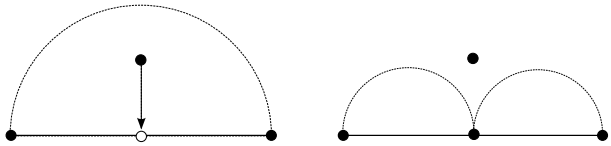
boundary element is then removed from the front and the new boundary elements are added to the front, depending on their visibility. This process terminates when no elements remain in the front.

The advantages of this method are the good control mechanism over the element size and the quality of the generated elements. A major drawback is that the quality of the generated elements depends heavily on the quality of the boundary elements and the colliding fronts. Different implementations of this type of mesh generation technique suffer from severe robustness issues.

Due to the fact that the advancing front depends heavily on the quality of the boundary, the first step, the processing of the boundary, assures that all boundary elements are satisfying the conforming Delaunay property [3]. The input to this step is a topologically two-dimensional and geometrically three-dimensional boundary representation using 2-simplices.

To get a Delaunay tessellation all encroached boundary elements need to be identified and refined. Therefore, not only the boundary vertices but also the volume vertices are taken into account. One straightforward method is to refine the boundary element by an orthogonal projection of the encroaching vertex onto the boundary element, as depicted in Figure 1.



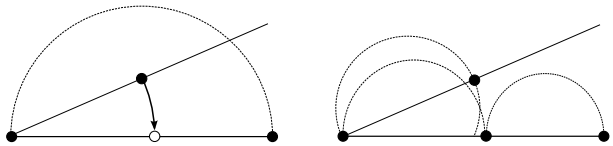
1: A surface element and the circumcircle which is encroached by a volume vertex (left). The resulting two Delaunay surface edges, after the orthogonal projection of the encroaching vertex (right).

The refined boundary element is split into new boundary elements, depending on the dimension of the boundary element, e.g., a projected vertex onto a boundary edge is split into two new boundary edges. This procedure creates new boundary elements, being locally Delauany.

There is a second case, where the encroaching boundary or volume vertex is incident to another boundary element and, using an orthogonal projection, the created refinement would itself become an encroaching vertex, due to numerical errors. This situation may lead to an endless refinement loop, which limits the applicability of the orthogonal projection.

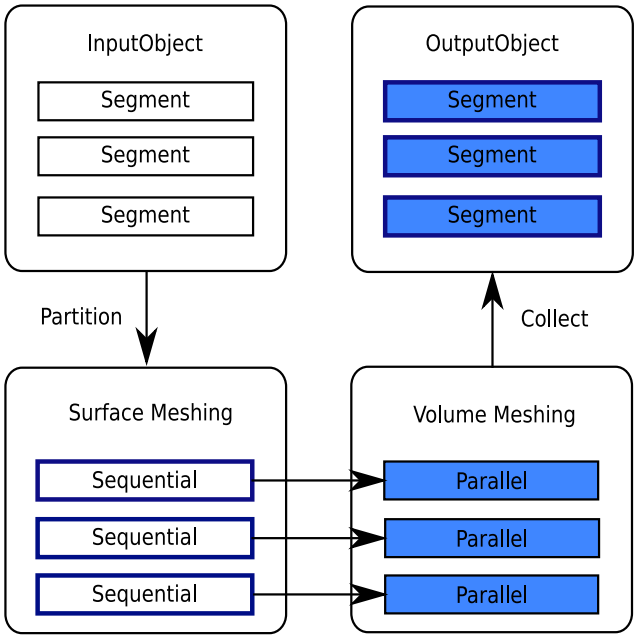
For this case an azimuthal rotation of the encroaching vertex around the intersection of the boundary elements instead of the orthogonal projection is performed. An example for the azimuthal rotation is depicted in Figure 2.

The necessary projections and rotations to fulfill the Delaunay criterion are controlled by abstract rules [4]. Using these abstract rules the procedure of mesh generation, e.g., how new points are inserted or how certain elements are treated during the meshing process, is defined. The rules are specified in a



2: An edge and the circumcircle which is encroached by a vertex on an incident edge (left). The resulting two Delaunay surface edges after the azimuthal rotation of the encroaching vertex (right).

unit coordinate system and the current element is transformed to this unit coordinate system, a matching rule is applied, and the results are transformed back to the original mesh. The procedure of choosing a matching rule can be performed by various criteria, e.g., element size or element quality.



3: An overview of the presented meshing approach. Starting from an initial input geometry the surface preprocessing step is performed. The segments are meshed in parallel and in the last step the resulting meshed segments are merged into one output geometry.

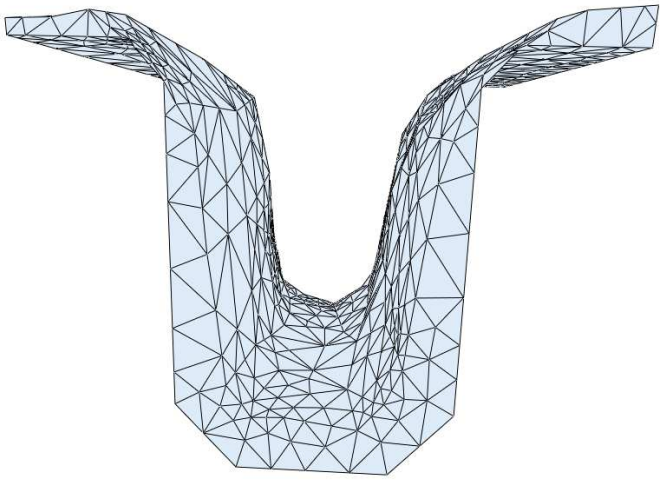
In the subsequent step the advancing front algorithm traverses all existing boundary elements and creates new volume elements according to the conforming Delaunay property. The volume vertex closest to the boundary element, which does not encroach the boundary element, is used to create a new volume element [3].

It has been shown, that if all elements are locally Delaunay, then the whole tessellation is globally Delaunay [3], which proves that the presented Delaunay meshing approach results in a Delaunay conforming volume mesh. Figure 3 depicts our developed parallel meshing approach, starting with the sequential surface treatment step. Then the volume meshes of

Example	Sequential Meshing	Two-fold Meshing	Four-fold Meshing	Num. points	Num. segments
Diffusion Example	105sec	59sec	52 sec	3.6e5	2
MOSFET	164 sec	65 sec	28 sec	2.6e4	6
Levelset	39 sec	19 sec	16 sec	1.3e4	3

I: Comparisons of the mesh generation and included mesh adaptation times (in seconds) on AMD's X2 5600 and AMD's Phenom 9600 Quad-Core.

the individual segments are created in parallel before being collected to form the final mesh.



4: A surface, extracted from the level set algorithm, already imported in CSG and processed using several logical operations.

This parallel mesh approach can not only be used for mesh generation but also to refine a given mesh. Due to the constrained boundary representation and the advancing front, all subdomains can be adapted in parallel.

Additionally, by exploiting the features provided by GSSE [5], we have combined Constructive Solid Geometry (CSG) [1] with existing methods of mesh generation [6] to further add flexibility in modeling. In the CSG approach all geometrical operations rely on formally derived operations [4].

IV. Results

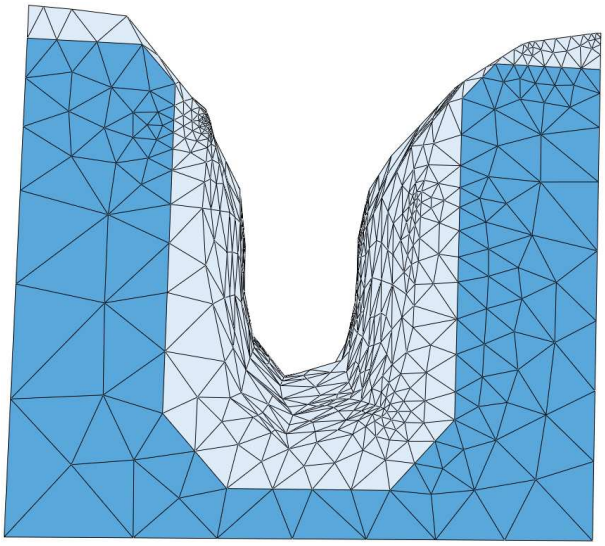
The following presents meshing examples, where the implicit level set surfaces were remeshed before applying the CSG operations. Using the presented parallel meshing procedure and the CSG topography approach the construction of meshes for the etching and deposition steps during process simulation can be performed robustly and more efficiently.

The procedure starts with the creation of the desired structure using the CSG method. There are several CSG primitives, which allow to create any structure performing logical operations on these primitives.

Further steps such as deposition or etching are performed using a levelset approach with a certain speed function [7]. Then the resulting surface is extracted using the marching cubes algorithm [8].

We extract a topological two-dimensional and geometrical three-dimensional surface from the structured mesh. Figure 4 shows the surface output from the levelset method for a trench.

Further logical operations can be applied to this segment once the surface is included in the domain. This step is again executed with CSG operations. The resulting segment is shown in Figure 5.

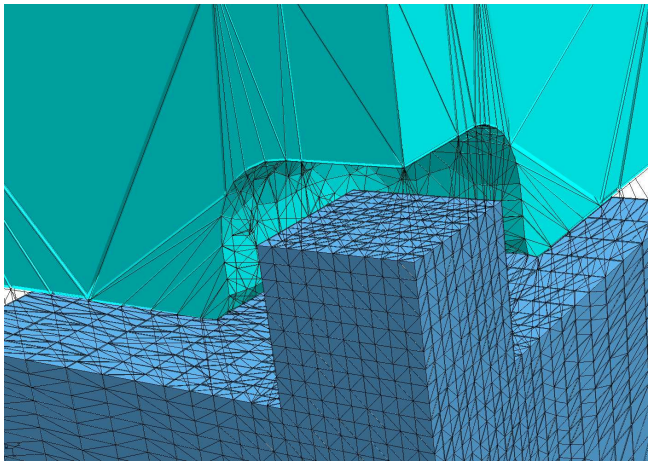


5: The resulting trench with a deposited layer originating from an anisotropic speed-function.

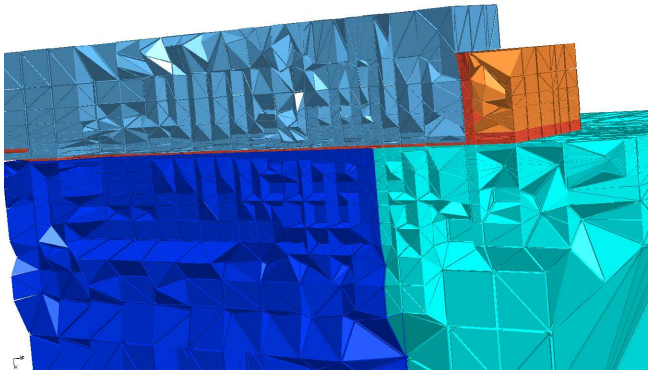
Once the surface is included in the domain, additional logical operations on this segment can be performed, e.g. remeshing or limiting the local feature sizes. Adaptive meshing can also be easily performed, because the refinement step is not different from the actual meshing step. Also the cycle of depositing a new surface or etching from the mesh can be performed several times using the same mesh. All volume meshing processes are performed in parallel. Results of the parallel meshing process can be viewed in Table I. Figures 6-8 depict additional industrial examples, which have been meshed using our approach.

V. Conclusion

The discussed robust, parallel mesh generation approach provides concepts to overcome limitations of three-dimensional mesh generation for process and device simulation. Due to the application of multi-paradigm programming

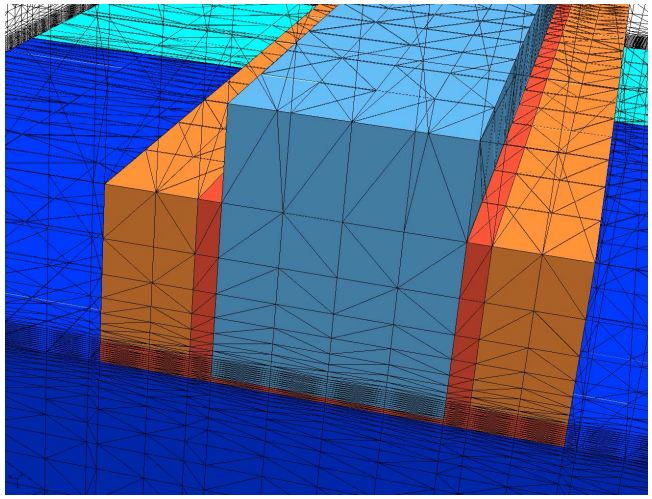


6: The mesh resulting from the extraction of an implicit surface.



7: Meshing of areas with different feature sizes for well-adapted device simulation meshes.

techniques, the whole mesh generation process can be performed in parallel, thereby drastically decreasing the runtime demands of the meshing process and, as a result, decreasing the overall runtime of TCAD simulations.



8: Meshing of thin layers of a three-dimensional device structure is made possible without imposing additional meshing overhead.

VI. Acknowledgment

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

References

[1] R. Heinzl and T. Grasser, in *Proc. Conf. in Sim. of Semiconductor Processes and Devices* (Tokio, 2005), pp. 211–214.

[2] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray, in *SCG '04: Proc. 20th ASCG* (ACM Press, 2004), pp. 290–299.

[3] J. R. Shewchuk, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997.

[4] J. Schöberl, Comput. Visual. Sci. **1**, 41 (1997).

[5] R. Heinzl, M. Spevak, P. Schwaha, and T. Grasser, in *Proc. of the PARA Conf.* (Umea, Sweden, 2006), p. 61.

[6] F. Stimpfl, R. Heinzl, P. Schwaha, and S. Selberherr, in *ESM 2007* (St. Julians, Malta, 2007), pp. 506–513.

[7] J. A. Sethian, *Level Set Methods and Fast Marching Methods* (Cambridge University Press, 1999).

[8] P. Bourke, *Polygonising a Scalar Field*, 1994, http://local.wasp.uwa.edu.au/ pbourke/geometry/polygonise/.