The Viennese TCAD System

S. Selberherr

and

F. Fasching, C. Fischer, S. Halama, H. Pimingstorfer, H. Read†,
H. Stippel, P. Verhas, K. Wimmer

Institute for Microelectronics, Technical University of Vienna, Austria
†ECE Department, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

A new TCAD system is presented which is capable of performing complex development tasks by means of a powerful interaction language and an efficient database system. The integration of tools is supported by implementing it as a layered product.

1 Introduction

The demands on technology CAD (TCAD) range from simple simulator coupling to process and device technology optimization, e.g. [4]. Our system as shown in Fig. 1 is controlled through an interaction language based on a LISP interpreter, providing homogeneous integration of the data, tool and task levels.

2 The PIF Database and the Application Interface

The format upon which the Viennese TCAD system is built is an enhanced and extended intertool mode of a widely used profile interchange format (PIF) proposed in [1]. The TCAD database is accessed from programs with the help of an application interface. Our implementation of this interface is strictly layered thus conforming to the most recent software engineering standards. A *system layer* at the very bottom is used to hide system specifics from the rest of the application interface. Thus the interface is open to all operating systems and not restricted to UNIX. A *caching layer* implementing a segment-buffer caching algorithm sits on top of the system layer, which significantly enhances access speed and memory utilization. The *basic layer* then is used to access primitive objects which hold primitive data types as well as efficiently compressed arrays. The *interface layer* deals with PIF objects which are made up of primitive objects. This interface layer is designed to work with C applications specifically designed for PIF. To provide a consistent interface to

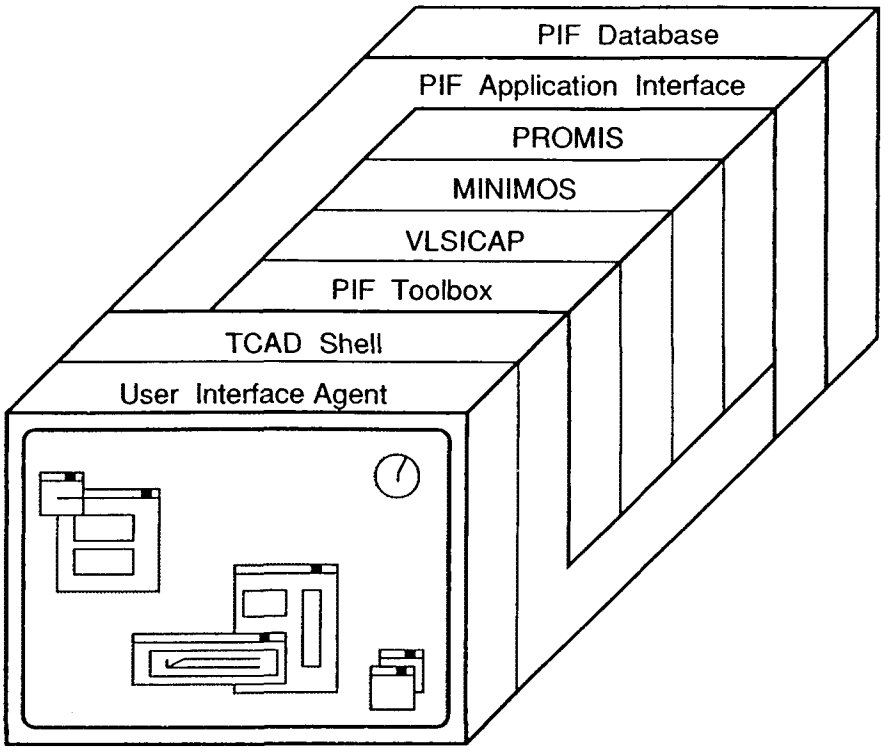

Figure 1: TCAD System Overview and Layer Structure

existing C and FORTRAN applications, an *application layer* has been designed which deals with all TCAD objects based on PIF. Therefore adding any new TCAD tool (simulator, measurement interpreter, correlator, etc.) is a simple and straightforward task.

3 TCAD Tools

Modules callable at the tool level include all kinds of simulators (process, device, interconnect), grid manipulators, discretizers, solvers, measurement data translators, optimizers, graphical editors, previewers, etc., some of which are incorporated in a PIF toolbox which also provides for intersite to intertool format conversion and vice versa.

Presently the process simulator PROMIS, e.g. [3], the device simulator MINIMOS, e.g. [5], and the interconnect capacitance simulator VLSICAP, e.g. [6], have been integrated. New tools can be added very easily by replacing input and output functions with corresponding application layer functions (see below). This small change yet allows data level integration into the TCAD system. However, the full power of the system can be exploited if the new simulator is provided with a shell language interface which allows the required data to be passed in form of PIF object handles. Additional flexibility can be gained by splitting the simulator (e.g. separating grid generation, discretization and solver parts) and by combining the new modules with existing TCAD tools into task level programs almost arbitrarily. To do so, the modules must adhere to a data format convention

[2]. The major advantage when building a new simulator is that it is no longer necessary to provide a specific grid generator, solver, etc., since these tools are readily available on the shell level. Therefore, simulator designers are able to concentrate on the specialized parts of simulator construction. The executable modules are usually small and can be run (in parallel) on different machines under control of the TCAD shell, thus yielding considerable speed improvement. When modularized appropriately, the most time consuming parts (e.g. linear solvers) can be executed on a supercomputer communicating with the TCAD shell running on a graphics workstation using our PIF linear solver communications protocol [2].

4 The TCAD Shell and the User Interface Agent

Modules or tools are directly callable as shell functions of the interaction language, thus enabling programs written in this language to call these tools. This structure allows arbitrarily complex tasks to be performed, ranging from simply calling a single module interactively over coupling simulators via a shell function to running whole optimization loops as background processes. Starting tools or shell functions on different machines is also possible (distributed processing). A major influence in the decision to use LISP as the interaction language is that there is no distinction between program and data structures. This allows, for example, a process flow representation to be either executed directly in the shell as a program, or to be stored in the database as data.

A powerful extension is the User Interface Agent (UIA) which allows graphical control of the TCAD system including editing, manipulating and viewing geometries, simulation results and symbolic process flow representations. In addition, the experienced user can directly use the shell language to create new functions or modify existing ones. The environment does not depend on the graphical interface which is inherently system dependent. It could as well be used without it (terminal capability is enough), although it is more convenient to use the UIA. Task level programs can be executed in both interactive and batch mode. An example of a task level program for minimizing the bulk current of a device by means of modifying the LDD implant dose is presented in Fig. 2.

5 Future Plans

Various popular simulators which have not been developed at our institute will be incorporated into our system. Further emphasis will be laid on distributed computing within the TCAD environment. Establishing a link to horizontal layout design will make the TCAD framework complete, allowing all activities to be performed in a homogeneous, highly flexible and expandable environment for the process and device engineer.

```
;;; sample TCAD shell task level function
(defun minimize-i-b (LDDimpl-dose obj-hdl)
   (run-minimos obj-hdl)
   (setq i-b-act
      (extract obj-hdl "BulkCurrent"))
   (do ((test-criterion i-b-act
         LDDimpl-dose obj-hdl))
      (setq LDDimpl_dose
         (compute-new-dose i-b-act
          LDDimpl-dose obj-hdl))
      (run-minimos obj-hdl)
      (setq i-b-act
         (extract obj-hdl "BulkCurrent"))))
;;; sample usage - edit a geometry
(setq obj-handle (ped))
; call the function
(minimize-i-bulk 5e12  obj-handle)
; the output i-bulk/i-drain [%]

0.01132
```

Figure 2: Sample TCAD function

Acknowledgement

Our work is sponsored by DIGITAL EQUIPMENT CORPORATION, Hudson – USA, SIEMENS CORPORATION, Munich – Germany, and SONY CORPORATION, Atsugi – JAPAN.

References

[1] S. Duvall, *An Interchange Format for Process and Device Simulation*, IEEE Trans. CAD, Vol. 7, pp. 489-500, 1988.

[2] F. Fasching *et al.*, *Viennese Integrated System for TCAD Applications*, Institute for Microelectronics, 1990.

[3] G. Hobler *et al.*, *RTA-Simulation with the 2D Process Simulator PROMIS*, Proc. NUPAD III, pp. 13-14, 1990.

[4] E.W. Scheckler *et al.*, *A Utility-Based Integrated Process Simulation System*, Symp. on VLSI Technology, pp. 97-98, 1990.

[5] S. Selberherr, *Three Dimensional Device Modeling with MINIMOS 5*, Proc. VLSI Workshop, pp. 40-41, 1989.

[6] F. Straker *et al.*, *Capacitance Computation for VLSI Structures*, Proc. EUROCON, pp. 602-608, 1986.