A Multi-Mode Mesh Generation Approach for Scientific Computing

F. Stimpfl, R. Heinzl, P. Schwaha, and S. Selberherr

Institute for Microelectronics, TU Wien
Gußhausstraße 27-29/E360, A-1040 Vienna, Austria
E-mail: {stimpfl|heinzl|schwaha|selberherr}@iue.tuwien.ac.at

KEYWORDS

Mesh generation, TCAD, Delaunay, Advancing front, Scientific Computing

ABSTRACT

With the steady evolution of software tools and programming languages, several programming paradigms have become available and new methodologies related to library centric application design have been developed. In addition to this paradigm shift, the increase of computing power requires an adaptation of well known techniques for parallel computing. Various problems of different disciplines appear, for which no efficient and definite solutions exist. This calls for new paradigms to be established and the development of new ways which fully exhaust the combination of sophisticated software and powerful hardware.

The area of spatial discretization, unstructured mesh generation, is one of these areas, which still exhibits various issues from fields such as modular software design, high performance, and most importantly robustness. In this work problems for mesh generation are identified and solutions explored, which anticipate the given challenges. This approach is combined in a software library based on a generic scientific simulation environment and compared to existing tools.

INTRODUCTION AND MOTIVATION

The field of scientific computing as a whole, and Technology Computer Aided Design (TCAD) in particular, relies on either domain-specific closed applications, or components tied very closely to specific representations which were often built for very fixed purposes and then wrapped into a large and monolithic toolkit. To this date, data structures and algorithms are implemented in a heavily application specific way, making their reuse practically impossible.

One of the most fundamental issues in TCAD, spatial discretization, requires certain and diverse constraints during mesh generation. Many software tools exist which solve very narrow sets of problems in special and well defined settings, but unfortunately most of the solutions to these problems are implemented repeatedly.

With the steady evolution of software engineering techniques and methodologies, new paradigms have emerged which offer great opportunities related to orthogonal module design and high performance approaches. Using a generic programming approach instead of object orientation, data structures and algorithms can be implemented orthogonally, where some languages, e.g. C++, offer additional high performance possibilities as well. In addition to these basic software engineering issues, the calculating power of modern computer systems, which has been doubling every 18 months in the past, and the emerging CPU parallelism, require different programming paradigms and application design methodologies.

These changes have to be taken into account as well as the upcoming shift to fully three-dimensional simulations [1]. This of course also implies, that three-dimensional mesh generation is getting more and more important in the next decade, especially in the field of TCAD where up to now, no comprehensive mesh generation application has emerged.

Mesh generation is a quite difficult task to be implemented on computers, because a rigoros algorithmic formulation does not exist. The emerging paradigm of computational topology will support mesh generation in the coming years [4, 17].

Different fields of TCAD application impose a variety of different constraints and requirements on mesh generation, e.g. topography simulation requires a good approximation of surface elements, while ion implantation simulation requires high density near the surface, according to the gradient of the ion distribution. Diffusion simulations add a need for a fine mesh at interfaces in addition to a high mesh density near the surface. The complex field of device modeling even requires a completely different type of mesh, necessitating a remeshing step for the whole input structure. In summary, it can be observed that each simulation step has completely different requirements on the underlying spatial discretization. Therefore, meshing is still one of the major showstoppers in the field of TCAD.

In this work we present a comprehensive modular, library centered approach which focuses on the problems described above. Already existing concepts and libraries in the field of scientific computing, which have proven to be successful, are combined on a new software basis.

PROBLEMS WITH MESH GENERATION

Before introducing different techniques of mesh generation, common problems are sketched which impose various constraints, restrictions, and difficulties for the task of mesh generation in general.

- **Different element size:** The generated mesh can consist of elements of different size, meaning, that adjacent elements vary greatly in size, e.g. a thin layer of oxide within a three-dimensional device, which is more than three order of magnitudes different in size.
- **Boundary requirements:** Various boundary requirements cause problems for mesh generation, e.g. a tetrahedra configuration known as Schönhart polyhedra [13] or the inclusion of preset surface edges, which imposes additional difficulties, if the surface requires Delaunay conformity [6].
- **Cospherical point sets:** Points are said to be cospherical, if at least n + 2 point are located on the perimeter of an n-dimensional sphere S, where S does not contain any other points of the mesh. This problem only arises, when the Delaunay criterion has to be met.
- **Degenerated cells:** Degenerated cells may result due to several reasons, such as the requirement to adhere to a given boundary or that the generated mesh has to include a given point cloud. The problem with these degenerated cells is that they may be unsuitable for the subsequent simulation tasks, e.g. interpolation of physical quantities. This problem increases further, as degenerated cells cannot be refined without adding additional degeneration. Therefore several classes of degeneration are classified and various mesh adaptation techniques are used to minimize the number of these elements.

As already pointed out, the application design for mesh generation not only has to deal with the given difficulties based on mesh generation itself, but it also requires various software engineering methodologies to develop robust applications.

MESHING METHODS

There are several methods for mesh generation, which are currently in use for scientific computing, such as:

- Giftwrapping
- Incremental Delaunay Triangulation
- Advancing Front
- Advancing Front with Delaunay Triangulation

To present the advantages and disadvantages of these different approaches, a brief overview of each is given in the following:

Giftwrapping

Starting with any element, a sphere is expanded around the element until the next point is found, where the extension of the sphere is controlled by a given direction vector. Figure 1 illustrates the idea behind the giftwrapping algorithm, which is suitable for use in multiple dimensions. Only convex hulls can be created using the giftwrapping algorithm and cospherical point sets impose additional difficulties.

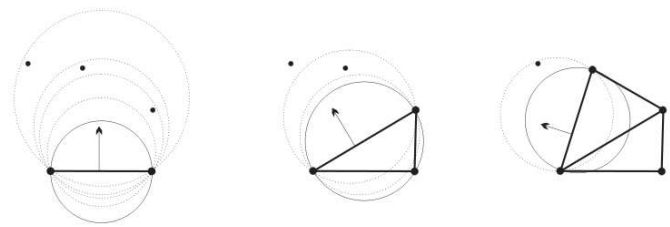

Figure 1: Giftwrapping example

Incremental Delaunay Triangulation

Many triangulations exist for a set of points in a plane. A triangulation which meets the so called circumcircle or Delaunay property is called a Delaunay triangulation. This property states that the circumcircles of the triangles may contain no points beside the points forming the triangles (see Figure 2).

One of the main advantages of the Delaunay triangulation in two dimensions is the minimization of the maximum angles and the maximization of the minimum angles. In three dimensions this advantages cannot be easily accomplished due to non-unique flip constraints [14].

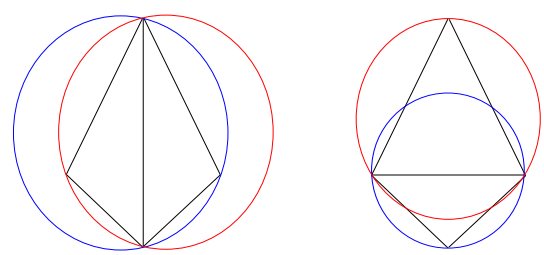

Figure 2: Delaunay property

An additional difficulty for this method is the incorporation of boundaries due to the fact that incremental Delaunay triangulation uses the convex hull of the input point set only. Given boundary representations have to be marked and reconstructed after the meshing process has completed.

Figure 2 shows two examples of a simple mesh. The left part violates the Delaunay criterion, because the cirumcircles include the points of both triangles. The right part is a correct Delaunay triangulation, because

the circumcircles of both triangles contain only their respective three points.

Advancing Front

The advancing front algorithm starts with a set of boundary elements of a given dimension, e.g. edges. These edges form the initial front which is advanced into the simulation domain. An edge of this set is chosen to form a new triangle, either with an existing point or a newly created point. The current edge is then removed from the front and the two new edges are, depending on their visibility, added to the front. This process terminates when no edges remain within the front.
Figure 3 shows a simple advancing front triangulation. The figure sketches the advancing front mesh generation concept in two dimensions. The dotted line represents the current front. New triangles are inserted, by joining the two ends of a front edge to either a newly created point or an existing point, one at a time [10].

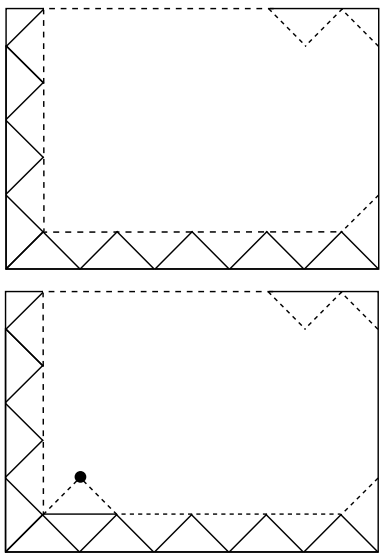

Figure 3: Advancing Front [10]

In contrast to the Delaunay triangulation, the advancing front method, easily incorporates the boundary. It starts from the given boundary representation and advances the front into the simulation domain. A major drawback of this method is that the quality of the generated elements heavily depends on the quality of the boundary elements. For different implementations of this type of mesh generation technique, the robustness issues are severe. The advantages of this method are the good control mechanism for the element sizes, the quality of the generated elements, and the not required Delaunay property which can optionally be incorporated easily. Parallelization can be quite challenging for surface creation, because new surface elements depend on the previously generated surface element. If the surface is partitioned, volume mesh generation, on the other hand, can be parallelized in order to distribute the generation process on several computers. Using the advancing front algorithm, it is no problem to divide the simulation domain into smaller domains and treat each domain on its own. However, boundary information, e.g. different materials, has to be available, otherwise the problem is similar as with surface mesh generation. A method of incorporating the Delaunay property into the advancing front method has already been proposed [10], which adds new points ahead of the front and triangulates them according to the Delaunay criterion. This algorithm works like the conventional advancing front algorithm, only point insertion is extended to satisfy the Delaunay criterion.
Figure 4 shows the idea behind the algorithm. When inserting a new point, there are two possibilities:

- either the point is not inside any existing triangle circumcircle,
- or there exists at least one triangle, whose circumcircle contains this new point.

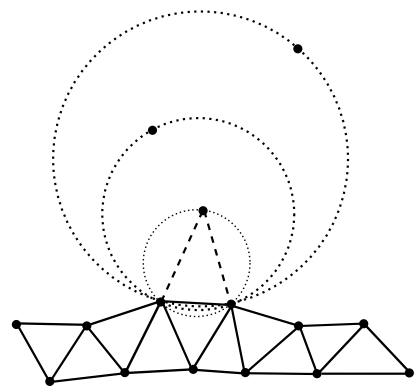

Figure 4: Advancing Front Delaunay Triangulation [10]

A different implementation is available using an abstract rule set [12], which describes different methods of controlling the advancing front. Basically, a rule describes where points are to be generated. The rule also manages changes in the advancing front. Some boundary elements have to be added while others have to be deleted. If no matching rule is found, the quality class is decreased. This allows the application of a rule of a lower quality level, which may also delete the current element.

SOFTWARE DESIGN

The meshing methods, described in the previous section, have been included as modules in the Generic Scientific Simulation Environment (GSSE [8]). The software design of our meshing library can be seen in Figure 5. The following presents our solution of the previously mentioned meshing problems.

- **Different element size**
  To cope with the problem of different element sizes we

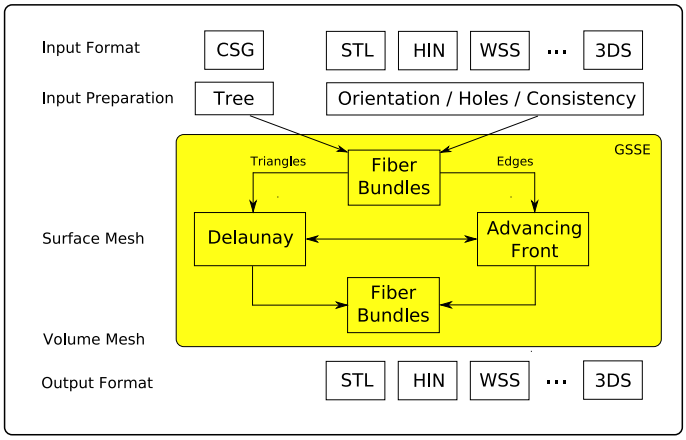

Figure 5: Design of GSSE

use the concept of a local feature size (lfs) [11]. The lfs of a point p is defined as the radius of the smallest disk centered at p, which intersects two non-incident vertices or segments of the planar straight line graph. The lfs guarantees that there are no abrupt changes in the size of the elements, which is especially important with the advancing front algorithm.

- **Boundary requirements**
  The application of the advancing front algorithm based on the introduced abstract rule mechanism easily incorporates the given boundary elements. Additionally, the Delaunay criterion can be used to create conforming boundary representations.
- **Cospherical point sets**
  This problem arises, when a mesh is used for, e.g. the finite volume method. The Voronoi graph [14] is therefore used to model flux conservation. Due to the fact that a Voronoi graph is the dual graph of a Delaunay graph, the input mesh has to exhibit the Delaunay property. Our approach optionally guarantees Delaunay conformity for both the surface and the volume.
- **Degenerated simplices**
  The use of the advancing front algorithm in combination with the already introduced quality rules and the corresponding point placement strategies reduces the degeneration of elements. The existing rules were extended to meet the Delaunay criterion.

Related to software engineering issues, the proposed solution uses a library centric application design paradigm [8] based on a multi-paradigm approach.

To incorporate various input formats and to solve the problems with different data models, the proposed approach uses a fiber bundle data model. Fiber bundles were first introduced by Butler and Pendley [3] and afterwards enhanced [2] to include cell complex properties. The basic approach of this data model is the separation of cells into a base space and the connectivity information storage within a fiber space.

All current data models can be mapped to the fiber bundle data model. This gives the advantage to read and write various file formats. Due to the modular design it is also possible to quickly write new input or output modules to accommodate an even wider range of file formats.

An advantage of our approach is, that the developed modules can be combined in various ways, depending on the requirements of the subsequent simulation. For example a mesh, initially generated using the advancing front algorithm, can be refined afterwards at certain points, which are interesting for the simulation, or the mesh can be modified to meet the Delaunay criterion. The module design can also be seen in Figure 5.

Another advantage of the modularized development approach is the possibility to interface different mesh engines and thereby fully utilize the orthogonal application approach.

Our library also contains a resampling step which consists of several substeps [12]. First the points are extracted from the input model. This extraction only applies to the 'points of intersection', which are the points where at least three surfaces intersect. To solve this problem, a bisection algorithm based on geometric tests is used. Afterwards, using this minimal mesh, a new surface is generated, with an advancing front algorithm [12].

MESHING LIBRARIES

As it is our goal to focus on library centric design, various meshing libraries are incorporated and therefore briefly presented here.

TetGen: is developed by Hang Si, from the Weierstrass Institute for Applied Analysis and Stochastics, and generates a Delaunay triangulation. Additionally, it supports the creation of Voronoi diagrams and convex hulls for three-dimensional point sets [5, 15, 16].

Netgen: uses the described abstract rules for advancing front mesh generation and offers a Delaunay point cloud generation module [12]. This module lacks the modeling of a guaranteed Delaunay volume mesh as well as the conforming Delaunay property for surfaces.

deLink: was developed for TCAD applications based on an advancing front method combined with a Delaunay method which does not create new points automatically [7]. Therefore, additional point clouds created separately have to be used to refine structures. The created meshes satisfy the Delaunay criterion for surfaces and volumes.

OUR APPROACH

It is important to highlight that TetGen and Netgen cannot deal with corrupted input representations, e.g. holes. deLink and GSSE use various methods to correct the input representations, where our approach can

also use a separate surface remeshing step. GSSE also incorporates constructive solid modeling [8]. Table 1 presents an overview of the features of each mesh generation module.

Features	TetGen	Netgen	deLink	GSSE
Delaunay	+	x	+	+
Advancing Front	-	+	-	+
Remesh Steps	-	+	-	+
Automatic Points	-	+	-	+
CSG Input	-	+	-	+
Polygon Input	-	-	+	+
Repair Surface	-	-	+	+
Multi Material	-	-	+	+

Table 1: Results of the comparison

EXAMPLES

This section presents some examples of meshes, created by the four previously outlined tools. The examples are analyzed and the quality distribution is presented. To compare the meshing tools, we categorize the elements of the mesh using two main parts [9]. First four classes of quality types, defined by the number of small dihedral angles, as shown in Figure 6 are set up.

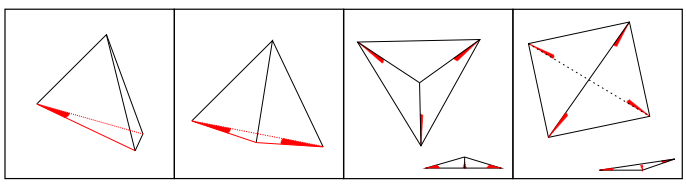

Figure 6: Four different classes of degenerated tetrahedra (wedge, spade, cap, sliver), sorted by the number of acute dihedral angles.

Then, the tetrahedra are classified by the number of degenerated triangles, like daggers and blades. The dagger has one short edge and at least one small angle, where the blade has no short edge and therefore one large and two small angles. Figure 7 shows the needle (or spire) with three daggers (the short edges are marked in the figure), the slat (or splinter) with two opposite short edges and, therefore, four daggers. Finally, the spindle does not have short edges and has therefore four blades as triangles.
The Figures 8 and 10 show the basic functionality and the boundary representation of the different meshing kernels. The Figures 9 and 11 summarize the respecitve statistics. The pictures in Figure 12 and 13 present examples from TCAD / topography simulation.

CONCLUSION

We have identified several problems of mesh generation and presented our solution to these problems. A meshing library based on GSSE has been developed to collect these solutions. Finally, a comparison between GSSE and established meshing kernels was performed showing, that GSSE has more mesh generation possibilities than the other investigated tools. The analysis of the examples also shows that GSSE produces an output of at least the same performance and even better quality than the other tools.

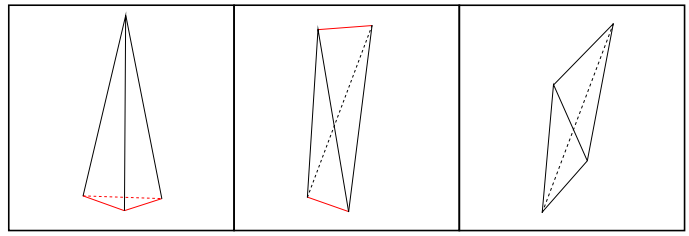

Figure 7: Three different types of degenerated tetrahedra (needle, slat, spindle).

Acknowledgment

This work has been supported by the Intel Corporation and the Austrian Science Fund FWF, project P19532-N13.

REFERENCES

[1] International Technology Roadmap for Semiconductors, 2005.

[2] W. Benger. *Visualization of General Relativistic Tensor Fields via a Fiber Bundle Data Model*. Dissertation, Freie Universität Berlin, 2004.

[3] D. Butler and M. Pendley. A Visualization Model Based on the Mathematics of Fiber Bundles. *Computers in Physics*, 3(5):45–51, September/October 1989.

[4] H. Edelsbrunner and D. Guoy. An Experimental Study of Sliver Exudation, 2001.

[5] H. Edelsbrunner and N. R. Shah. Incremental Topological Flipping Works for Regular Triangulations. *Algorithmica*, (15):223–241, 1996.

[6] P. Fleischmann. *Mesh Generation for Technology CAD in Three Dimensions*. Dissertation, Technische Universität Wien, 2000.

[7] P. Fleischmann and S. Selberherr. Enhanced Advancing Front Delaunay Meshing in TCAD. In *Proc. Conf. on Sim. of Semiconductor Processes and Devices*, pages 99–102, 2002.

[8] R. Heinzl. *Concepts for Scientific Computing*. Dissertation, Technische Universität Wien, 2007.

[9] R. Heinzl and T. Grasser. Generalized Comprehensive Approach for Robust Three-Dimensional Mesh Generation for TCAD. In *Proc. Conf. on Sim. of Semiconductor Processes and Devices*, pages 211–214, Tokio, September 2005.

[10] D. Mavriplis. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *J. Comput. Phys.*, 117(1):90–101, 1995.

[11] J. Ruppert. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *J. of Algorithms*, 18(3):548–585, 1995.

[12] J. Schöberl. NETGEN - An Advancing Front 2D/3D-Mesh Generator Based on Abstract Rules. *Comput. Visual. Sci.*, 1:41–52, 1997.

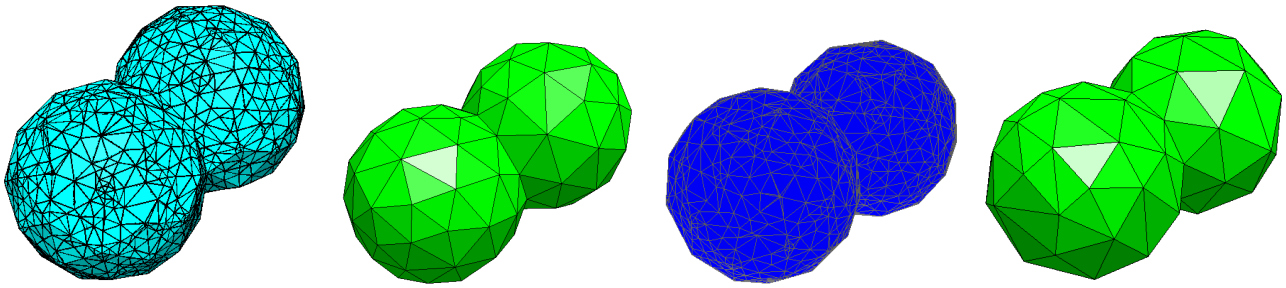

Figure 8: A simple sphere, but yet a complex task for a mesh generator, is used to highlight the different approaches related to the investigated types of meshing algorithms, from left to right: TetGen, Netgen, deLink, GSSE.

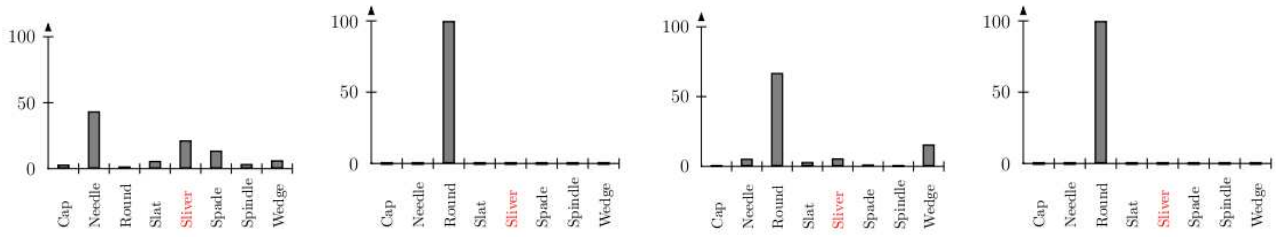

Figure 9: Statistics for the meshing example in Figure 8, from left to right: TetGen, Netgen, deLink, GSSE.

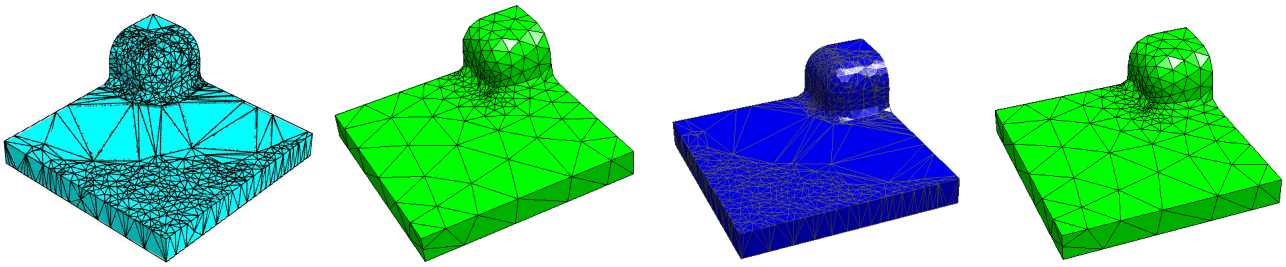

Figure 10: An example from topography simulation is used as example to illustrate the different types of meshing results, from left to right: TetGen, Netgen, deLink, GSSE.

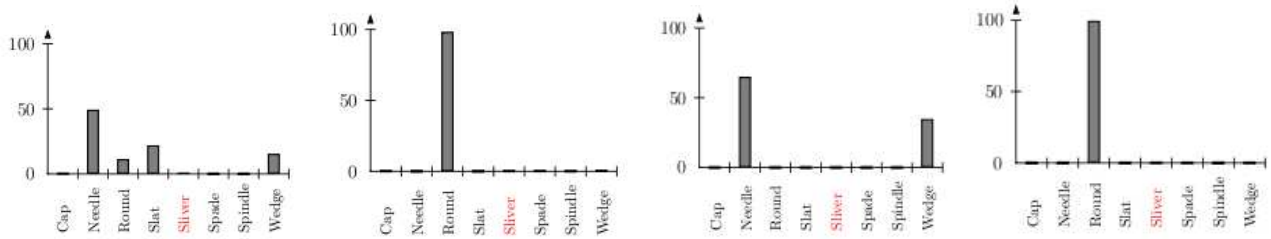

Figure 11: Statistics for the meshing example in Figure 10, from left to right: TetGen, Netgen, deLink, GSSE.

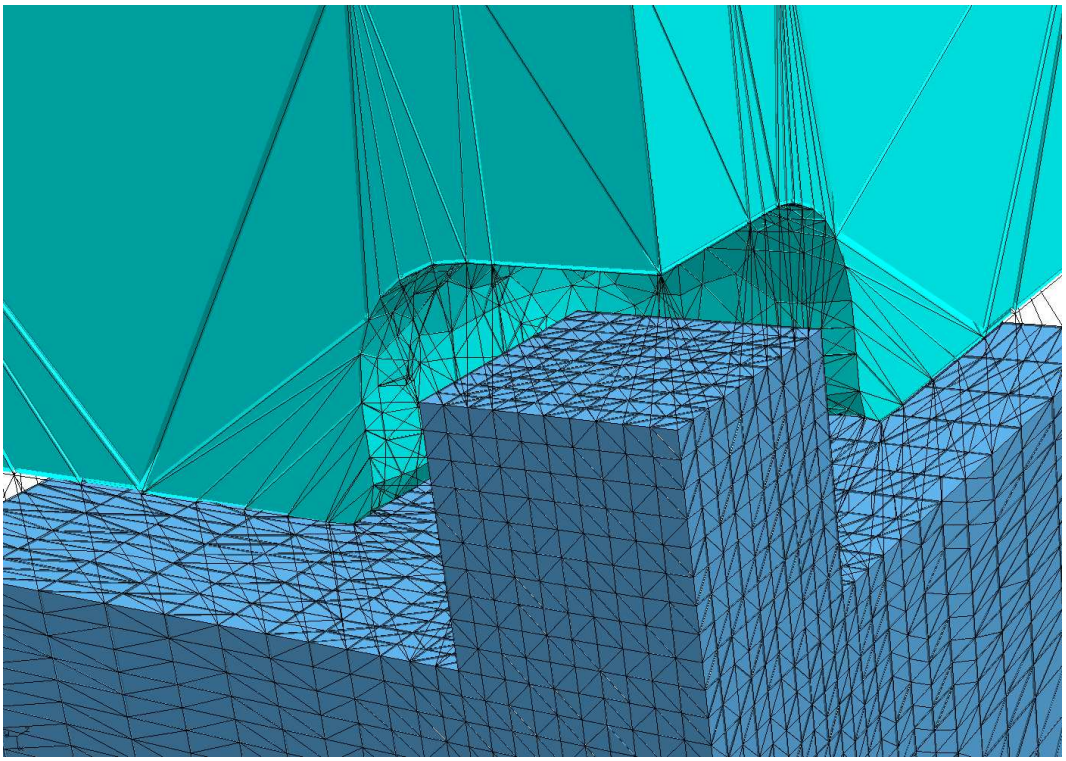

Figure 12: The complete example from topography simulation with the additional constraint of including a given point cloud as can be seen in the bottom part of the structure. The middle part is made transparent to expose the surface mesh. The top part is not required for subsequent simulation and is therefore meshed without quality constraints.

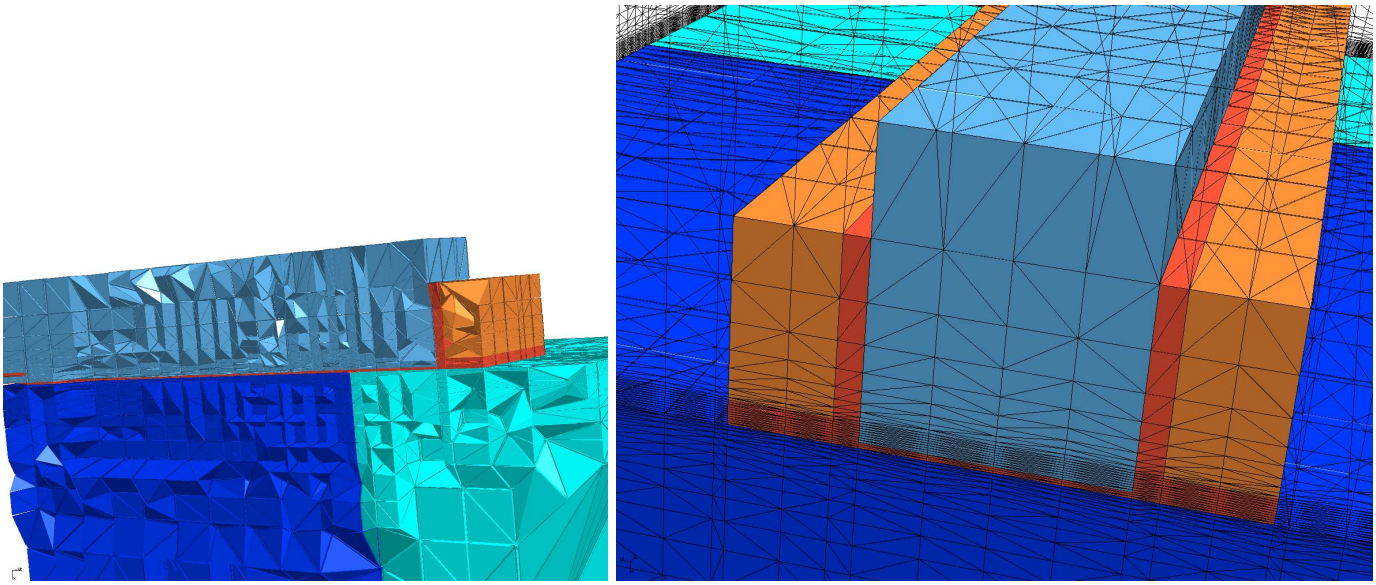

Figure 13: A three-dimensional device structure for a MOSFET with an additional externally supplied point cloud. The important part is the regularity of the elements in the channel region. The different aspect ratios, e.g. the thin red oxide part and the large blue silicon part, are also an additional complication for the mesh generation algorithm.

[13] E. Schönhart. Über die Zerlegung von Dreieckspolyedern in Tetraeder. In *Journal Mathematische Annalen*, March 1928.

[14] J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997.

[15] H. Si. On Refinement of Constrained Delaunay Tetrahedralizations. *15th Intl. Meshing Roundtable*, September 2006.

[16] H. Si and K. Gaertner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. *14th Intl. Meshing Roundtable*, pages 147–163, September 2005.

[17] A. J. Zomorodian. Topology for Computing. In *Cambridge Monographs on Applied and Computational Mathematics*, 2005.

BIOGRAPHIES

FRANZ STIMPFL was born in Vienna, Austria, in 1980. He studied computer science at the Technical University Vienna. He joined the Institut für Mikroelektronik in September 2007, where he is currently working on his doctoral degree.

RENÉ HEINZL studied electrical engineering at the Technical University Vienna. He joined the Institute for Microelectronics in November 2003, where he finished his doctoral degree in September 2007. In April 2005 he achieved first place at the doctoral competition at the EEICT in Brno. His research interests include computational science, programming paradigms, high performance programming techniques, solid modeling, scientific visualization, and algebraic topology for TCAD.

PHILIPP SCHWAHA studied electrical engineering at the Technical University Vienna. He joined the Institute for Microelectronics in June 2004, where he is currently working on his doctoral degree. His research activities include circuit and device simulation, device modeling, and software development.

SIEGFRIED SELBERHERR received the Ph.D. degree in technical sciences from the Technical University Vienna in 1981. Since that time he has been with the Technical University Vienna as professor. Dr. Selberherr has been holding the "venia docendi" on "Computer-Aided Design" since 1984. As of 1988 he has been chair professor of the Institut für Mikroelektronik. From 1998 to 2005 he served as Dean of the "Fakultät für Elektrotechnik und Informationstechnik" at the Technical University Vienna. His current topics of interest are modeling and simulation of problems for microelectronics engineering.