A TCAD ENVIRONMENT FOR PROCESS AND DEVICE ENGINEERING

H. Pimingstorfer, S. Halama, and S. Selberherr

Institute for Microelectronics
Technical University of Vienna, Austria

Abstract — A new technology CAD environment is presented which integrates simulation tools and capabilities required in process and device development by using LISP as interaction and programming language. In TCAD shell functions LISP code is combined with functions for user interaction, data manipulation, simulator calls and visualization in an operating system independent manner. An X11-based graphical user interface is provided for convenient and intuitive control of simulation tools. By means of this integrated TCAD environment the user is able to concentrate on performing complex development tasks rather than on supervising single simulator runs.

INTRODUCTION

The demands on technology CAD (TCAD) range from simple simulator coupling over process and device characterization to technology optimization. Different kinds of tools are required: graphical editors, process, device, and interconnection simulators, parameter extractors, optimizers, postprocessors, etc. Integration of all these tools into a homogeneous TCAD environment is achieved by three key features: A common format for data exchange, a powerful shell language for extension and customization, and a graphical and interactive user interface.

SYSTEM OVERVIEW

An overview of our integrated TCAD system [1], [2] is given in Fig. 1. The data interchange format we have chosen is an enhanced and extended version of the well-known profile interchange format (PIF) proposed in [3]. The PIF database is accessed by programs by means of an application interface which supports the implementation languages C, FORTRAN and LISP.

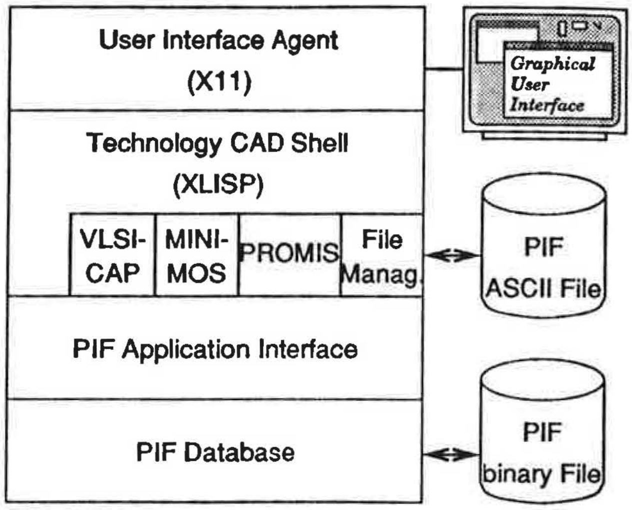

Figure 1: TCAD System Overview

TCAD SHELL LANGUAGE

On one hand the TCAD shell language is the command language with which the user interacts with the TCAD system; so it has to be interpreted. In addition, it must be able to run time consuming tasks as background processes or in batch mode. The other task of the TCAD shell language is to serve as an extension language, in which new functionality is added, customizations are specified, and macros or just shortcuts for frequently used shell command sequences are defined.

We have chosen LISP as the base of the TCAD shell because of the flexibility of this programming language and its independence of the operating system. Among the candidates of publicly available interpreters we picked XLISP [4] in its current version 2.1. This small interpreter is written in portable C with modularized design and exhibits a clear C-to-LISP interface.

The source code availability meets the need to implement TCAD shell functions in C, which are linked together with the original interpreter and are further handled like built-in functions.

TOOL INTEGRATION

Tools can be integrated in three ways, depending on the language they are programmed in:

- LISP tools just have to be loaded and executed by the TCAD shell. This is useful for high-level optimization loops or module sequencers, which consume only small amounts of the overall computation time.
- Tools in form of a C function just have to get a small C-to-LISP interface. Then they can be linked together with the shell and called just like normal built-in shell functions. This is useful for small and frequently needed tools which consume some computation time. They could as well be called as separate executables by a system call, but linking them to the shell eliminates the operating system overhead.
- Tools in any language that are separate executables can be called with a shell built-in system call function. Thus existing simulators, mostly coded in FORTRAN, can be used like any other shell function.

Until now the device simulator MINIMOS, e.g. [5], the process simulator PROMIS, e.g. [6], and the interconnect capacitance simulator VLSICAP, e.g. [7], have been integrated into the TCAD system.

SHELL FUNCTIONS

Shell functions specialized on MOS transistors for example, compute the threshold voltage and drain and/ or bulk current by invoking MINIMOS and returning the value of interest as a LISP expression. These functions combined with a one-dimensional optimizer are used, for instance, to find the maximum of the bulk current or of the relative transconductance. Combined with looping constructs, the shell functions are tailored to compute I/V characteristics or any other variation of an output quantity versus any allowed input key, applying a constant or an adaptive step size.

For each simulator run, the user is relieved from modifying an input deck with an editor, starting the simulator on the command line and getting the required values from the simulator output.

With few lines of TCAD shell code a new shell function, tailored to the very specific needs of the user, can be written as a combination of any tool callable at shell level and normal LISP code. The TCAD shell allows arbitrarily complex tasks to be performed, ranging from simply calling a single module interactively over coupling simulators to running whole optimization loops as background processes.

USER INTERFACE

The TCAD shell serves as a textual user interface to the TCAD system in cases where terminal capability is required to be enough. For higher convenience, the User Interface Agent (UIA) has been designed which allows graphical control of the TCAD system.

An interface to the X11R4 window system has been implemented as part of the LISP interpreter, based on X Toolkit, Intrinsics and Athena widgets to address the portability issue between workstations from different vendors. Specialized widgets have been added to provide for comfortable specification of numerical values, file selection and vector graphics capabilities. In principle, widget callbacks cause LISP expressions to be evaluated. The flexibility gained by combining X11 with a LISP interpreter enables the system to accommodate to the very specific needs and peculiarities of existing environments and applications. Due to the interactive nature of the construction of the user interface, the actual layout and major parts of the functionality are subject to customization and configuration.

For example, to support the creation of user interfaces for simulator features and any kind of TCAD shell functions, a utility for automatic generation of interaction panels (consisting of a set of widgets and callback functions) from a brief abstract description (in LISP syntax) was implemented. Some of these panels are shown in Fig. 2. They are used for specifying input parameters and starting a tool execution. Default values are supplied wherever possible. Each parameter modification is immediately checked, thus detecting simple but most often made input errors like leaving the allowed value range or selecting a badly interfering set of choice parameters as soon as possible and not only when a tool fails to execute.

Another example for the advantageous combination of X11 and LISP is the hierarchically organized, interactive help system which retrieves the information it

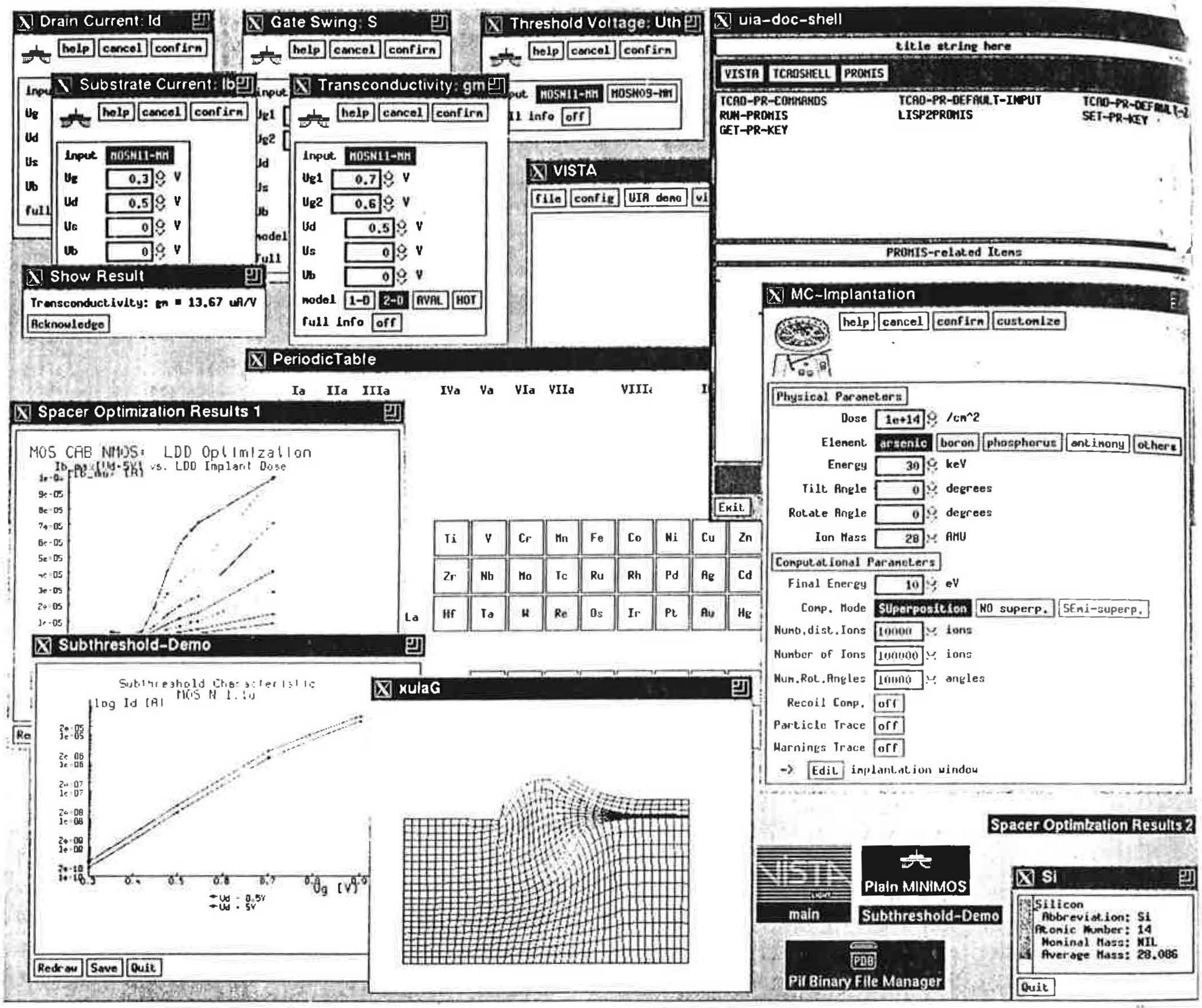

Figure 2: Graphical User Interface

needs out of special comments within the code at load time. Therefore the help system is guaranteed to be consistent with the actually loaded versions of all functions.

EXAMPLE

As an example, the bulk current of an n-channel MOS transistor has to be minimized by varying the dose of the lightly doped drain (LDD) implant. For this purpose process and device simulation are coupled within an optimization loop. The doping profiles simulated by PROMIS are characterized by several MINIMOS runs, computing the threshold voltage, the saturation current, and the maxima of the relative transconductance in the linear regime and of the bulk current. The ratio of bulk to drain current for a constant bias condition is the value to drive the optimization loop for the LDD implant dose.

The TCAD shell program is shown in Fig. 3. In a concrete application the one-dimensional optimizer needed 8 iterations to explore a dose range from 10^{10} to $10^{16} cm^{-2}$ till the quotient of two consecutive dose values was less than 2. That means, 8 times PROMIS and due to the extensive device characterization about 100 times MINIMOS ran automatically under control of the TCAD shell.

```
; optimize LDD impl. dose for minimal ib/id[2.5/5.0]
; run PROMIS and characterize the profile running
; MINIMOS for U_th, Id_saturation, gm_max, Ib_max,
; and Ib/Id for each optimizer iteration
;
(defun minimize-ib/id
  (PR-INPUT MM-INPUT DIRECTIVE OCCUR KEY MIN MAX TOL
    &key (PR-BASENAME TCAD-PR-TFN) (LOG NIL)
    &aux PROFILE  RESULT-LIST  Ib/Id-VALUE)
  ;;1d-optimizer
  (golden-section
    #'(lambda (VALUE)
      ;;new profile name
      (setq PROFILE (new-profile-name PR-BASENAME
                          DIRECTIVE OCCUR KEY VALUE))
      ;;modify PROMIS input deck
      (set-pr-key PR-INPUT DIRECTIVE KEY VALUE OCCUR)
      ;;run PROMIS
      (run-promis PR-INPUT :EXEC-MODE "i")
      ;; run MINIMOS several times
      (setq RESULT-LIST (append RESULT-LIST
        (list (list VALUE
          (u-th      MM-INPUT :PROFILE PROFILE)
          (id[bias] MM-INPUT
                    :UG 5.0 :UD 5.0 :PROFILE PROFILE)
          (gm-max   MM-INPUT 1.0 3.0 0.2
                    :UD 0.1 :PROFILE PROFILE)
          (ib-max   MM-INPUT 1.0 3.0 0.2
                    :UD 5.0 :PROFILE PROFILE)
          (setq Ib/Id-VALUE (ib/id[bias] MM-INPUT
                :UG 2.5 :UD 5.0 :PROFILE PROFILE)))))
      Ib/Id-VALUE); end lambda
    MIN MAX TOL :LOG LOG); end golden-section
  ;; return result list
  RESULT-LIST)
```

Figure 3: Example TCAD Shell Program

CONCLUSION

The use of LISP as top-level implementation language for our TCAD system opens up new possibilities in terms of flexibility and customization. Very complex development tasks can be performed by utilizing the interfaces to simulation tools and the database together with the programming language features of LISP. In contradiction with common expectations, a relatively small percentage of the computation time is spent inside the LISP interpreter. Generally spoken, the interpreter is busy with the task of redirecting and responding to events coming from the X11 system and controlling the execution of simulation tools. Hence, the overall system performance does mainly depend on the performance of the simulation tools themselves.

ACKNOWLEDGEMENTS

This project is supported by the research laboratories of: AUSTRIAN INDUSTRIES - AMS Int. at Unterpremstätten, Austria; DIGITAL EQUIPMENT Corp. at Hudson, USA; SIEMENS Corp. at Munich, FRG; and SONY Corp. at Atsugi, Japan.

REFERENCES

[1] S. Selberherr *et al.*, *The Viennese TCAD System*, Proc. Int. Workshop on VLSI Process and Device Modeling, pp. 32-35, Oiso, Japan 1991

[2] F. Fasching *et al.*, *An Integrated Technology CAD Environment*, Proc. Int. Symp. on VLSI Technology, Systems and Applications, pp. 147-151, Taipei, Taiwan, 1991.

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

[4] D. M. Betz, *XLISP: An Object-oriented Lisp*, Version 2.0, Peterborough, NH, Febr. 1988.

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

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

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