P1

High-Quality Mesh Generation Based on Orthogonal Software Modules

Josef Weinbub*, Johann Cervenka*, Karl Rupp*†, and Siegfried Selberherr*
*Institute for Microelectronics, TU Wien, Vienna, Austria
† Institute for Analysis and Scientific Computing, TU Wien, Vienna, Austria
Email: {weinbub,cervenka,rupp,selberherr}@iue.tuwien.ac.at

***Abstract*—To provide simulation software in the field of TCAD with the utmost flexibility regarding generation and adaptation of meshes, a generic and high-quality meshing library, ViennaMesh, has been developed. The library is coded in C++ and utilizes modern programming techniques to wrap tasks, like mesh generation and mesh adaptation, into functional objects, which can then be concatenated to form the desired meshing process. Additionally, a meta-selection environment provides the ability to select a mesh generation kernel based on properties already defined at compile time. Code examples are depicted and briefly discussed. Moreover, several enhancements to existing mesh adaptation methods have been made, which are demonstrated based on meshes provided by industrial partners.**

I. Introduction

The transition from real world devices to the computer domain requires a discretization, meaning that a finite set of information, like geometry and topology data, approximately represents the actual device [1]. This fundamental mapping step is typically realized by meshing tools. Several research areas deal with various aspects of meshes, most importantly generation techniques, but also adaptation and classification techniques [2] [3]. Several tools are publicly available offering different approaches and properties [4] [5] [6] [7]. The different applied approaches offer advantages and disadvantages. For example, the advancing front based algorithms offer good control mechanisms for generated mesh element size and quality, but on the contrary rely heavily on the quality of the input boundary mesh and the colliding fronts.

Moreover, high-quality mesh generation for Technology Computer Aided-Design (TCAD) is a demanding field due to the challenging input geometries consisting of thin layers and complex surfaces [8]. Therefore, we focus on unstructured meshing tools based on simplicial mesh element topologies, for example, triangles and tetrahedrons. This type of mesh supports local mesh adaptation by simultaneously keeping the global number of mesh elements to a minimum [2]. For TCAD applications this capability is highly important not only due to the complex input geometries, but also due to the local regions of interest within simulation domains, for example, the conducting channel of a MOSFET. Those regions typically require locally a denser mesh, as otherwise the numerical error would be too large or numerical oscillations occur, which may ultimately render the simulation results unfeasible.

In addition to the challenges posed by the input geometry, the discretization error of most discretization schemes strongly depends on the underlying mesh due to the error which degenerated elements are likely to introduce [9]. Consequently, it is essential for simulation software in the field of TCAD to have access to high-quality mesh generation kernels.

Additionally, simulation applications have to be granted access to the various mesh related functionalities of the meshing tools to control, for example, the mesh adaptation process for the simulation at hand. This is typically realized by utilizing the Application Programming Interfaces (APIs) of the respective meshing tool. This utilization is usually feasible for a small number of meshing libraries, but it is not applicable when the number of interfacing tools increases. Each library provides its own specific API and therefore requires special treatment, which results in code duplication and ultimately in programming overhead. This fact introduces the need for an additional generic meshing layer which provides a unified interface to the multitude of meshing tools.

To tackle the introduced challenges, ViennaMesh [10] has been developed. The library not only provides a unified API access to various mesh related tools but also an improved hull mesh adaptation algorithm. The adapted hull mesh suits excellently for volume mesh generator kernels based on the advancing front [11] algorithm. The essence of this particular algorithm is to start from an initial set of boundary elements, which forms the so-called front, and then advance into the simulation domain by attaching mesh elements to the front. Consequently, if the front consists of high-quality elements, the probability of generated high-quality mesh elements is increased significantly.

This work is organized as follows. Section 3 introduces the generic meshing library ViennaMesh. Section 4 discusses the improvements of our hull mesh adaptation algorithm. Volume mesh results based on geometries provided by industry are depicted in Section 5.

978-1-61284-418-3/11/$26.00 © 2011 IEEE

II. A Generic Meshing Layer

ViennaMesh has been implemented to provide a unified access to various meshing related algorithms with special focus on the demands of modular device simulation software. Among these demands are extendibility and orthogonality of the individual software components. Extendibility refers to the ability to conveniently add additional mesh generation kernels as well as mesh adaptation and classification algorithms. This is especially of interest, as there is no unique approach to adapt meshes as well as to classify them. Mesh classification enables judgment of whether or not a mesh is of appropriate quality. Orthogonality of the software components allows to combine various meshing routines to setup specialized mesh generation process chains. By utilizing these possibilities the functionality as well as the robustness of ViennaMesh has been significantly improved. Although our major focus is on volume mesh generation, hull generation and adaptation are mandatory preprocessing tasks to prepare the input geometries for the volume generation step. Typically, three different major meshing related tasks can be identified: generation, adaptation, and classification.

To achieve a decoupling of the multitude of meshing tasks, modern programming techniques in C++ are applied, being the functional, meta, and generic programming paradigms.

The functional approach is used to model a dynamic approach of defining a meshing task. Different meshing tools can be arbitrarily concatenated to form a specific meshing process. Therefore, the input data which contains the input geometry enters the beginning of the meshing process chain and is forwarded from one module to the other. In the end the result can be extracted, as depicted in the following.

```
1 bnd_reader_type            bnd_reader("filename.bnd");
2 typedef wrapper<bnd, bnd_reader>     bnd_wrapper_type;
3 bnd_wrapper_type                     data(bnd_reader);
4 typedef mesh_generator<cervpt>::type hull_mesher_type;
5 hull_mesher_type                     hull_mesher;
6 typedef mesh_adaptor<orienter>::type orient_adapter_type;
7 orient_adapter_type                  orienter;
8 typedef mesh_adaptor<quality>::type  quality_adapter_type;
9 quality_adapter_type                 quality;
10 typedef mesh_generator<netgen>::type volume_mesher_type;
11 volume_mesher_type                  volume_mesher;
12
13 volume_mesher_type::result_type     result=
14   volume_mesher(quality(orienter(hull_mesher(data))));
```

A reader object is instantiated and the input geometry is extracted and transferred to the specific data structure (Line 1). A wrapper is used to provide a unified access for arbitrary data structures to ViennaMesh (Lines 2,3). Note that we apply a wrapper approach to decouple the meshing modules from the input data structure type. This enables the utilization of ViennaMesh with other applications or libraries without the need to copy the data into a new data structure. Instead, a wrapper has to be provided which provides traversal and access mechanisms without unnecessary copy operations. A hull mesh generator, orienter, and quality adaptation module is created (Lines 4-9). A volume mesh generator is constructed (Lines 10,11). The various meshing modules are concatenated and the result is retrieved (Lines 13,14). The basic principle is depicted in Figure 1.

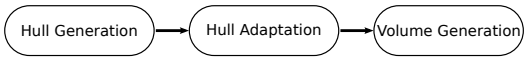

Fig. 1: Applied meshing process chain. An initial hull mesh is generated, which is then adapted in regard to orientation and quality. The adapted hull mesh is then used as input for the volume meshing step.

Meta-programming techniques are applied to provide a selection mechanism for mesh generation tools based on desired properties during compile time. If information about the input geometry is available during compile time, for example, the data structure which holds the input geometry provides meta-information, the mesh generation selection setup can be coupled with the data structure. Hence, no run time selection is required to choose a suitable mesh generator for a specific input geometry. This approach has another important advantage, as the underlying mesh generation kernel is decoupled from the application. As can clearly be seen in the following code snippet. There is no explicit indication which mesh generation kernel is actually used, only the properties which the mesh generator must provide.

```
1 typedef make_map<
2   key::celltype, key::geomDim,
3   val::simplex,  val::three >::type   properties;
4 typedef compute_mesh_generator<
5   properties>::type                   mesh_generator;
```

The associative relations in Line 2 and Line 3 define the properties of the requested meshing kernel. A meta-function evaluates the requested properties and selects a suitable meshing kernel to be further used as basis for the generation task (Lines 4, 5). In this case a three-dimensional, simplicial mesh generator is selected, which could be based on, for example, TetGen [5] or Netgen [11]. Note, that the set of properties can be arbitrarily extended to provide highly adjustable property setups.

Generic programming is used as the fundamental programming paradigm, and can be seen as the basis for the applied meta and functional programming paradigms. It enables to achieve, for example, highly extendable code, as for example depicted in the following.

```
1 typedef mesh_generator<netgen>::type  mesh_generation_type;
2 typedef mesh_adaptor<orienter>::type  mesh_adaptation_type;
```

Compile time tag-dispatched meta-functions are used to declare a mesh generator and adaptation type. The tag system can be easily extended to support additional meshing tools.

III. Improved Hull Mesh Adaptation

We present the continuation of previous work carried out at our institute regarding three-dimensional, unstructured mesh generation for TCAD [12] [13]. The major focus of this work is on improving the hull mesh adaptation process with regard to robustness and quality. This significantly improves the subsequent volume mesh generation step, as the hull mesh is the basis for the volume mesh generation. As already noted, this holds especially true for advancing front algorithms.

Among the improvements is an algorithm based on the well-known edge-removal [14] approach, which removes highly degenerated hull elements after the initial hull mesh generation step (Figure 2). The algorithm removes triangular hull mesh elements with a large ratio of the longest to the shortest edge by collapsing the shortest edge.This significantly improves the robustness of the successive mesh generation and adaptation tools. For example, the distinction between the two vertices of very short edges may become numerically inconclusive.

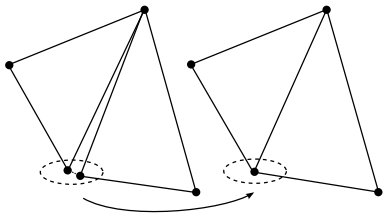

Fig. 2: Principle of the edge-removal algorithm. A short edge is removed and the corresponding vertices are merged.

Furthermore, an orientation tool has been implemented which is capable of orienting the hull mesh elements of a multi-segment, also known as multi-material, meshes. The orientation is performed counter-clockwise when looking onto the mesh element. Such an orientation is, for example, required by Netgen [4]. This is especially of high importance as input meshes might not be oriented either consistently or in a counter-clockwise manner. The challenge of orienting a hull mesh is to determine whether or not the normal vector of a mesh element is pointing outwards or inwards with respect to the segment. To overcome this obstacle a ray-intersection test has been implemented. The algorithm generates a normal vector of one surface mesh element of a segment and sets up a ray pointing in the direction of this vector. This ray is then tested, if it intersects with any of the cells on the related segment. If the number of intersections is zero or an even number, the normal vector points outwards. On the contrary, if the number is an odd number, the vector points inwards and therefore has to be corrected (Figure 3). Note, that this approach is based on the assumption that each segment of the mesh for itself is a closed (without a boundary) two-dimensional manifold [15]. The intersection algorithm of the Computational Geometry Algorithms Library (CGAL) [6] has been utilized.

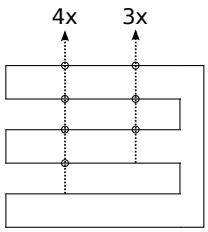

Fig. 3: Principle of the ray-intersection algorithm based on an exemplary two-dimensional, concave mesh domain. The left ray intersects an even number of times, hence it points outwards. The right ray crosses the surface an odd number of times, therefore the ray points inwards.

This ray intersection test is performed for one hull element per segment. When its direction has been determined and possibly corrected, the orientation is then consistently imposed on the other elements of the segment.

IV. Results

Figures 4-7 depict results for various industry provided meshes in the field of TCAD. The volume meshes are shown both for the raw and the adapted input hull meshes. Our adaptation tool has been used to improve the quality of the hull mesh. Netgen (Version 4.9.13) has been used as volume meshing kernel. For the volume mesh generation tasks, the default parameters result in poor mesh qualities (top). However, as our hull adaptation module is capable of self-adjustments, no human parameter interaction at all is required to achieve high-quality meshes (bottom). Note that often the number of cells and points is increased due to the adaptation. However, as depicted in Figure 6 a significant decrease in cells and points can sometimes be achieved by simultaneously increasing the overall mesh quality. Furthermore, we use a specific classification scheme for triangles and tetrahedrons [13]. By analyzing the dihedral angles of a tetrahedron four different types of degeneracy are identified: *wedge*, *spade*, *cap*, and *sliver*. Furthermore the triangular faces of a tetrahedron can be analyzed by investigating the angles and the edge ratios, which therefore yields three different types of degenerated triangles: *needle*, *slat*, and *spindle*. Aside from the degenerated elements a good element, a so-called *round*, is characterized by edges of similar length [9]. Generally, a high percentage of round elements indicates a high-quality mesh, whereas a high percentage of degenerated elements, like *slivers* or *wedges*, denote a low-quality mesh. A significant increase of the mesh quality can be identified for all meshes.

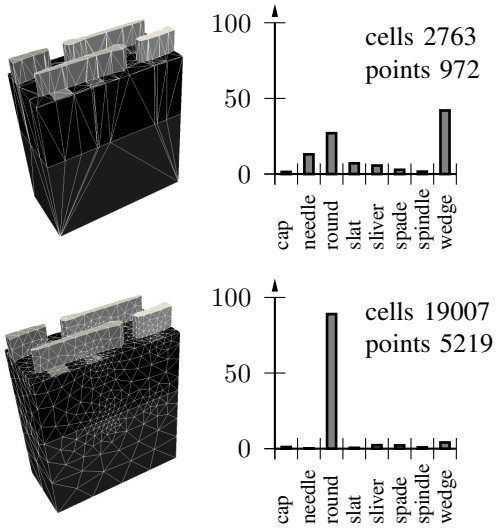

Fig. 4: Comparison of the generated volume meshes by Netgen of the initial (top) and the adapted (bottom) discretization of a SRAM cell. Almost all degenerated elements have been removed, especially the wedges, needles, and slats. Additionally, the percentage of the round elements has been increased remarkably from 27% to 89%.

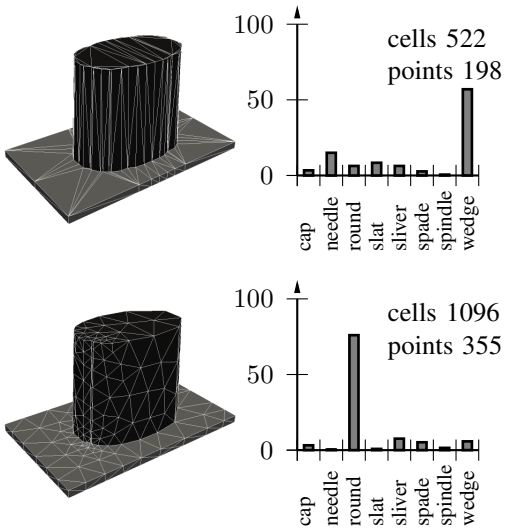

Fig. 5: Comparison of the generated volume meshes by Netgen of the initial (top) and the adapted (bottom) discretization of a lithographically generated structure. The presence of the degenerated wedge elements has been decreased drastically, whereas the percentage of the round elements has been increased significantly from 6.3% to 76%.

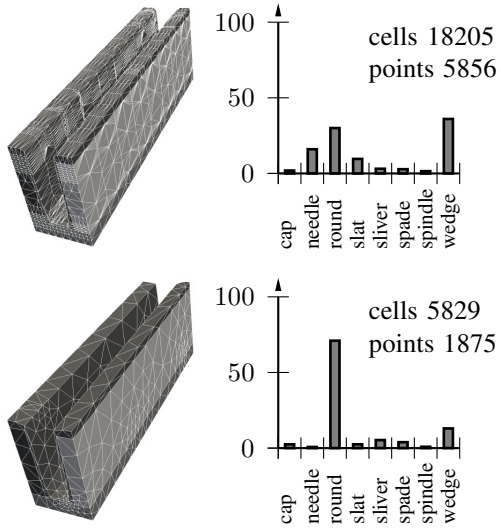

Fig. 6: Comparison of the generated volume meshes by Netgen of the initial (top) and the adapted (bottom) discretization of a trench structure. The degenerated needle and wedge elements could be reduced considerably, while the percentage of the round elements has been more than doubled, from 30% to 71%. Aside from the significant mesh quality improvement, the number of mesh elements has been reduced drastically.

V. Conclusion

In conclusion, we provide a means to access different meshing modules in a unified way, to tackle the challenge of mesh generation for TCAD. Those meshing modules cover generation, adaptation, and classification. The generic library ViennaMesh has been introduced to fulfill the discussed requirements. ViennaMesh is open-source and publicly available [10].

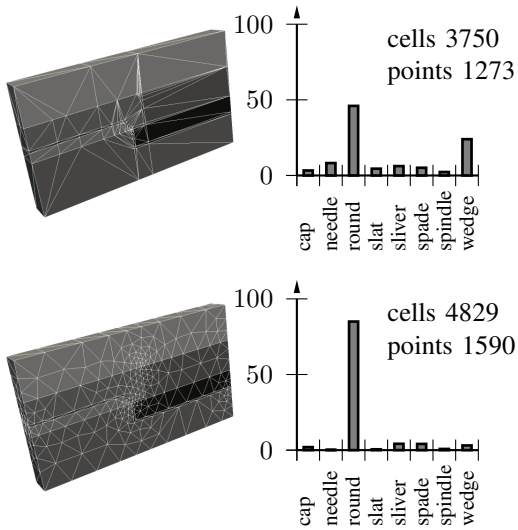

Fig. 7: Comparison of the generated volume meshes by Netgen of the initial (top) and the adapted (bottom) discretization of an interconnect structure. An overall decrease of the degenerated elements can be identified. The percentage of the round elements has been increased from 46% to 85%.

Acknowledgments

This work has been supported by the European Research Council through the grant #247056 MOSILSPIN. Karl Rupp gratefully acknowledges support by the Graduate School PDE-Tech at the TU Wien.

References

[1] M. Bern and D. Eppstein, "Mesh Generation and Optimal Triangulation," *Computing in Euclidean Geometry*, pp. 23–90, 1992.

[2] J. Shewchuk, "Delaunay Refinement Mesh Generation," Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1997.

[3] B. Klingner and J. Shewchuk, "Agressive Tetrahedral Mesh Improvement," in *Proceedings of the International Meshing Roundtable*, 2007, pp. 3–23.

[4] "Netgen." [Online]. Available: http://sourceforge.net/projects/netgen-mesher/

[5] "Tetgen." [Online]. Available: http://tetgen.berlios.de/

[6] "Computational Geometry Algorithms Library." [Online]. Available: http://www.cgal.org/

[7] "Triangle." [Online]. Available: http://www.cs.cmu.edu/ quake/triangle.html

[8] P. Fleischmann and S. Selberherr, "Enhanced Advancing Front Delaunay Meshing in TCAD," in *Proceedings Simulation of Semiconductor Processes and Devices*, 2002, pp. 99–102.

[9] J. Shewchuk, "What is a Good Linear Element?" in *Proceedings of the International Meshing Roundtable*, 2002, pp. 115–126.

[10] "Viennamesh." [Online]. Available: http://viennamesh.sourceforge.net/

[11] J. Schöberl, "NETGEN An Advancing Front 2D/3D Mesh Generator Based on Abstract Rules," *Computing and Visualization in Science*, no. 1, pp. 41–52, 1997.

[12] F. Stimpfl, R. Heinzl, P. Schwaha, and S. Selberherr, "A Robust Parallel Delaunay Mesh Generation Approach Suitable for Three-Dimensional TCAD," in *Proceedings Simulation of Semiconductor Processes and Devices*, 2008, pp. 265–268.

[13] R. Heinzl and T. Grasser, "Generalized Comprehensive Approach for Robust Three-Dimensional Mesh Generation for TCAD," in *Proceedings Simulation of Semiconductor Processes and Devices*, 2005, pp. 211–214.

[14] E. B. de l'Isle and P.-L. George, "Optimization of Tetrahedral Meshes," *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, IMA Volumes in Mathematics and its Applications*, vol. 75, pp. 97–128, 1995.

[15] L. Kinsey, *Topology of Surfaces*. Springer, 1997, ISBN-10: 0387941029.