CONCURRENT SIMULATION OF PLASMA REACTORS

Marc Rieffel, Stephen Taylor, and Jerrell Watts
Scalable Concurrent Programming Laboratory
Syracuse University, Syracuse, NY 13244
e-mail: steve@scp.syr.edu
and
Sadasivan Shankar
Technology Computer Aided Design Group
Intel Corporation, Santa Clara, CA 95052

KEY WORDS

VLSI and Simulation, Multiprocessors, Concurrency, Monte Carlo, Plasma.

ABSTRACT

This paper summarizes the computational techniques behind a novel concurrent simulation method used for studying the neutral flow inside plasma reactors. The technique is intended to cope with low pressure flow (less than 1.5 Torr) in realistic three-dimensional geometries. It is based on the Direct Simulation Monte Carlo method to accurately model rarefied gas flow. The concurrent formulation operates on a broad variety of shared-memory multiprocessors, multicomputers, and networked workstations.

INTRODUCTION

Plasma processing equipment accounts for approximately 20 percent of the setup cost for a new semiconductor manufacturing plant. The cost of these production facilities is escalating, as are the research and development costs of introducing each new generation of processing technology. One of the key pieces of equipment in microelectronics manufacturing is the *plasma reactor*, used in 30 to 40 percent of processing steps. It is widely recognized that the development of computational tools for modeling plasma reactors is essential to reduce the costs, both of validating new reactor designs, and of improving manufacturing processes.

Process engineers seek simulation methods that are increasingly realistic, pushing toward three-dimensional models of complex geometries and reacting chemistry. Acceptable simulation results require not only appropriate chemical and physical models, but also substantial investments in geometric modeling, grid generation, post-processing, and analysis techniques. These steps in the design cycle are interrelated, and the process of obtaining a meaningful result is frequently a time-consuming, iterative process. Simulation techniques will continue to have limited engineering applicability until each of these steps is addressed in a comprehensive simulation methodology.

This paper summarizes a novel concurrent simulation technique intended to address the overall design cycle of a simulation. It is described in the context of the simulation of low-pressure (less than 200 Pa / 1.5 Torr) neutral flow in the Gaseous Electronics Conference (GEC) reference cell. The technique integrates a variety of ideas taken from computational fluid dynamics and finite-element methods. A central aim is to exploit existing industrial tools, already in use by process engineers, to shorten the design cycle to acceptable engineering timescales.

The reactors of interest involve weakly collisional plasmas (with degree of ionization $\approx 10^{-4}$) with electron-impact gas-phase reactions and ion-enhanced surface reactions. Neutral flow is of importance in studying these systems since it determines the reactor pressure, the center-of-mass motion, and the production of chemically-active species. In addition, the lateral location and possible agglomeration of particulates are determined by the characteristics of neutral flow. Since pressure drops are higher in low pressure reactors, the study of neutral flow is critical to optimal design. The flow can be supersonic close to the showerhead, and good design is necessary to avoid large pressure drops and highly asymmetric flows.

Neutral flow in low pressure reactors is in the *transition regime*. The mean free path of particles is on

the order of the characteristic length of the reactor; this is too large for continuum methods, but too small for free-molecular simulations. The Direct Simulation Monte Carlo (DSMC) method (Bird 1994) is an alternative technique for modeling rarefied gas dynamics. Neutral flow calculations require only simple chemical models, yet are extremely useful from an engineering perspective. Moreover, these calculations are a natural precursor to more complex studies of ion and electron transport.

COMPUTATIONAL TECHNIQUE

The concurrent simulation technique involves four primary stages:

1. **Automatic Grid Generation.** A three-dimensional geometry definition is taken directly from CAD/CAM descriptions already available to process engineers. An unstructured tetrahedral grid is then constructed automatically.
2. **Static Partitioning.** The grid is subsequently partitioned for execution on any of a wide variety of concurrent architectures.
3. **Concurrent DSMC.** Scalable concurrent algorithms are then to reduce the numerical simulation time. These algorithms model neutral flow using the DSMC technique. Adaptive gridding is used to automatically maintain the accuracy of the simulation. Dynamic load balancing is used to maximize processor utilization.
4. **Post-Processing.** Finally, simulation results for each partition are re-constituted and analyzed using particle-based visualization tools.

Grid Generation & Partitioning

Reactors are typically designed using CAD/CAM packages, where engineers specify the curves and surfaces that define the reactor geometry. The concurrent simulation technique uses a boundary-fitted computational grid, extracted from this CAD/CAM definition.

Axially-aligned grids are unable to represent the curved geometry of a realistic reactor. Structured hexahedral grids are difficult to generate automatically for complex geometries. In contrast, mature techniques exist for automatic generation and smoothing of unstructured tetrahedral grids. These grids are able to efficiently represent complex geometries, and standard algorithms are available for their adaptive refinement.

Realistic computational grids can often be too large to fit in the memory of any single processor. The concurrent simulation technique is therefore based on the concept of *over-partitioning:* The grid is decomposed into a number of partitions, and several partitions are mapped onto each processor. Each of these partitions represents a portion of the grid that can be simulated concurrently by communicating information to neighboring partitions. This allows the memory requirements for each processor to be inversely proportional to the number of processors, and avoids address conflicts on shared-memory architectures.

The initial partitioning is based on a volume decomposition that preserves *locality*. Cells are sorted in each dimension and grouped according to their sorted positions. Adjacent cells are therefore likely to be mapped to the same partition. This procedure attempts to minimize the surface area of the partitions, and hence the total amount of communication required during a simulation. It also attempts to minimize the number of neighbors for each partition and therefore the number of separate communications required.

This simple partitioning strategy is sometimes inadequate if a large number of partitions is required. It allows the number of cells in a partition to vary widely, and it allows a partition to have an undesirably large surface area. These problems can result in imbalances in processor load and memory usage, and in excessive communication. In these cases, static balancing is necessary. This is achieved using a concurrent algorithm that computes the solution to the heat equation, $\frac{\partial u}{\partial t} = \alpha \nabla^2 u$ where u is an estimate of the memory and computational cost of a partition, and $\nabla^2 u$ is a measure of local imbalance. The parameter α represents the desired accuracy of load balance specified when setting up the simulation; for example, $\alpha = 0.1$ balances each processor to within 10%. The variable u is computed as the sum over all cells in a partition, $u = \sum_{cells} A + BV_{cell}$ where A is an estimate of the fixed cost per cell, and B is an estimate of the cell cost proportional to the cell volume V_i. Each cell is treated as a discrete quantity of heat that can be transferred between adjacent partitions in order to evenly distribute the cost. Cells are selected for transfer in such a way as to preserve locality and to minimize the area of surfaces shared between partitions. This process terminates when all partitions have roughly the same memory and computational costs.

Partitions are mapped to computers in such a way as to place neighboring partitions in the same or adjacent computers, again maximizing locality. Information

is exchanged between partitions on neighboring computers using message-passing, while information is directly exchanged between partitions on the same computer (or computers sharing memory). The mapping of multiple partitions to each computer allows the overlap of communication and computation: if one partition is blocked, awaiting communication, another partition on the same computer may still proceed.

Over-partitioning allows partitions of differing sizes to be mapped to the same computer in order to balance load and memory. Specifying the number of computers, and therefore the total number of cells per computer, provides control over the *granularity* of the computation, i.e., the ratio of computation to communication. Generally speaking, this corresponds to the ratio of the volume (n^3 for a characteristic dimension n) to the surface area of all partitions in a computer (n^2). Adjusting the granularity allows the simulation to be matched to a wide variety of concurrent architectures. On platforms with high communication costs, such as networks of workstations, a small number of large partitions may be used. By contrast, a larger number of smaller partitions may be used on distributed-memory multiprocessors that employ fast communication technology.

Concurrent DSMC

The Direct Simulation Monte Carlo (DSMC) method solves the Boltzmann equation by simulating individual particles. Since it is impossible to simulate the actual number of particles in a realistic system, a smaller number of simulation particles are used, each representing a large number of real particles. Statistical techniques are employed to obtain the correct macroscopic behavior. Computational grid cells are initially filled with simulation particles according to density, temperature, and velocity specifications. The simulation takes discrete steps in time, during which a *transport model* is used to move particles, a *collision model* is used for particle-particle interactions, and a *boundary model* is used for interactions between particles and surfaces. Macroscopic properties, such as density and temperature, are computed by appropriate averaging of particle masses, positions, and velocities.

Figure 1 outlines the concurrent algorithm executed by each partition of the computational grid. For the most part, particle transport is local within a partition (2a), however, a particle may move across a cell face on the boundary between two partitions. In this case, it is communicated to the appropriate neighboring partition (2b). In a single timestep, a particle may cross several partition boundaries and thus require several rounds of communication. In order to improve communication efficiency, all particles exchanged between a single pair of partitions are combined into a single message. Once all of the particles have been placed in their new cell locations, the collision and boundary models are employed independently within each cell (2c). At the end of a timestep, global information, such as the total number of particles in the domain, is computed (2d) and a check is made to determine if grid adaption (2e) or load balancing (2f) would be advantageous.

Figure 1: **Parallel Algorithm**

For each partition:

1. Initialize partitions according to initial conditions (locally)
2. While more steps are necessary
   (a) Calculate new particle positions (locally)
   (b) Exchange particles between partitions (local communication)
   (c) Collide particles (locally)
   (d) Compute global information (global communication)
   (e) Adapt Grid
   (f) Balance Load
3. Conclude computation

Transport Model. The transport model is concerned with moving particles through the computational grid for a specified period of time. It uses ray-tracing techniques to determine the final location of a particle at the end of a timestep. In an unstructured grid, this is achieved by computing a particle's path in each cell that it traverses.

To determine whether a particle will move into another cell, it is necessary to calculate which cell face the particle's trajectory will first intersect. In the presence of a uniform acceleration, particles follow *parabolic trajectories*. The intersection time between the particle's curved trajectory and each cell face, is calculated and the closest face along the trajectory is determined. The intersection time for that face is compared to the

timestep length Δt. If the intersection time is greater than Δt, the particle will not hit any faces in the current timestep and can therefore be moved directly to its new position. If, however, the intersection time is less than the timestep, the particle will hit the closest face. Most cell faces are internal to partitions, and particles move through them to adjacent cells. If the cell face is on the boundary between partitions, the particle is sent to the adjacent partition. If the cell face corresponds to a physical surface, the boundary model is employed to determine the interaction between the particle and the surface.

Since the transport model is primarily local to a partition only information about the cell containing the particle is used. This contrasts sharply with DSMC implementations that require global grid information in order to move particles. In that case, the entire grid must be replicated at each computer, limiting scalability.

Collision Model. The collision model characterizes particle-particle interactions. Simplistic collision models can lead to erroneous results, while unnecessarily complex models can dramatically increase simulation time. Binary collisions are fundamentally an N-squared interaction for N particles. For simulations involving millions of particles, it is impractical to consider all possible interactions. If the mean free path is larger than the cell size, there can be no statistically significant spatial gradients due to collisions. The DSMC method therefore needs only consider collisions between particles in the same cell. Collisions are thus performed independently within each cell and concurrently in each partition. Statistical techniques are used to determine the correct number of collisions in time proportional to the number of particles in the cell.

The collision model uses an acceptance-rejection method based on collision tests. The frequency of collision tests in a cell is computed. The collision model makes a series of tests at the appropriate frequency and probabilistically chooses particles to collide updating their post-collision velocities appropriately. A variety of collision models have been developed, however, for the Argon simulation presented here the Variable Soft Sphere (VSS) model (Koura 1992) is sufficient.

Boundary Model. During grid generation, cell faces on the same surface are grouped together. When configuring a simulation, a surface type is specified for each group. The surface type of a face determines particle-surface interactions on that face. The three surface types used in the GEC simulation are *inflow, outflow*, and *accommodating*. Standard DSMC techniques are used to model inflow and thermally-accommodating surfaces. The outflow model allows surfaces of the reactor to be partly-reflective. The reflectivity, r, specifies the probability that a particle hitting a surface will be specularly reflected back into the reactor. Particles that are not reflected exit the reactor, as though through an exhaust pump. The reflectivity is adaptively modified during the simulation so as to maintain the pressure at a chosen reference point within the reactor.

During grid partitioning, an additional surface type, *partition*, is created to represent shared boundaries between partitions. Recall that a partition surface is introduced at each cell face located on the boundary between partitions. A particle arriving at a partition surface is sent to the appropriate neighboring partition. The time taken for the particle to reach the surface is subtracted from the current timestep to obtain the remaining time during which the particle must be moved upon arrival at the neighboring partition.

Grid Adaption

In the DSMC method, error is related to the local Knudsen number, the ratio of the mean free path to the cell size. If the cell size is too large, erroneous results are obtained. If the cell size is too small, computational resources are wasted, and statistical scatter is increased. The mean free path is an average parameter computed during a simulation. Thus it is not possible to generate an optimal grid before simulations have been conducted.

To resolve this issue an iterative manual approach can be used. A rough initial grid is used to obtain estimates of the mean free path throughout the grid. Based on visual inspection of these estimates, a new grid is generated, with increased or decreased resolution where necessary. This process is repeated until an acceptable solution is found. This approach requires time-consuming manual intervention for non-trivial geometries and increases the overall simulation time. An alternative is to adapt the grid automatically during the simulation to ensure accuracy without wasting cells.

In our current simulations, a course initial grid is generated that serves to accurately define the reactor geometry. Periodically during the simulation, the local Knudsen number is computed in each cell. Cells with insufficient Knudsen numbers are marked for adaption so long as they contain a sufficient number of particles. The adaption technique leverages standard methods (Kallinderis 1993) drawn from finite element analysis. When a cell is marked for adaption, an edge in that cell is selected. A new point is introduced along

this edge, each cell that shares the edge is split into two new cells, and each face that shares the edge is split into two new faces. Careful selection of edges ensures that repeated adaption does not cause the grid quality to deteriorate.

Since grid refinement is a local process, it can usually be conducted within a single partition. In cases where an adapting edge lies on a partition boundary, only the partitions sharing that boundary need to be aware of the adaption. Thus the communication and synchronization costs of grid adaption are minimal. Furthermore, grid adaption can be performed infrequently during a calculation without substantial loss of accuracy.

Load Balancing

Recall that the computational cost of a single simulation timestep for a grid partition is a function of the number of grid cells and particles that it contains. Both these quantities change dynamically in the presence of particle transport and grid adaption. It is therefore impossible to initially partition a complex three-dimensional grid so that load will be balanced among processors throughout the computation. For this reason, in general, no static partitioning and mapping will yield an optimal load balance.

The concurrent simulation technique employs a load balancing algorithm based on the notion of heat diffusion: workload at a given computer is treated as heat to be diffused to other computers. This process is described by the parabolic heat equation, $\frac{\partial u}{\partial t} = \alpha \nabla^2 u$ where u is a measure of the work required for a single timestep in each partition and α is a specified accuracy of balance. The algorithm is composed of five distinct phases (Watts 1996):

Load Evaluation: The load of each partition is estimated based on empirical timings. Partition loads are summed on to determine computer loads.

Profitability Determination: Once computer loads have been measured, the presence of a load imbalance can be detected. If the gain of load balancing exceeds the cost, load balancing is performed.

Work Transfer Calculation: Neighboring computers exchange load information, and solve the heat equation, using a second-order Crank-Nicholson method. This process determines the ideal workload that must be transferred between each neighboring pair of computers in order to balance a calculation. This is a continuous value calculated to achieve an ideal distribution of computational load among computers.

Partition Selection: A set of partitions, each representing a discrete quantity of load, is selected for exchange between neighboring computers, in order to satisfy the ideal workload as closely as possible. This selection procedure may require partition exchanges in order to achieve the desired work transfers.

Partition Migration: Once selected, partitions are transferred from one computer to another. This process is fault-tolerant. In the event that insufficient memory is available at the destination, a recovery procedure is invoked.

This technique has a variety of useful properties: It provably converges and its rate of convergence can be determined analytically. It involves only nearest-neighbor communication and is therefore scalable. Load balancing is initiated infrequently and only when the outcome will reduce total simulation time.

EXAMPLE SIMULATION

A thorough validation of the neutral flow model has been completed. Results for several one-dimensional heat transfer problems have been compared with analytical results (Zhong 1995; Rieffel 1995). Results for two-dimensional hypersonic flow past a cylinder, with multiple gas species, each with rotational and vibrational energy modes, have been compared to numerical results from Ivanov *et. al.* (Gimelshein 1996). Suitable experimental data is not yet available for neutral flow in the GEC Reference Cell.

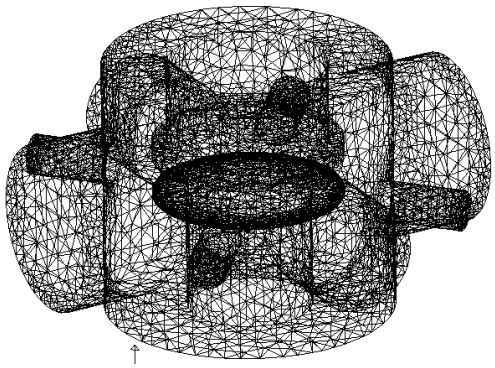

Figure 2: Initial Grid

The simulation shown here uses the geometry shown in Figure 2. Simulations were performed using Argon. Gas was injected through the showerhead above the wafer at 300K, with a density of $6.4 \times 10^{20} m^{-3}$ and a speed of 29.4 m/s; gas exits at the base of the reactor. The reactor walls are assumed to be accommodating

at 300K and the wafer is heated to 370K. The outflow surface adapts to maintain a pressure of 2.66 Pa (20 mTorr) at a point just above the center of the wafer. A timestep of 1×10^{-5} seconds was used.

Figure 3 shows an example of the flow results that are obtained and illustrates the average flow velocity in vertical plane. The flow rapidly curves around the wafer and down to the outflow. There is a clear non-uniformity above the wafer. Flow velocity is much higher at the edges of the wafer than at the center.

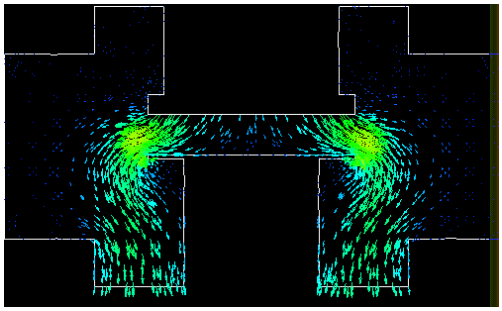

Figure 3: Vertical Velocity

Figure 4 shows the total kinetic energy of the system, obtained by summing the kinetic energy of each particle in the domain. This drops from a high initial value of around 0.05 J to a converged value of around 0.03 J. The slight increase around 0.15 seconds is the result of activating grid adaption. Results were obtained by averaging from around 0.15 seconds, and represent a point in the computation where all convergence has occurred and the measured pressure is within 1% of the target pressure.

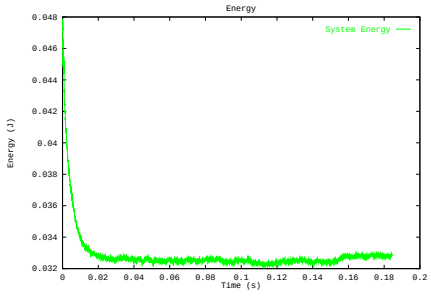

Figure 4: System Energy

CONCLUSION

This paper has outlined a novel concurrent DSMC method and demonstrated the method on three-dimensional simulations of the GEC reference cell. Large-scale simulations of realistic reactor geometries are now possible within realistic engineering timescales. Using a relatively small parallel machine such as a 14-processor SGI Power Challenge or a 72-processor Intel Paragon, simulations have been conducted on a variety of proprietary reactors that have had a direct impact on the understanding of process effects. These problems can typically be configured and simulated in about a week using several million particles.

ACKNOWLEDGMENTS

This paper summarizes the concurrent technique behind a substantial simulation effort being conducted jointly with Mikhail Ivanov and his group at the Russian Academy of Sciences, and David Weaver and his group at Phillips Laboratory, Edwards AFB. The research is sponsored by Intel Corporation and the Advanced Research Projects Agency under contract number DABT63-95-C-0116.

BIBLIOGRAPHY

Bird, G.. 1994. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Clarendon Press, Oxford, England.

Gimelshein, S.; G. Markelov; and M. Rieffel. 1996. "Collision Models in the Hawk DSMC Implementation." Technical Report CS-96-16. Department of Computer Science, Caltech, Pasadena, CA 91125.

Kallinderis Y. and P. Vijayan. 1993. "Adaptive Refinement-Coarsening Scheme for Three-Dimensional Unstructured Meshes." *AIAA Journal.* 31. 8.

Koura, K. and H. Matsumoto. 1992. "Variable soft sphere molecular model for air speciess." *Physics of Fluids.* 4. 5.

Rieffel, M.. 1995. "Concurrent Simulations of Plasma Reactors for VLSI Manufacturing." Masters Thesis CS-95-012. Department of Computer Science, Caltech, Pasadena, CA 91125.

Watts J.; M. Rieffel; and S. Taylor. 1996. "Practical Dynamic Load Balancing for Irregular Problems." In *Parallel Algorithms for Irregularly Structured Problems: IRREGULAR '96 Proceedings, 1117.* Springer-Verlag LNCS.

Zhong X. and K. Koura. 1995. "Comparison of Solutions of the Burnett Equations, Navier-Stokes Equations, and DSMC for Couette Flow." In *Rarefied Gas Dynamics 19:1.* Oxford University Press.