PARALLEL AND DISTRIBUTED OPTIMIZATION IN TECHNOLOGY COMPUTER AIDED DESIGN

R. Strasser, R. Plasun and S. Selberherr

Institute for Microelectronics, TU Vienna
Gußhausstraße 27–29, A–1040 Vienna, Austria
Phone: +43/1/58801-3680, Fax: +43/1/5059224,
e-mail: strasser@iue.tuwien.ac.at, URL: http://www.iue.tuwien.ac.at

ABSTRACT

We present the simulation environment SIESTA (SIMULATION ENVIRONMENT FOR SEMICONDUCTOR TECHNOLOGY ANALYSIS) which provides facilities for time effective parallel distributed simulation. An optimization framework and its components which explore these features are described. Due to distributed simulation, the overall simulation time is drastically reduced by SIESTA. Optimizations are performed on a heterogeneous cluster of UNIX workstations. A load balancing mechanism is used for effective selection of computation hosts. An example sketches the rigorous calibration of a semiconductor device simulator MINIMOS(Simlinger et al. 1995).

INTRODUCTION

Technology CAD tools (Halama et al. 1993, Pichler 1997) deliver accurate and highly valuable predictions of the performance of semiconductor devices based on technological parameters. These tools reduce the design cycle time by substituting simulations for test runs in the fabrication facility. However, computation time limited these investigations to individual simulations with tedious human interaction.

Advances in computing power and the ongoing progress in the development of simulation tools provide the resources that are necessary for simulations going beyond traditional simulations in this field. Rigorous optimizations with practicable time requirements became possible. The major reason why such experiments are often not carried out is the absence of a mediator between simulation tools and computer resources. Although at most sites plenty of workstations are potentially available, most of them are not used. Furthermore, simulations often have to be started manually and simulation results are individually analyzed which turns out to be extremely time consuming and error prone. In some extreme situations the simulation setup time is the time limiting factor. The framework SIESTA closes this gap by providing a methodology for efficient simulation problem formulation and by efficiently distributing the workload on a workstation cluster.

OPTIMIZATION DEMANDS

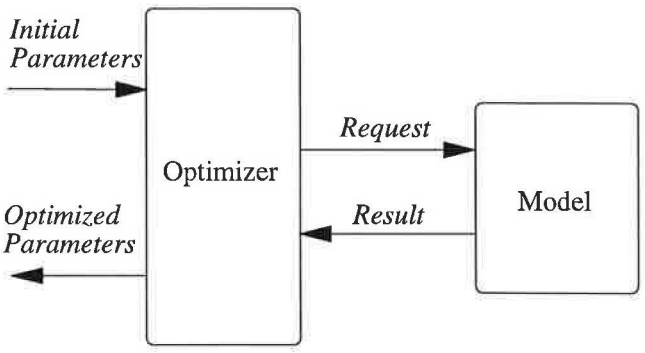

Figure 1: A typical setup for an optimizer applied to the optimization of some system parameters of a model

Figure 1 shows the typical setup for an optimization task. Such a configuration is also used for the optimization of semiconductor devices and their fabrication processes. Several process parameters (implantation energy/dose, annealing time, mask dimensions) which determine the electrical characteristics of a semiconductor device are controlled by the optimizer and altered in order to find an optimum of some characteristic device parameter (I_{on}/I_{off}, V_{th}, g_m). In such a scenario not the optimization itself, but the huge amount of model evaluations (which themselves incorporate several calls to simulation tools) represents the challenge. Despite of the existence of high performance workstations, distributed parallel computation on a workstation cluster has to be exploited in order to reduce the overall computation time to a practicable time scale. Nevertheless, system components, namely the optimizer and the system model, must provide parallelization features.

MODEL REPRESENTATION

The model is an abstract vehicle which provides an evaluation interface between the optimizer and arbitrary

kinds of evaluations. Parameters and their bounds as well as responses are defined in the model. From the implementation point of view it serves as the base class which provides the communication protocol for more specialized models that inherit this functionality. As already mentioned the model has to offer mechanisms which allow for parallel (asynchronous) evaluation. This means that a client of the model is able to request several evaluations in parallel and receives notification in the event that individual results are available. Also errors and other more specific events are communicated to the client. To facilitate a clean data interface a so called *request* class has been established which is used for data exchange. This means that the set of input parameters is handed over to the model by means of the request object and the corresponding results are returned by the same object.

OPTIMIZER REPRESENTATION

Similar to the implementation of the model, an interface which handles communication and data transfer to the optimizer exists. Using inheritance again, various implementations of optimizers can easily be integrated due to a well defined optimizer interface. The actual optimizer is usually represented by a separated system process which reads data (evaluation results) on its standard input and writes data (such as evaluation request) to standard output. As mentioned before, the more parallelization an optimizer allows, the more efficient the computation resources can be used. Currently interfaces to a nonlinear optimizer with bound constraints DONLPQ (Spellucci 1996) and to a Levenberg-Marquart optimizer exist. The former is used for global nonlinear optimization of device characteristics and the latter is employed for calibration applications (Grasser et al. 1998).

PROCESS MODEL

The components described so far form a general framework for optimization. Specific models have to be designed to act as a representation for real world systems or processes. In this sense the *process model* is an abstraction of a semiconductor device which was derived by a technological process with specific parameters. In some degenerate cases it can also just represent an existing device and its electrical behavior (as necessary for pure device simulation). Figure 2 depicts the components of a process model. In addition to the components of the base class there is the so called *flow* and a list of *aliases*.

The flow describes a sequence of filters, each of which is identified by a key (`gate-etch` and `ldd-implant` in the example) representing a specific simulator call. Each filter is controlled by its controls which are used for simulator control. The output of each filter has to be compatible with the input of its successor (`Wafer`

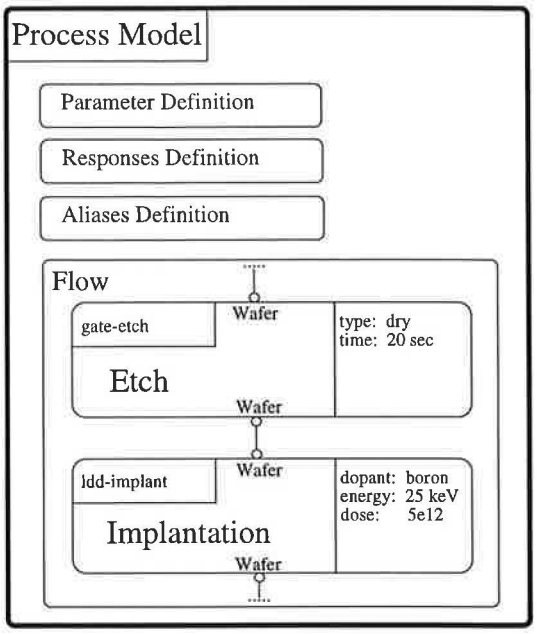

Figure 2: The process model represents a technological process and the resulting semiconductor device

in Figure 2), which is checked by the process model. One can imagine a filter as a call to a simulator. However, the filter represents a general simulator interface and several subclasses of the base filter class fulfill the needs of specific classes (classified by simulator invocation and input data type) of simulation tools. Basic functionality like error control, logging, data type checking is inherited from the base class.

This concept of filters allows to cover a wide range of simulation tasks; it also serves as a good abstraction for actual processing steps a wafer is exposed to.

The aliases entries map several flow controls (which are internal to the process model) to parameters of the process model, which are accessible from outside the process model.

QUEUE MANAGER

To get the work done that arises from the evaluations requested by the optimizer one needs to spawn several system jobs that perform the actual calls to the simulators. If parallel evaluation is a requirement also the parallel execution of system jobs is necessary. Furthermore, a mechanism to distribute workload on a workstation cluster is a must because otherwise parallelization would not result in a decreasing elapsed time required for the computation.

The SIESTA *Queue Manager* offers all these facilities. Its `task` class of objects offers the facilities to create system processes, read their input, and receive notification when they terminate. To further minimize computation time SIESTA performs dynamic load balancing with individual weights and load limits for computation hosts.

This mechanism leads to minimization of computation time and at the same time prevents hosts from being overloaded since the load is continuously monitored.

The load values that are queried by SIESTA are subject to time delays due to the common definition of the system load of the UNIX operating system. Usually a system job is reported with a delay of up to one minute which inhibits simple implementations of load balancing. The SIESTA Queue Manager takes this into account by estimating the effective load.

APPLICATION

SIESTA has proven to be extremely powerful applied to the calibration of the semiconductor device simulator MINIMOS. Starting from traditional values for parameters (electron mobility, work function) of MINIMOS these values are altered in order to minimize the least square error between simulated values of device currents and available measurements. For the estimation of the time required for the calibration let us assume the following: Transfer curves (I_D/V_G) with a total number of operating points of N are available, the number of parameters to be calibrated is M, W workstations are available for computation, and the typical computation time required per operating point is one minute (which is a realistic value for a workstation with a 200 MHz Pentium Processor). The evaluation of one set of parameters requires N simulations. Optimization usually needs to compute gradients of responses with respect to the optimized parameters which results in $M \times N$ evaluations. Assuming that each gradient computation is followed by an additional evaluation the number of experiments per iteration is $(M+1) \times N$. For $N = 30$, $M = 4$, $W = 15$ and 100 iterations the resulting computation times are summarized in Table 1. It shows that the required time can be reduced from approximately one week to one day. Furthermore, this means that such calibration experiments performed in a way as described should be part of the standard simulation repertoire of everyone who performs simulations.

	local	distributed/parallel
time	10 days 10 hours	16 hours

Table 1: An estimation of computation times demonstrates the potential of SIESTA's facilities for distributed parallel computing

ARCHITECTURE AND IMPLEMENTATION

The functionality comprised by SIESTA is provided by strictly separated modules. Reuse of modules for completely different applications is therefore possible and has proven to be effective. Examples of SIESTA modules are *Gui, Queue Manager, Optimizer* and the *Model* module. The modular architecture will have positive impact on further development and will significantly improve maintainability. Based on a LISP interpreter (Betz 1989, VISTA 1996) with a powerful object system, SIESTA is a highly portable system available on a wide range of UNIX systems.

CONCLUSION

An optimization framework dedicated to the modeling and optimization of semiconductor devices and processes has been presented. A compact and flexible methodology for the specification of simulation problems has been described. The mechanisms for parallel and distributed simulations drastically reduce simulation time which makes the framework very attractive to practical problems arising at industrial sites. Optimizations can be carried out within a reasonable timescale with a minimum effort to setup the simulations.

REFERENCES

Betz, D. 1989. *XLISP: An Object-Oriented Lisp, Version 2.1.* Apple Peterborough, New Hampshire, USA.

Grasser, T.; Strasser, R.; Knaipp, M.; Tsuneno, K.; Masuda, H., and Selberherr, S. 1998. Calibration of a Mobility Model for Quartermicron CMOS Devices. this conference.

Halama, S.; Fasching, F.; Fischer, C.; Kosina, H.; Leitner, E.; Pichler, C.; Pimingstorfer, H.; Puchner, H.; Rieger, G.; Schrom, G.; Simlinger, T.; Stiftinger, M.; Stippel, H.; Strasser, E.; Tuppa, W.; Wimmer, K., and Selberherr, S. 1993. The Viennese Integrated System for Technology CAD Applications. In Fasching, F.; Halama, S., and Selberherr, S., editors, *Technology CAD Systems* pages 197–236 Wien. Springer.

Pichler, C. 1997. *Integrated Semiconductor Technology Analysis.* Dissertation Technische Universität Wien.

Simlinger, T.; Kosina, H.; Rottinger, M., and Selberherr, S. 1995. MINIMOS-NT: A Generic Simulator for Complex Semiconductor Devices. In de Graaff, H. and van Kranenburg, H., editors, *25th European Solid State Device Research Conference* pages 83–86 Gif-sur-Yvette Cedex, France. Editions Frontieres.

Spellucci, P. 1996. Solving General Convex QP Problems via an Exact Quadratic Augmented Lagrangian with Bound Constraints. `http://www.mathematik.th-darmstadt.de/ags/ag8/spellucci`.

VISTA 1996. *VISTA Documentation 1.3-1, VLISP Manual.* Institut für Mikroelektronik Technische Universität Wien, Austria.

ACKNOWLEDGMENT

This work was significantly supported by the "Christian Doppler Forschungsgesellschaft", Vienna, Austria and Austria Mikro Systeme, Unterpremstätten, Austria.