Parallelization of a Monte-Carlo Ion Implantation Simulator for Three-Dimensional Crystalline Structures

A. Hössinger[1], M. Radi[1], B. Scholz[2], T. Fahringer[2], E.Langer[1], and S. Selberherr[1]

[1]Institute for Microelectronics, TU Vienna
Gusshausstr. 27–29, A-1040 Vienna, Austria
[2]Institute for Software Technology and Parallel Systems, University of Vienna
Liechtensteinstr. 22, A-1092 Vienna, Austria

Abstract

The simulation of ion implantation using a Monte-Carlo method is one of the most time consuming tasks in process simulation, which makes it a first-order target for parallelization. We present a parallelization strategy for the Monte-Carlo ion implantation simulator MCIMPL based on the message passing interface (MPI), with an almost linear performance gain.

1. Introduction

When simulating semiconductor production processes, ion implantation is a very important, but also one of the most critical steps, concerning the simulation time. Due to the complicated structures and the small dimensions of modern semiconductor devices, Monte-Carlo simulation methods often have to be used to describe non-planarity effects, phenomena resulting from ion channeling and large tilt angles, and to provide accurate point-defect distributions for rapid thermal annealing processes. To reach the expected accuracy, three-dimensional simulations have to be performed with sophisticated models [1][2], especially for very shallow implantation conditions. By meeting all these requirements the simulation times exceed one night or even more on high-end workstations for large structures. Therefore the parallelization of the Monte-Carlo ion implantation simulation process step is desirable to avoid a bottleneck in the process simulation flow, because normally a cluster of workstations is available to perform process simulation and optimization. We present a parallelization method which allows a distributed simulation on a cluster of single processor workstations. Our parallelization strategy is based on MPI and it allows to reuse all sophisticated methods and models [1][3] developed for the single processor version without modification.

2. Parallelization Strategy

We use a master-slave strategy, where the master process provides all the I/O operations and controls and synchronizes the behavior of all slaves which perform the actual simulation. The basic concept of the Monte-Carlo ion implantation simulation method is that the trajectories through the simulation domain are calculated for a large number of ions. The final positions of calculated particles and the number of generated point-defects are stored in a histogram which is used to derive the particle

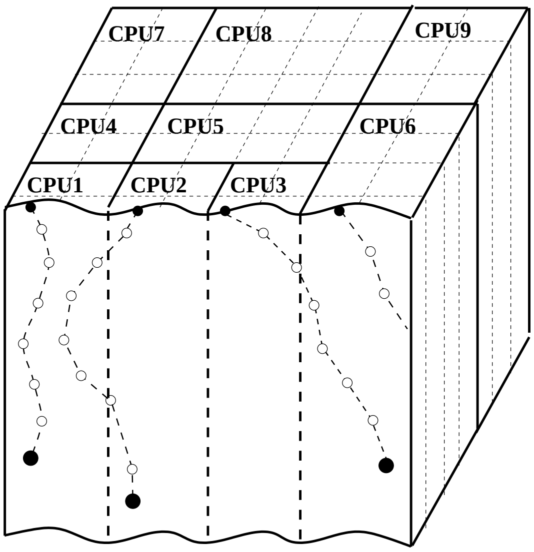

Figure 1: Schematic presentation of the split of the simulation domain into subdomains and of the distribution of the subdomains among several processors.

and point-defect distributions. The parallelization is achieved by splitting the simulation domain into several rectangular prismatic subdomains (Fig. 1). Each available CPU is responsible for several of these subdomains. This means that all particles moving through a certain region of the simulation domain are calculated by a certain CPU and that all simulation results are stored in the memory belonging to a certain CPU. Thereby the trajectories calculated as well as the memory consumption are distributed among several workstations. When initializing the simulation the master process determines the distribution of the subdomains according to the number and the speeds of the available CPUs, considering the following conditions.

$$\frac{V_i}{CPU_i} \simeq const., \forall i \tag{1}$$

$$\sum_i \frac{O_i}{V_i} \to max \tag{2}$$

V_i, O_i are, respectively, the volume and the surface of a prismatic scope belonging to one processor, as denoted in Fig. 1 by the thick lines. CPU_i is the relative computing capability (e.g, floating point operations per second) of one processor. (1) ensures that every CPU gets a reasonable amount of trajectories for calculation, because a faster CPU can calculate more trajectories than a slower CPU. Due to the fact that the implanted ions are equally distributed over the device surface the number of trajectories that have to be calculated by one CPU is proportional to the volume of the prismatic scope belonging to this CPU, assuming that the same models are used throughout the whole simulation domain.

(2) guarantees a minimum of communication between the CPUs. It is possible that a particle moves to the prismatic scope of another CPU during its motion through the simulation domain. In that case the particle described by its physical and modeling properties has to be exchanged between two CPUs. Due to the fact that such communication events always limit the performance gain of a parallelized application it is desirable to minimize this communication. The probability that a particle leaves the scope of a CPU is all the lower the larger the volume V_i of the scope, and it is all the higher the larger the interface area O_i to other CPUs.

In order to take into account the influence of crystal damage on the trajectory of an ion a transient simulation is explicitly introduced. This is not necessary for a single processor version, because in that case the single trajectories are calculated one after the other while in the parallelized version the order of calculation is not deterministic. It is assumed that the ions belonging to the same time step do not influence each other and therefore the order of calculation is of no relevance within one time step. This requirement is met if the number of ions that are calculated per time step is significantly smaller then the total number of calculated ions.

3. Simulation Flow

First the master parses the command-line, reads the input files and initializes the physical properties of the simulation domain, the physical models and the implantation conditions. The initialization data are sent to all slaves. Then the master creates the subdomains and evaluates a distribution scheme. The subdomains and the distribution scheme are sent to all slaves. Thereby all processes are informed about the scope of responsibility of all other processes too. Whenever a particle trajectory leaves the scope of responsibility of a slave, this slave knows where to send the particle.

After this initialization the master process calculates the initial conditions of the implanted ions and prepares them for being sent to the slaves. When all ions in all subdomains belonging to one time step are prepared one package of ions is sent to each slave. Then the initial conditions of the ions of the next time step are prepared for sending before the master process enters a wait-loop where it determines if all slaves have finished the last time step. The sending, packing and waiting is repeated until the total number of simulated ions is calculated. Afterwards the master process sends a 'Simulation Finished' request to all slaves, collects the information about all simulated ion trajectories, performs the statistical analysis for the resulting doping and point-defect distributions, prepares the generation of the output and writes the output files.

The slaves enter a wait-loop immediately after the initialization process, where they are waiting for a request either from the master or from a slave. Five types of requests are managed.

(a) **Ion Package Request:**
The trajectories of the ions in the ion package which is sent either from the master or from another slave are calculated. During the trajectory calculation simulation data can be stored in or received from the area of responsibility of another slave. Moreover a simulated particle can be moved to another slave by sending an appropriate request.

(b) **Store Simulation Data Request:**
Simulation data that have to be stored in the local memory are received from another slave. Information about the type and the position of the data is received before storing the data.

(c) **Send Simulation Data Request:**
Information about the type and the position of required data is received, and the corresponding simulation data are sent to another slave.

(d) **New Time Step Request:**
The initialization procedure for a new time step is called.

(e) **Simulation Finished Request:**
While the slave returns to the wait-loop when he has finished one of the above requests he leaves the loop in case of a 'Simulation Finished' request. All simulation results are sent to the master before the slave terminates his operation.

4. Results

By measuring the performance gain on a network of workstations it has turned out that this parallelization strategy delivers an almost linear performance gain (Fig. 2), but only if the processor load is constant on all CPUs throughout the simulation. In case of varying processor loads the performance gain can decrease dramatically, because the calculation time for all trajectories at one time step is not constant anymore on all CPUs. Thereby one slave always holds up all other slaves due to synchronization at the end of each time step. Additionally it has to be mentioned that the size of one subdomain must be larger than the lateral range of the implanted ions to keep the communication overhead low. This limits the number of available subdomains and thereby the number of CPUs that can be used for a parallel simulation. However for three-dimensional application the number of subdomains is larger than 1000 which normally exceeds the number of available processors.

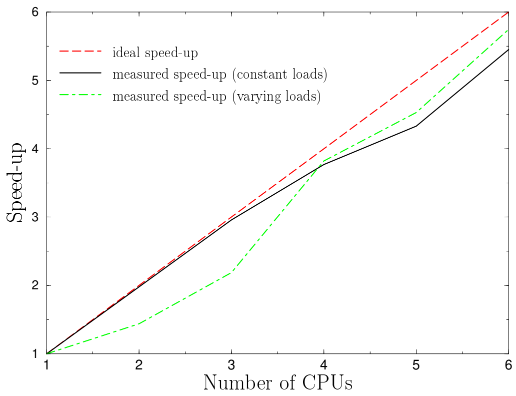

Figure 2: Measured speed-up compared to the ideal speed-up for two different load situations.

5. Conclusion

We have presented a method for parallelizing a Monte-Carlo ion implantation simulator with a minimum of communication overhead and therefore an almost linear speed-up. For strongly varying processor loads a dynamic load balancing should be implemented. Besides a tremendous reduction of the simulation time a splitting of the memory requirement is achieved, which allows to utilize several small workstations for the simulation of large three-dimensional problems.

6. Acknowledgment

This work has been carried out within the SFB project AURORA, funded by the Austrian Science Fund (FWF).

References

[1] A. Hössinger and S. Selberherr, "Accurate Three-Dimensional Simulation of Damage Caused by Ion Implantation," in *Proc. 2nd Int. Conf. on Modeling and Simulation of Microsystems*, pp. 363–366, Apr. 1999.

[2] A. Hössinger, S. Selberherr, M. Kimura, I. Nomachi, and S. Kusanagi, "Three-Dimensional Monte Carlo Ion Implantation Simulation for Molecular Ions," in *Proc. of the 5th Int. Sym. on Process Physics and Modelling in Semiconductor Technology*, pp. 18–25, Apr. 1999.

[3] W. Bohmayr, A. Burenkov, J. Lorenz, H. Ryssel, and S. Selberherr, "Trajectory Split Method for Monte Carlo Simulation of Ion Implantation," *IEEE Trans.Semiconductor Manufacturing*, vol. 8, no. 4, pp. 402–407, 1995.