Towards a Free Open Source Process and Device Simulation Framework

J. Weinbub*, K. Rupp*†, L. Filipovic*, A. Makarov*, S. Selberherr*
*Institute for Microelectronics, TU Wien, Gußhausstraße 27-29, 1040 Wien, Austria
†Institute for Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8-10, 1040 Wien, Austria
e-mail: weinbub@iue.tuwien.ac.at

Abstract. The core aspects of our approach for an open source simulation framework are discussed and application results are shown.

Introduction. The field of semiconductor process and device simulation offers a plethora of publicly available simulation tools [1]. However, only a fraction of the tools is available under an open source license. This impedes the progress of research in academia, as researchers are unable to access the code base of previously implemented software and extend it. In such a case, simulation tools must eventually by re-implemented. Clearly, this introduces additional development overhead which has a negative impact on the actual net research time. Therefore, the field of semiconductor process and device simulation in academia can greatly benefit from free open source software packages. These facts have already been established by various software developments, like the Archimedes project [2]. Other fields, such as computational fluid dynamics, show that this concept works by providing multi-purpose simulation frameworks, like the OpenFOAM software package [3]. Our work details an approach to the design of a simulation framework in C++, which is aimed at a free open source release. Our focus is on supporting the dynamics within an academic environment, primarily imposed by an inherent flow of scientific personnel. This refers to the typical scenario of researchers' temporary employment within academic institutions. Therefore, we focus on extendibility, maintainability, flexibility, and high-performance. We apply modern programming techniques, like generic programming, and utilize third-party libraries, like the Boost libraries [4]. Thereby the overall code base is reduced and re-usable, intuitive, and high-performing implementations are achieved.

The Framework. Our approach is based on an extendible framework design (Figure 1), which utilizes the so-called self-registering technique [5]. For each simulation tool a separate plugin is provided, which acts as a wrapper for the individual software packages. The framework supports the execution of plugins, which, due to their eventual interdependencies, requires a graph-based execution scheduler (Figure 2 left). The scheduler supports the execution on computing nodes by the Message Passing Interface (MPI), which enables the utilization of an arbitrary distributed computing environment (Figure 2 right). The data exchange between the plugins and the framework is based on a notification system. Each plugin notifies the framework of its input and output data dependencies. The system centrally stores the required datasets and makes them available for the plugins, using an MPI capable transmission system.

Results. Some simulation results of three of our in-house developed simulation tools are shown, all of which have been executed as plugins in our framework. Figure 3 depicts the evolution of a surface over time computed using our Level Set simulation tool [6]. Figure 4 shows the electron distribution in a symmetrically sliced, active FinFET device, evaluated using our spherical harmonics expansions (SHE) based Boltzmann equation (BE) solver [7]. Figure 5 outlines the magnetization vector field of a switching process for a penta-layer MTJ STT-RAM computed using our micromagnetic simulation tool [8].

Summary. Our approach for an open source simulation framework has proven to be efficient in integrating heterogenous simulation packages with small effort.

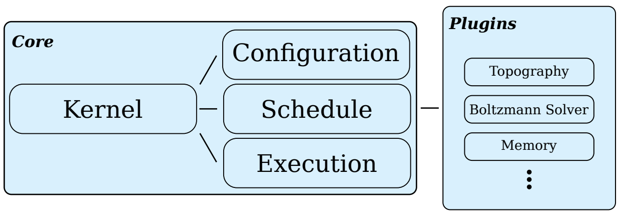

Fig. 1: The design of the framework is depicted. The plugins are controlled by the core parts of the framework.

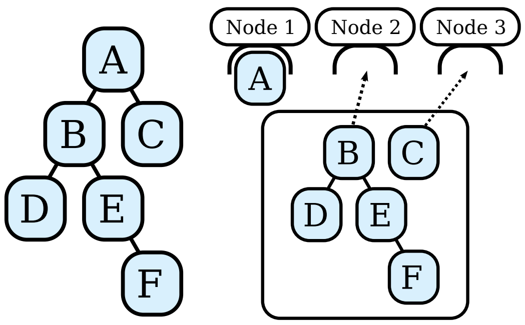

Fig. 2: **Left:** A task-graph is used to map the plugin dependencies derived from the user input. **Right:** The tasks are assigned to distributed computing nodes according to their dependencies. As soon as plugin A is finished, the plugins B and C can be executed in parallel.

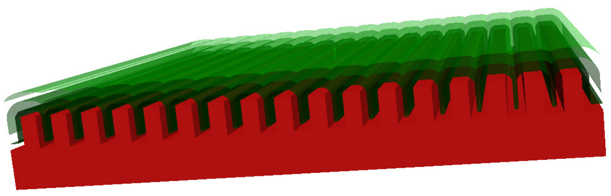

Fig. 3: A result of our Level Set simulation plugin is shown. The green surfaces denote the evolution of the surface position over time starting from an initial surface (red).

Acknowledgment. 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.

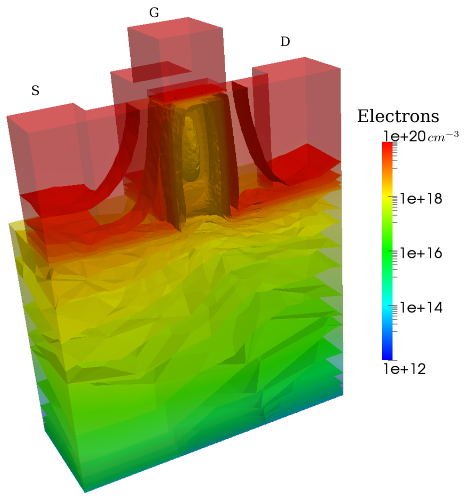

Fig. 4: An electron distribution based on a sliced, active FinFET device is shown, which has been computed using our SHE-based BE solver.

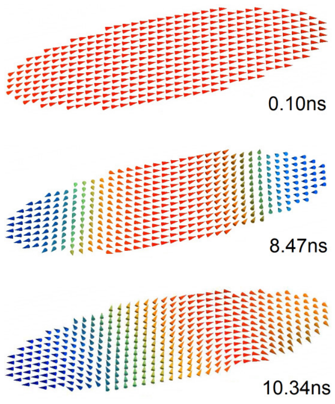

Fig. 5: The magnetization of a penta-layer STT-RAM at different time steps is shown, computed using our micromagnetic simulation tool. The colors indicate the x-component of the magnetization.

References

[1] nanoHUB, *http://nanohub.org/*
[2] Archimedes, *http://www.gnu.org/software/archimedes/*
[3] OpenFOAM, *http://www.openfoam.com/*
[4] Boost, *http://www.boost.org/*
[5] D. Kharrat, *et.al.*, *Self-Registering Plug-ins: An Architecture for Extensible Software*, CCECE, 2005.
[6] L. Filipovic, *et.al.*, *Parallelization Strategy for Hierarchical Run Length Encoded Data Structures*, PDCN, 2011.
[7] K. Rupp, *et.al.*, *On the Feasibility of Spherical Harmonics Expansions of the Boltzmann Transport Equation for Three-Dimensional Device Geometries*, IEDM, 2011.
[8] A. Makarov, *et.al.*, *Reduction of Switching Time in Pentalayer Magnetic Tunnel Junctions with a Composite-Free Layer*, Phys. Status Solidi Rapid Res. Lett. **5** (12), 2011.