Simulation Environment for Semiconductor Technology Analysis

Ch. Pichler, R. Plasun, R. Strasser, and S. Selberherr

Institute for Microelectronics, TU Vienna, Gusshausstrasse 27-29, A-1040 Vienna, Austria
Phone +43/1/58801-5239, FAX +43/1/5059224, e-mail pichler@iue.tuwien.ac.at

A programmable simulation environment for semiconductor technology analysis is presented. The *SIESTA* project sets out to combine the advantages of a comfortable, intuitive visual user interface with the flexibility and versatility of a high-level programming language. It is designed to provide a set of framework services to ensure a straight-forward definition of complex technology analysis tasks. Special emphasis has been put on establishing in an object-oriented fashion a uniform and easy-to-use interface for applications and extensions supplied by the user. LISP objects are used to represent basic entities like process steps, process flows, experiments, and agents for design of experiments (DoE), response surface modeling (RSM), and optimization and fitting modules.

The integration of heterogeneous process and device simulation tools is based on PIF [1] as a common tool data exchange format. Nevertheless, native tool data formats are supported as well, ensuring maximum flexibility in the choice of simulation tools for a particular task or at a particular site, while minimizing computation overhead and numerical errors due to excess data conversion. The automatic generation of split points both for sequentially and simultaneously initiated runs and the parallel and distributed execution of independent split tree branches allow a fast computation of large-scale experiments. A persistent run data base keeps the results (output files and extracted data) of each completed step and prevents unnecessary re-computations.

Architecture and Components. The implementation of the simulation environment is based on the VISTA framework [2] and it's extension language VLISP, an enhanced version of XLISP. A class of *evaluable entity* (EVE) objects has been defined to provide uniform access to basic services like process flow simulation and RSM evaluation as well as to complex tasks specified by the user. They establish a standardized interface to perform evaluations and can be defined in terms of existing EVEs in a nested manner. E.g., the minimization of the bulk current for a given process is encapsulated in an object that is evaluated for a set of initial value vectors for the optimizer generated by the DoE module or by a simple LISP loop.

Fig. 1 shows the main functional components and their interactions. All communication between objects and between the framework and external executables works on an asynchronous basis to allow simultaneous control of a number of independent tasks within the simulation environment. The optimizer module [4] performs optimizations under non-linear constraints and nonlinear parameter estimations, the DoE module creates experiment inputs according to standard design schemes, and the RSM module generates and evaluates polynomial response surfaces of up to second order. Both the DoE and RSM modules support a number of transformations to adequately represent non-linear systems.

The run controller uses a process flow representation [3] to build the desired devices. It generates split trees and controls parallel computation of simulation jobs on the network. A number of framework tools provide gridding and geometry services to ensure data consistency and facilitate data exchange between different simulators. Note that both batch and interactive operations are fully supported. A batch file can be used to run complete analysis tasks over night, to customize the environment and to define new objects, or to load and initiate an analysis while monitoring and controlling it via the visual user interface.

Applications. To illustrate the use of high-level commands to specify a complex TCAD task, fig. 2 shows the definition of a central composite circumscribed design (CCC) for the bulk current and the threshold voltage for the N-device as functions of implantation energies and doses for a standard Si-gate CMOS process [5]. The `Define-Task` statement creates an EVE from a process flow description, `Eve-Define-Control` statements mark specific process parameters as control variables for subsequent evaluations. The `Eve-Set-Control` initializes nominal values and ranges of all control variables as well as transformations to use. The process consists of 79 fabrication steps and has been simulated on a mixed cluster of UNIX workstations using PROMIS, TSUPREM4, and MINIMOS simulators. Fig. 3 shows part of the visual user interface during parallel computation of the runs created by the DoE module.

Conclusions. A highly programmable environment designed to support the most complex of TCAD tasks has been presented. The concept of evaluable entities provides a sound basis for the definition and execution of nested and aggregate analysis problems. Heterogeneous simulators and framework services are effectively encapsulated and form the basic building blocks for higher-level applications in process design, process analysis and process optimization.

[1] F. Fasching et al., "A PIF implementation for TCAD purposes", *SISPED IV*, 1991, pp. 477–482

[2] S. Halama et al., "The Viennese Integrated System for Technology CAD Applications", *Technology CAD Systems*, Springer 1993, pp. 197–236

[3] Ch. Pichler et. al., "Process Flow Representation within the VISTA Framework", *SISPED V*, Springer 1993, pp. 25–28.

[4] Ch. Pichler et. al., "TCAD Optimization Based on Task-Level Framework Services", *SISPED VI*, Springer 1995, pp. 70–73.

[5] G. Schumicki and P. Seegebrecht, *Prozeßtechnologie*, Springer 1991, pp. 370–380.

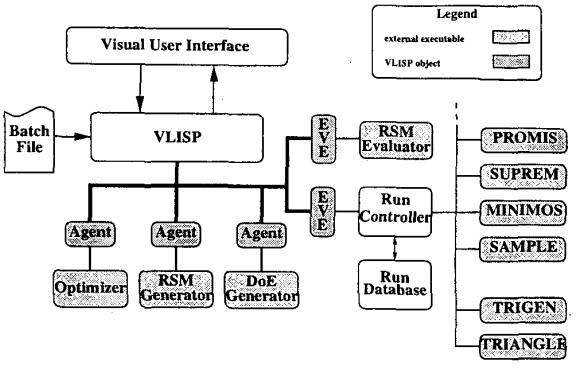

Figure 1: Overview of basic functional components. Evaluable entities (*EVE*) provide a uniform interface for computation requests, agents connect external services with the framework. Both agents and EVEs are VLISP objects.

```
;; use existing process flow for experiments and create EVE object

(setq eve (Define-Task "schumicki" :flow "~/processes/schumicki.sfe"))

;; define alle control and responses variables by specifying process step
;; name and variable in the process flow

(Eve-Define-Control eve "nw-dose" "*.N-WELL-IMPLANT" "dose")
(Eve-Define-Control eve "nw-energy" "*.N-WELL-IMPLANT" "energy")
(Eve-Define-Control eve "ldd-dose" "*.N-LDD-IMPLANT" "dose")
(Eve-Define-Control eve "ldd-energy" "*.N-LDD-IMPLANT" "energy")
(Eve-Define-Response eve "V-TH" "MINIMOS EXTRACT")
(Eve-Define-Response eve "I-SUB" "MINIMOS EXTRACT")

;; choose specific values for this analysis,

(Eve-Set-Control eve "nw-dose" 5e12 :min 1e12 :max 1e13 :trans 'log)
(Eve-Set-Control eve "nw-energy" 150 :min 120 :max 180)
(Eve-Set-Control eve "ldd-dose" 1e12 :min 7e11 :max 2e12 :trans 'log)
(Eve-Set-Control eve "ldd-energy" 100 :min 80 :max 120)

;; create half-factorial CCC design for EVE

(Create-Experiments eve :design 'CCC :cube 'HF)

;; start evaluation of experiments

(Build eve)
```

Figure 2: Batch file to define and submit high-level tasks. Control and response variables are selected from an existing process flow description. A set of experiments is generated by the DoE module and submitted to the run controller for execution.

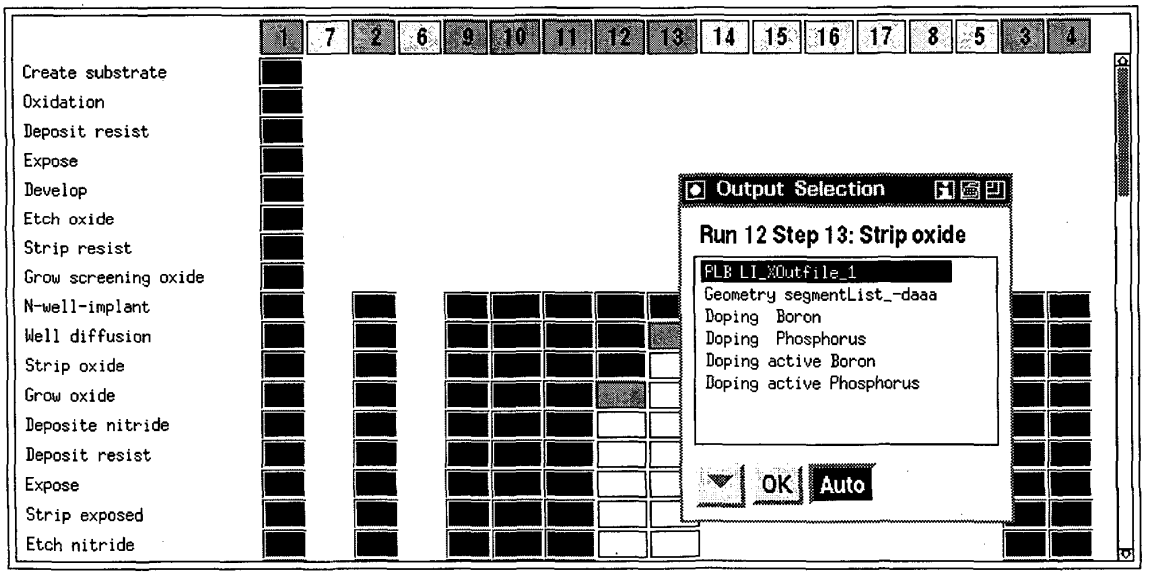

Figure 3: Part of the visual user interface during parallel, distributed computation of 17 runs generated via a half-factorial central-composite design for a complete CMOS process. 9 runs are active, while the remaining 8 are waiting for data to become available at split points. Direct access is provided to data generated at each step.

Acknowledgment. Our work is significantly supported by Austria Mikro Systeme AG, Unterpremstätten, Austria; Christian Doppler Gesellschaft, Vienna, Austria; Digital Equipment Corporation at Hudson, USA.