

Evaluating Parallel Feature Detection Methods for Implicit Surfaces

Christoph Lenz^a, Alexander Scharinger^a, Michael Quell^a,
Paul Manstetten^a, Andreas Hössinger^b, and Josef Weinhub^a

^aChristian Doppler Laboratory for High Performance TCAD, Institute for Microelectronics, TU Wien

^bSilvaco Europe Ltd., St Ives, UK

A common approach when simulating semiconductor fabrication processes is to rely on an implicit surface representation: Multiple level set functions are used to model topographical changes during individual process steps [1]. To perform high resolution simulations with feasible run times, one approach is to use adaptive resolutions to capture local features, such as sharp edges. A precise and fast feature detection method, guiding the adaption mechanism, is desirable to optimize the overall computational efficiency.

We implemented and compared three shared-memory parallelized methods to detect features of a level set function: (1) *Normal flagging* which compares the angle of each normal vector to the normal vectors in a local neighborhood. In contrast, the other two methods are based on calculating the *mean curvature* of the surface, i.e., using the (2) *general formula* for mean curvature of a level set function and the (3) *shape operator*. However, the latter requires the level set function to be a signed-distance function [2]. All three methods flag the same regions. However, they differ in the extent of the required neighborhood information (stencil) and how many floating-point operations (FLOPs) are performed per grid point. The shared-memory parallelization is accomplished by domain segmentation based on the underlying hierarchical run-length encoded data structure of the level set representation [1].

Fig. 1 shows the performance results for all three feature detection methods benchmarked on a single node of VSC-4 and based on a representative *Trench* geometry stemming from an exemplary semiconductor fabrication process. The speedup for methods 1 and 2 is comparable for up to 48 threads, whereas method 3 shows inferior speedup. However, the absolute run time of method 3 is 3 times

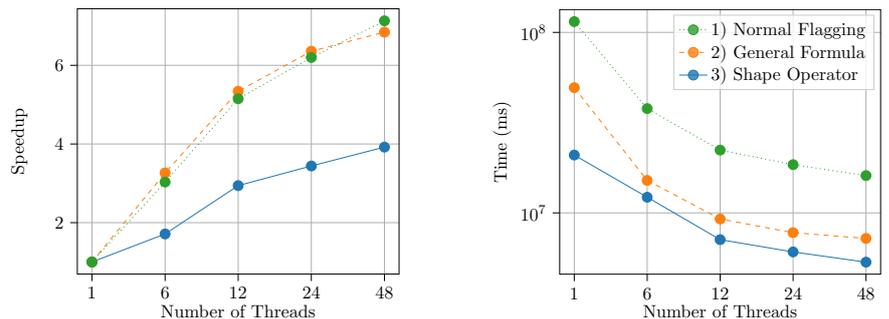


Fig. 1: Run time and speedup of a *Trench* geometry with 10^8 grid points.

smaller than method 1 and about 1.5 times smaller than method 2. The low multi-core scalability is expected due to the fact that all methods require a relatively low amount of FLOPs and are thus inherently memory-bound. That being said, method 3 has the shortest serial run time as it requires the smallest stencil. However, its speedup is thus also limited as data traffic dominates due to an even smaller number of required FLOPs compared to the other methods.

Acknowledgement. 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. The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

References

- [1] O. Ertl: Numerical Methods for Topography Simulation, Doctoral Dissertation, TU Wien (2010)
- [2] Sethian, J.A.: Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press (1999)