Performance Optimization of a Parallelized Three-Dimensional Monte-Carlo Ion Implantation Simulator

Andreas Hössinger, Erasmus Langer, and Siegfried Selberherr
Institute for Microelectronics, TU Vienna
Gusshausstr. 27-29, A-1040 Vienna, Austria

KEYWORDS

Monte-Carlo, Performance Analysis, Distributed Processors, VLSI & Simulation, Ion Implantation.

ABSTRACT

We present a parallelization method based on MPI (Message Passing Interface) for a Monte-Carlo simulator for the simulation of ion implantation into three-dimensional structures. By this method the simulation domain is geometrically distributed among several CPUs which exchange simulation data during the simulation. We optimize the performance gain by identifying bottlenecks of this strategy when it is applied to arbitrary shaped simulation domains consisting of various materials, which requires the application of varying physical models within the simulation domain and which makes it impossible to determine a reasonable domain distribution before starting the simulation. Due to a feedback procedure between the parallelization strategy and the simulation by online performance measurements, we obtain an almost linear performance gain on a cluster of workstations with just slightly varying processor loads. Besides an increase in the performance the parallelization method achieves a distribution of the memory requirement, which allows also the use of small workstations for three-dimensional simulations.

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 and phenomena resulting from ion channeling and large tilt angles. To reach the expected accuracy, three-dimensional simulations have to be performed with very sophisticated models [Hössinger and S. Selberherr 1999], [Hössinger et al. 1999], especially for very shallow implantation conditions. By meeting all these requirements the simulation times exceed one night or even more for large structures, depending on the required accuracy and the applied models. Therefore the parallelization of the Monte-Carlo ion implantation simulation step is desirable to avoid a bottleneck in the process simulation flow, since normally a cluster of workstations is available to perform process simulation and optimization [Strasser et al. 1998].

PARALLELIZATION STRATEGY

We use a master-slave strategy based on MPI (Message Passing Interface), 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 parallelization is achieved by splitting the simulation domain into several rectangular prismatic subdomains. Each available CPU (Slave) is responsible for a prismatic scope (thick lines) consisting of several of these subdomains and, therefore, for a certain part of the simulation domain as shown in Fig. 1.

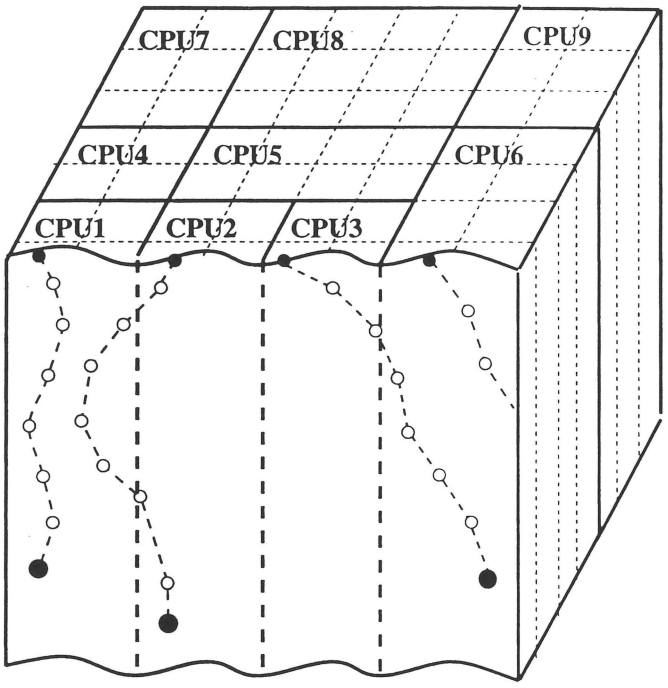

Figure 1: Schematic presentation of the split of the simulation domain into subdomains and of the distribution of the subdomains among several processors. The prisms denoted by the thick lines are the scopes of responsibility of the slaves.

When performing the Monte-Carlo ion implantation the master process calculates the initial conditions for all ions

and distributes them among the slave processes which perform the calculation of the ion trajectories through the simulation domain. The simulation results are the spatial distribution of the implanted ions when they come to rest due to interactions with the nuclei and the electrons of the target material, and the distribution of material defects that are generated by the implanted ions. The simulation results are stored locally at the slaves. Thereby not only a parallelization of the simulation is achieved but also a partitioning of the memory requirement.

Besides the distribution scheme the trajectories of the ions are denoted by the long dashed lines in Fig. 1. When an ion moves to the scope of responsibility of an other slave information has to be transfered to the other slave. It is obvious that communication between slaves can only occur if an ion is located in the vicinity of a subdomain border. Thereby the probability for communication increases with a decreasing subdomain size and with an increase in the lateral range of the ions. By choosing a subdomain size larger than the lateral range of the implanted ions the communication overhead due to data or ion exchange between the slaves is almost negligible. This is the major advantage of this parallelization method because normally communication limits the performance gain of a parallelized application and requires therefore very fast communication networks. Due to the fact that the communication has just very small influence on the performance the simulation can be performed on a cluster of workstations connected by a standard network.

While in the single processor version the implanted ions are calculated successively whereby the damage accumulation during the implantation is considered automatically, a transient simulation is introduced explicitly in the parallelized version. This means that all ion trajectories belonging to one time step can be calculated independently while a synchronization of all slaves is necessary after each time step. This synchronization is achieved by the master process which successively sends ion packages at each time step to the slaves. Between two time steps the master process waits until all slaves have completed their calculations before sending a new package of ions. After finishing the simulation the master process collects all simulation results from the slaves, does some postprocessing, and writes the output files. The following list summarizes the simulation flow of the master and of the slaves.

Master Process:

- Parses the command line.
- Reads the input files.
- Initializes the physical properties of the simulation domain, the physical models and the implantation conditions.
- Sends the initialization data to all slaves.
- Creates the subdomains and evaluates the distribution scheme.
- Sends the subdomains and distribution scheme to all slaves.
- Calculates the initial conditions for all ions of the first time step.
- Distributes the ions among the slaves.
- Calculates the initial conditions for all ions of the next time step and prepares them for distribution.
- Waits until all slaves have finished their calculations before synchronizing the slaves and distributing the ions of the next time step.

} Repeated until the simulation is finished.

- Collects all simulation results from the slaves.
- Performs a statistical analysis of the resulting doping and point-defect distributions.
- Prepares the generation of the output and writes the output files.

Slave Process:

- Receives the initialization.
- Receives the distribution scheme.
- Repeatedly waits for a request and processes the request.
- Terminates after processing the 'Simulation Finished' request.

PERFORMANCE

The most important aspect of this parallelization strategy is to distribute the subdomains among the available CPUs in a manner that the simulation time per time step and CPU is constant for all CPUs. If this requirement is not met some slaves always hold up all other slaves because of the synchronization after each time step. Thereby the performance gain due to parallelization can become quite low because most of the processors are idle most of the execution time (Fig. 2). Worth mentioning is that even in the optimal case the average idle time of all CPUs will not be zero, because the calculation time for one ion trajectory is fluctuating due to the statistical nature of the simulated physical process. The larger the number of ions per time step the smaller the fluctuation.

For the initial distribution of the subdomains it has to be assumed that the number of implanted ions per subdomain and time step is constant and that the calculation time per implanted ion is also constant everywhere in the simulation domain. By only considering the performance ratios of the CPUs the number of subdomains N_i handled by CPU i which has a performance factor of CPU_i is

$$N_i = \frac{N_{sub}}{CPU_i} \cdot \left(\sum_{j=1}^{N_{CPU}} \frac{1}{CPU_j} \right)^{-1} . \qquad (1)$$

Figure 2: Average ideltime of the slaves relative to the execution time for two different distributions of the subdomain for a simulation with five slaves. The left bars result from a poor distribution while the right bars result from an optimized distribution.

N_{CPU} is the total number of available CPUs and N_{sub} is the total number of subdomains. As already mentioned the slaves can exchange simulation results or complete ions by sending and accepting requests during the calculation. To minimize this communication the interface area between the scopes of responsibility (Fig. 1) is minimized when distributing the subdomains, by successively adding subdomains to a CPU considering following conditions in the specified order:

- A subdomain with the largest number of interfaces to other subdomains of the CPU is prefered.
- By adding a subdomain the aspect ratio of the generated area of responsibility must be closest to one.
- A subdomain laying at the border of the simulation domain is prefered to subdomains inside the simulation domain.

If more than one subdomain pass all these requirements one is selected randomly.

When the first ion trajectories are calculated it normally turns out that the assumption of a globally constant calculation time is not correct, because the material composition is different in the subdomains and thereby different physical and statistical models [Hössinger and S. Selberherr 1999], [Bohmayr et al. 1995] lead to varying calculation times among the subdomains. In order to optimize the performance the master process measures the time that each slave needs to finish the calculation of the ion package of the first time step that it received from the master. Due to the measured time, the calculation time for one time step is estimated for every subdomain and the subdomains are redistributed accordingly. To avoid an update of the simulation data at the slaves, which takes significantly longer than the calculation of one time step, all simulation results are cleared and the simulation is started again with the first time step. As a consequence an almost linear performance gain is achieved for arbitrary simulation domains (Fig. 3), as long as the loads of the CPUs do not fluctuate too much.

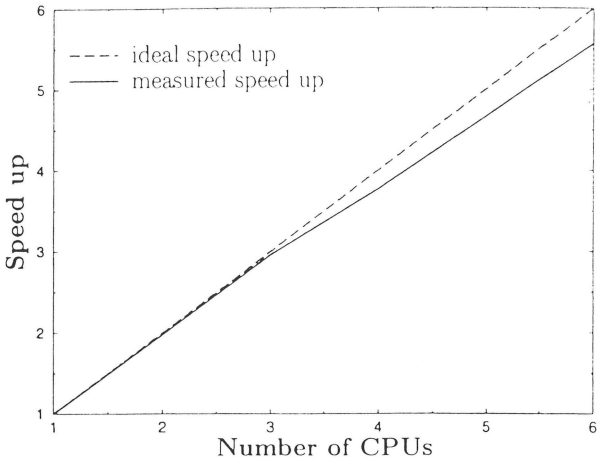

Figure 3: Measured speed up compared to the ideal speed up.

CONCLUSION

We have presented a method for parallelizing a Monte-Carlo ion implantation simulator with a minimum amount of communication overhead. An almost linear speed up is obtained for arbitrary shaped simulation domains even if various statistical methods are applied throughout the simulation domain, by reducing the the average idle time of the CPUs to a minimum.

ACKNOWLEDGMENT

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

REFERENCES

[Hössinger and S. Selberherr 1999] 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*, (San Juan, Puerto Rico, Apr.), pp. 363–366.

[Hössinger et al. 1999] 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*, (Seattle, N.J. Apr.), pp. 18–25.

[Strasser et al. 1998] R. Strasser, R. Plasun, and S. Selberherr, "Parallel and Distributed Optimization in Technology Computer Aided Design," in *12th European Simulation Multiconference - Simulation: Past, Present and Future* (R. Zobel and D. Moeller, eds.), pp. 78–80, Society for Computer Simulation, June 1998.

[Bohmayr et al. 1995] 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.