tracing time in all our experiments has been improved on average by 15%, furthermore, demanding real world geometries from process TCAD have shown a compelling improvement of 12% of time spent on ray tracing. The execution time of our simplification method is on average 17% slower than the reference algorithm. When considering entire topography simulations, the accelerated ray tracing significantly exceeds the additional time spent on our simplification method.

Acknowledgments The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

1. P. Manstetten, Efficient Flux Calculations for Topography Simulation. Doctoral Dissertation, TU Wien (2018). http://www.iue.tuwien.ac.at/phd/manstetten/
2. J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science* (Cambridge University Press, 1999)
3. A.S. Glassner, *An Introduction to Ray Tracing* (Elsevier, 1989)
4. W.E. Lorensen, H.E. Cline Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Comput. Graph. **21**, 163–169 (1987)
5. P. Lindstrom, G. Turk, Fast and memory efficient polygonal simplification, in *Proceedings of the IEEE Visualization Conference*, pp. 279–286 (1998)
6. M. Garland, P.S. Heckbert, Surface simplification using quadric error metrics, in *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pp. 209–216 (1997)
7. H. Borouchaki, P.J. Frey, Simplification of surface mesh using Hausdorff envelope. Comput. Methods Appl. Mech. Eng. **194**, 4864–4884 (2005)
8. S.J. Kim, C.H. Kim, D. Levin, Surface simplification using a discrete curvature norm. Comput. Graph. **26**, 657–663 (2002)
9. F. Rudolf, Symmetry- and Similarity-Aware Volumetric Meshing. Doctoral Dissertation, TU Wien (2016). https://www.iue.tuwien.ac.at/phd/rudolf/
10. L. Gnam, High Performance Mesh Adaptation for Technology Computer-Aided Design. Doctoral Dissertation, TU Wien (2019). https://www.iue.tuwien.ac.at/phd/gnam/

11. L. Gnam, P. Manstetten, A. Hössinger, S. Selberherr, J. Weinbub, Accelerating flux calculations using sparse sampling. Micromachines **9**, 1–17 (2018)
12. I. Wald, S. Woop, C. Benthin, G.S. Johnson, M. Ernst, Embree: A kernel framework for efficient CPU ray tracing. ACM Trans. Graph. **33**, 143:1–143:8 (2014)
13. M. Meyer, M. Desbrun, P. Schröder, A.H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in *Visualization and Mathematics III*, pp. 35–57 (2003)
14. R. Straub, Exact computation of the Hausdorff distance between triangular meshes, in *Proceedings of the Conference of the European Association for Computer Graphics*, pp. 17–20 (2007)
15. R.Y. Rubinstein, D.P. Kroese, *Simulation and the Monte Carlo Method* (Wiley, 2016)