IMPLEMENTATION OF A TCAD FRAMEWORK

H. Stippel, F. Fasching, C. Fischer, S. Halama, H. Pimingstorfer,
W. Tuppa, K. Wimmer and S. Selberherr
Institute for Microelectronics, Technical University of Vienna,
Gußhausstraße 27–29, A-1040 Vienna, Austria

ABSTRACT

The integration of process and device simulators into modern Technology CAD (TCAD) systems has become a necessity because of the inherent communication, data transfer and maintainance advantages. Modern simulation programs and increasingly available computer resources turn complex tasks such as optimization loops into an affordable development tool. For the consecutive invocation and combination of different simulators a unique data format as well as a common control language is required. Moreover, integration opens up a variety of possibilities for the design of future simulators. We shall demonstrate how to make a stand-alone simulator work in a TCAD environment. After a short description of the basic components, the paper will focus on simulator integration; finally, some novel insights derived from the design of the whole framework will be highlighted.

GENERAL OVERVIEW

The main parts of the VISTA system (Viennese Integrated System for Technology CAD Applications [1]) are shown in Fig. 1.

VISTA consists of a *PIF Database*, which is an enhanced intertool version of the well-known profile interchange format proposed in [2]. To accomodate for the needs of existing TCAD applications, the original PIF syntax was restructured by reducing the number of different constructs, adding a few new constructs such as tensor product grid definition, and by defining additional semantical rules for the use of standardized attributes. Our PIF implementation is able to handle arbitrary LISP expressions for process flow representation; even whole TCAD shell programs can be stored consistently along with the core simulation data.

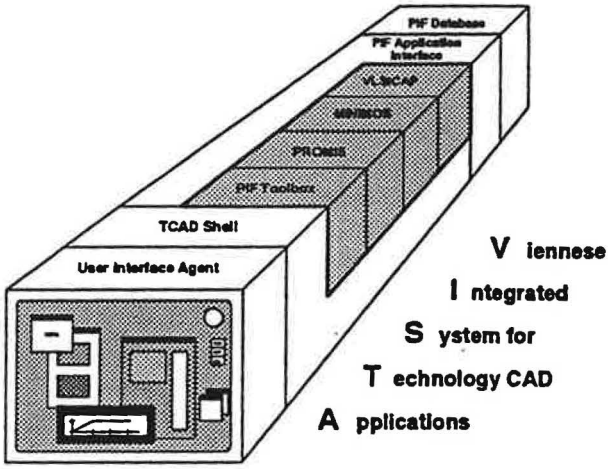

Figure 1: VISTA system overview

Simulators and other tools access the PIF database using the *PIF Application Interface* (PAI), which supports several languages including C, FORTRAN and LISP. The PAI is a procedural interface for accessing the binary PIF database. It provides an object-oriented functionality for creating, reading and modifying PIF objects. In this way the application programmer does not need to know too much about the PIF syntax to be able to use the PIF database. The PAI was designed as a strictly layered product to guarantee the necessary functionality, performance and extensibility.

A system layer hides all system dependencies concerning communication with the operating system from the rest of the PAI. On top of the system layer, the file layer deals with physical files and objects. The compression and the caching layer take care of performance and space requirements. The basic layer handles the structure of the information stored in the PIF file dealing with primitive objects, such as bytes, integers or reals. The interface layer allows access to the PIF objects suited for advanced C and is the

standardized interface to the PIF database. The application layer provides a more comfortable access to PIF objects for applications written in C, FORTRAN or LISP.

For the conversion from the binary intertool form to the intersite ASCII format the PIF binary file manager (PBFM) has been developed. In this way the intersite exchange of PIF files between different hardware and software platforms, for instance via electronic mail, is supported.

The *TCAD shell* is responsible for starting up the simulations. Time-consuming tasks are performed in the background or in batch mode. Additionally, an interpretive language must be used as the shell language to provide interactive control. Moreover, the TCAD shell is highly customizable and easily extensible to allow simple additions of new functionalities. We have chosen a LISP dialect as the extension language because of its flexibility and its independence from the operating system. XLISP [3] is written in portable C. For TCAD purposes, it has been extended by some functions (e.g. for the graphical user interface), which can be used in the same manner as built-in functions.

If the TCAD shell had been the front-end, the user would have had to learn LISP to perform simple simulations. To free the user from the constraint of learning LISP – like the PAI does for the simulation tool programmer who writes PIF access routines –, the *User Interface Agent* (UIA) has been designed. It provides dialog boxes, menus and various other objects based on so-called widgets for input deck assembly for various simulation tools. In this manner the user gets a comfortable point-and-click interface to his simulator and does not need to edit the input deck manually [4].

Considerable work in this field has also been done by other groups. A possible realization of a database can be seen in [5]. Aspects of user interfaces are discussed in e.g. [6], [7]. Approaches to general TCAD frameworks are presented in [8] and [9].

TOOL INTEGRATION

The integration of simulators into VISTA [10] can be achieved in two stages, namely the data level and the task level integration. This is illustrated in Fig. 2. The implementation of these stages is discussed in the following two sections.

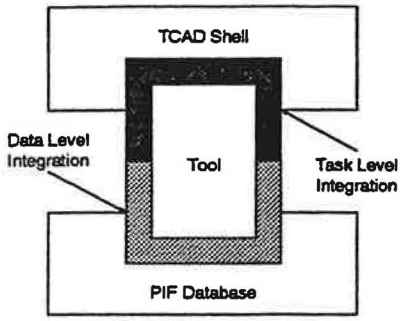

Figure 2: Tool Integration

Data Level

For the data level integration (see Fig. 3) the data input and output of a conventional simulator are replaced by access to the PIF database. This can be done in different ways, depending on the effort which one wishes to invest. One possibility is the direct replacement of the I/O functions by PAI functions in the simulator code. The other possibility is the generation of a wrapper, which converts the PIF input file to a conventional input deck, and the simulator output back to a PIF file. In any case the PAI is used as the standard access interface to the binary PIF database. For all simulator-independent data a physically motivated, predefined standard must be strictly obeyed. In this manner, common simulator tasks such as input parsing or normalization need not be executed by the simulator itself.

One would expect that the use of a *Procedural Interface* to an object oriented binary database would lead to an increased simulator I/O burden. We will see that this is not the case.

Another advantage of the use of the PIF data format is that a site using this data format standard does not need to write a data translation routine for the coupling of different simulators. Every tool using the PIF format as the underlying data structure can be coupled to other ones using only a PIF in- and output part to be able to communicate with other modules. Without a unique data format, n simulators would require $n \cdot (n-1)$ translators in the worst case!

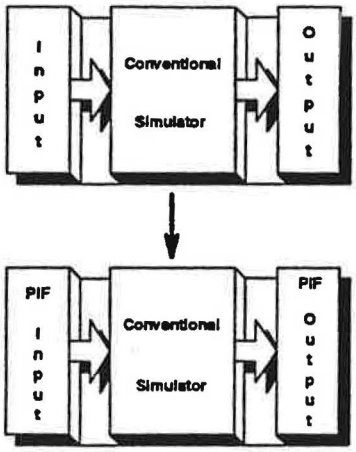

Figure 3: Data Level Integration

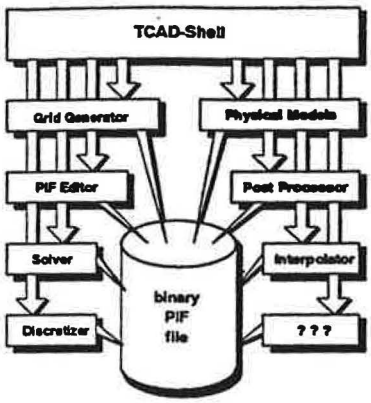

Figure 4: Task Level Integration

Task Level

In VISTA, the task level integration was designed to yield an even greater level of flexibility and functionality (see Fig. 4). For integration on this level, the simulator should be divided up into functional units, such as grid generators, discretizers, physical models or solvers. Since these units transform the PIF data from one well-defined state into another, they can be combined, for example via the TCAD shell. Every unit only needs to possess a standardized interface in order to be callable from LISP. This is the task level integration of the simulator. It has the advantage that after performing the necessary modularization, every functional unit can easily be maintained or replaced by a new one. Tools can then be directly linked to the TCAD shell, and, consequently, be true LISP functions. Alternatively, they can be separate executables which are only invoked by the TCAD shell.

Some of the modules which are derived from modularized simulators, as well as some lower level functions (for example *interpolate, add profiles, compute gradient, ...*), form the PIF ToolBox. This results in a collection of various functions ranging from simple initialization or arithmetic operations – such as adding two profiles defined over the same grid – to more complex modules like grid generators.

Building a new simulator from scratch usually requires a tremendous effort, whereas combining modules to create a new simulation instrument is a simple and straightforward task. The designer of a new simulator can focus on the physical problems, because one can combine new models with already existing tools of the toolbox.

The different options for integrating tools into the VISTA system shall now be exemplified using the in-house developed simulators, namely the process simulator PROMIS and the device simulator MINIMOS. At the moment there are several other simulators being integrated, i.e. the interconnect capacitance simulator VLSICAP [11] and the topography simulator SAMPLE [12].

PROMIS

The process simulator PROMIS [13] was integrated on the data level into VISTA. Before integration, the source code of PROMIS consisted of 251 routines which were partly independent from a semantic point of view, but not from the viewpoint of software architecture. During the integration into the VISTA environment, PROMIS was separated into several independent modules and all of them were turned into autonomous executables. This partitioning was physically motivated. Each part is now capable of simulating one specific process step (see Fig. 5).

Currently, modules exist for grid generation, analytical and Monte Carlo ion implantation into arbitrary structures, and for diffusion under inert and oxidizing conditions. After the partitioning of the si-

mulator, every executable was integrated on the data level into the VISTA system.

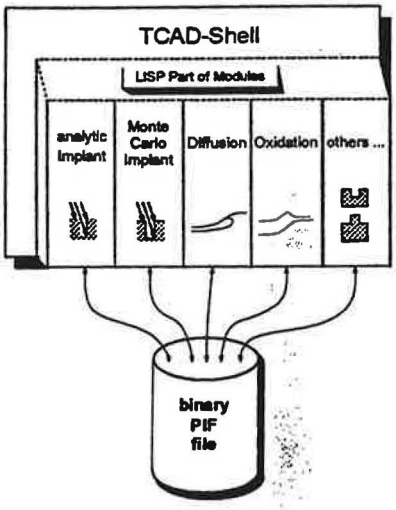

Figure 5: PROMIS in the VISTA system

The modules are linked to the TCAD shell by a small LISP part, which is responsible for preparing the PIF input file, for checking the input data for its integrity and finally for the invocation of the simulator modules. In this way, all the new executables may be seen as TCAD shell functions. Moreover, the communication between different PROMIS modules is now entirely based on the PIF database. Thus, it is very easy to replace and maintain the resulting parts of the process simulator independently.

Regarding only one separate executable, PROMIS can be seen as an example for the data level integration of a simulator. Exchanging one module with another one only implies invoking a new executable.

This new modularized concept, together with the programming capabilities of LISP, facilitate the definition of process flows without requiring any recompilation. Moreover, this shell implementation is highly suitable for customization, which is an indispensable prerequisite the rapidly changing demands of modern process simulation.

Contrary to what might be expected, code size was not increased by the use of the PAI. The code size of the resulting modules is about 1.4 MB instead of 1.5 MB. The code size needed for I/O could be decreased from 190 kB for the old PROMIS to 130 kB for the new PIF version. Additionally, the pass 1 of PROMIS which was responsible for checking certain semantical relationships was shifted to the TCAD shell which is much better suited for this task.

MINIMOS

The device simulator MINIMOS [14] serves as an example for the task level integration into the VISTA system. It has been totally separated into tools which perform elementary tasks, such as grid generation, equation solving, or the computation of physical parameters and coefficients (see Fig. 6). The consecutive invocation of these tools can be controlled by the TCAD shell, by a FORTRAN sequencing program, or otherwise.

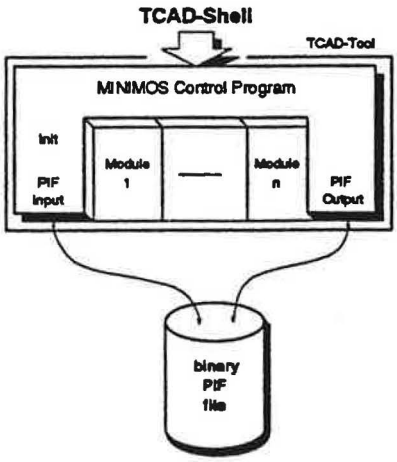

Figure 6: MINIMOS in the VISTA system

Every module of MINIMOS communicates with the other ones by accessing the PIF database. They read the required data from the PIF file and write the results back to it. This causes a very high data transfer rate. The results showed that this extensive use of the PAI did not reduce the efficiency of the simulator too much, because the total computation time was only increased by about 20 percent. The control program is currently implemented as a stack machine written in FORTRAN.

CONCLUSION

VISTA provides a set of simple basic functionalities as well as some high level tools. It gives the

user the possibility to get the very best out of his computer facilities through the use of its widely open concept.

The unique, highly standardized PIF data format serves as the base of the whole system, simplifying the data exchange. Using PIF as the underlying data format allows a simple coupling of different simulation tools.

Different integration levels can be achieved. Simulators can be incorporated into VISTA on the data or on the task level, depending on the amount of manpower invested in this coupling. Different tools support this integration optimally, resulting in an abstract tool description for an automatic binding to the TCAD shell.

ACKNOWLEDGEMENT

This project is supported by the laboratories of: AUSTRIAN INDUSTRIES – AMS at Unterpremstätten, Austria; DIGITAL EQUIPMENT Corporation at Hudson, USA; SIEMENS Corporation at Munich, Germany; and SONY Corporation at Atsugi, Japan.

REFERENCES

[1] F. Fasching, C. Fischer, S. Halama, H. Pimingstorfer, H. Read, S. Selberherr, H. Stippel, P. Verhas, K. Wimmer, *An Integrated Technology CAD Environment*, Proc. Int. Symp. on VLSI Technology, Systems and Applications, Taipei, Taiwan, pp. 147–151, 1991.

[2] S. Duvall, *An Interchange Format for Process and Device Simulation*, IEEE Trans. CAD, Vol. 7, No.7, pp. 489-500, July 1988.

[3] D. M. Betz, *XLISP, An Object-oriented Lisp*, Version 2.1, Peterborough, NH, Feb. 1988.

[4] S. Halama, F. Fasching, H. Pimingstorfer, W. Tuppa and S. Selberherr, *Consistent User Interface and Task Level Architecture of a TCAD System*, to be published in Proceedings to NUPAD IV, June 1992.

[5] C. H. Corbex, A. F. Gerodolle, S. P. Martin and A. R. Poncet *Data Structuring for Process and Device Simulations*, IEEE Trans. CAD, Vol. 7, No. 4, pp. 489–500, April 1988.

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

[7] P. A. Gough, M. K. Johnson, P. Walker and H. Hermans *an Integrated Device Design, Environment for Semiconductors*, IEEE Trans. CAD, Vol. 10, No. 6, pp. 808–821, June 1991.

[8] P. Lloyd, H. K. Dirks, E. J. Prendergast and K. Singhal, *Technology CAD for Competitive Products*, IEEE Trans. CAD, Vol. 9, No. 11, pp. 1209–1216, November 1990.

[9] A. Wong, W. Dietrich, and M. Karasick (Editors), *The SWR Architecture Document Version 0.2*, CAD Framework Initiative #162, Austin, TX, Mar 1991.

[10] S. Selberherr and F. Fasching, C. Fischer, S. Halama, H. Pimingstorfer, H. Read, H. Stippel, P. Verhas, K. Wimmer, *The Viennese TCAD System*, Proc. VPAD, Oiso, Japan, 1991.

[11] F. Straker, S. Selberherr, *Capacitance Computation for VLSI Structures*, Proc. EUROCON, pp. 602-608, 1986.

[12] A. R Neureuther, *IC Process Modeling and Topography Design*, IEEE Proceedings, special Issue on VLSI Design: Problems and Tools, Vol. 71, No. 1, pp. 121-128, January 1983.

[13] G. Hobler, S. Halama, K. Wimmer, S. Selberherr, H. Pötzl *RTA-Simulation with the 2D Process Simulator PROMIS*, Proc. NUPAD III, pp. 13-14, 1990.

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