INVITED PAPER

Technology CAD for Smart Power Devices

T. Simlinger, Ch. Pichler, R. Plasun, and S. Selberherr

Institute for Microelectronics, TU-Vienna
Gußhausstraße 27–29, A–1040 Vienna, Austria
e-mail: simlinger@iue.tuwien.ac.at, URL: http://www.iue.tuwien.ac.at

Summary

Technology CAD has already proven to be an attractive supplement to standard process development methodologies for VLSI technology. The Vienna Integrated System for TCAD Applications is such a simulation environment which offers great flexibility and a large number of tools for process development. On the other hand smart power devices gain increasingly interest for applications which need to combine low voltage logic and power output devices as, e.g, for automotive electronics. The flexibility of TCAD makes it also applicable for smart power technology to cut down development costs and cycle times.

1 Introduction

In recent years smart power technology has become a standard for various applications such as for automotive electronics. Its ability to integrate low voltage logic, low voltage analog circuits, and power output devices on a single chip in combination with low cost and robustness makes smart power technology attractive for a large number of applications. Integration of low voltage and power circuits offers a flexibility which allows for solutions tailor made to a specific problem.

The integration of devices and circuits with concurrent requirements on process technology which exhibit low voltage logic and power output devices gains problems difficult to solve for even a few specific applications. The "optimum" technology will be hard to define even if it might exist at all.

Altogether these facts make simulation very attractive to cut down development costs and reduce system design cycle times significantly. The development of simulation environments which allow for the simulation of a complete process rather than a specific process detail has come to a point where such environments are an attractive supplement or even alternative to standard process development methodology. The Vienna Integrated System for TCAD Applications (VISTA) [1] is a simulation environment which comprises tools for simulating the various processes which are, i.e., required for smart power technology and offers the benefit of optimizing critical parameters and investigating numerous process variants at low costs.

This article focuses on the specific aspects which make VISTA suitable for smart power device development. The applicability of VISTA for smart power technology is demonstrated on the simulation and optimization of a vertical DMOS transistor (VDMOS transistor) process. This example includes simulation of several process steps such as epitaxy, ion implantation, annealing, and etching as well as device simulation to determine the electrical characteristics (on-resistance, break-down voltage) of the processed device for optimization.

The VDMOS transistor is a commonly used power device for smart power applications. The example is designed for $V_{DS} = 100$ V and an on-resistance $R_{DS(on)} = 0.9\ \Omega$. The optimization gives the thickness and doping concentration of the transistors epi-layer such that the on-resistance reaches a minimum. The obtained process parameter values compare very well to the values of a corresponding process already in use.

The next two sections describe the structure of VISTA and the optimization module in more detail followed by the example section. In the final section some conclusions are drawn and an outview is given on the further development of VISTA.

2 The VISTA/SFC Simulation Environment

With shrinking device dimensions and decreasing product-development cycles, fully-automated TCAD analysis of complete semiconductor processes and devices is becoming increasingly important. The Vienna Integrated System for TCAD Applications (VISTA) and its Simulation Flow Control (SFC) module form a programmable simulation environment for VLSI technology analysis, focusing on heterogeneous tool integration, process flow simulation, and high-level task support including response surface modeling (RSM) and optimization. Based on process and device simulation capabilities with a variety of simulation tools, split-lot experiments can be defined for fabrication process flows and simulation sequences. The parallel and distributed execution of independent split tree branches allow a fast computation of large-scale experiments. A persistent run data base keeps all simulation results and prevents unnecessary recomputations. 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. The combination of a comfortable, intuitive visual user interface with the flexibility and versatility of a high-level programming language for TCAD applications results in a powerful tool for TCAD integration, development, and production use.

2.1 Architecture and Components

Figure 1 gives an overview of the principal components of the VISTA/SFC TCAD environment. As the central coordinating instance, the *task control* layertakes care of controlling all activities initiated via the GUI, the ASCII interface, or a batch file. It establishes object-oriented interfaces for all task-level services. The VLISP *shell interpreter* [2] – not shown in the figure – provides the basis for the implementation of all other internal modules. It provides interfaces to the operating system, the graphical user interface (GUI), and the PIF Application Interface(PAI) [3] to conveniently access simulation data stored in the Profile Interchange Format (PIF) [4]. All operating-system dependent services are encapsulated by the VLISP interpreter, which ensures portability over a wide variety of operating systems and platforms.

The *flow editor* (Figure 2) offers an intuitive and convenient graphical interface for writing process flows. It supports the definition of process flows in a hierarchical and modular manner in terms of tool-independent process statements as well as explicit tool statements. Process flow information is interpreted by the *run controller*, which together with the *run data base* forms the core components for the management of iterative and parallel split-lot experiments. The run controller takes care of the detection of splits, of scheduling multiple runs in parallel operation on workstation clusters, and offers a number of operation modes to facilitate development and debugging of both processes and simulation tools. The run data base stores and retrieves simulation output data and extracted data of any format, with the PIF format being used as primary exchange format for wafer data.

All simulation tools and auxiliary data-manipulation tools are accessed through a layer of *binding functions* that encapsulate tool specifica, establishing a set of VLISP functions to allow the invocation of all simulation tools in a uniform manner as simple function calls. All tool parameters as well as output and error redirections are passed as arguments to the binding function, ensuring a maximum of independence of the tool binding layer and the task control layer. All binding functions together are organized as the *tool application layer* that makes all tools available to the task control layer.

At the other end of the functional hierarchy, task-level tools are interfaced with the task control layer by interface agents which establish communication channels with concurrent executables like optimizers, design-of-experiment (DoE) tools [5], and response-surface-modeling (RSM) modules [6]. While being run as clients of the task control layer, they also operate as servers for more complex applications, and are linked together by callback connections.

Figure 3 shows the graphical user interface of the VISTA/SFC simulation environment. The topmost window contains the graphical user interface of the run data base and run controller modules. The process steps of the selected process flow are listed on the left side, the split tree of simulation runs in the current project provides direct access to all data of all computed steps and gives a quick summary of the activity states of all simulations. The *System Jobs* window shows all active and queued system jobs started on behalf of the run controller. The *Experiment Table* window contains a spreadsheet representation of all simulation runs and control and response variables defined for the process flow. Task-level tools like DoE and RSM generation are directly accessible. The *Hosts* window displays the busy-state of all network hosts used for submitting system jobs.

2.2 Task Encapsulation

When focusing on the system responses of process flows and devices, all simulation-related concerns should not be dealt with, but rather be handed down to some service that delivers the requested results. Furthermore, no distinction should be made with respect to the kind of procedure to invoke to get these results. More precisely, task-level applications should be liberated as far as possible from subtleties

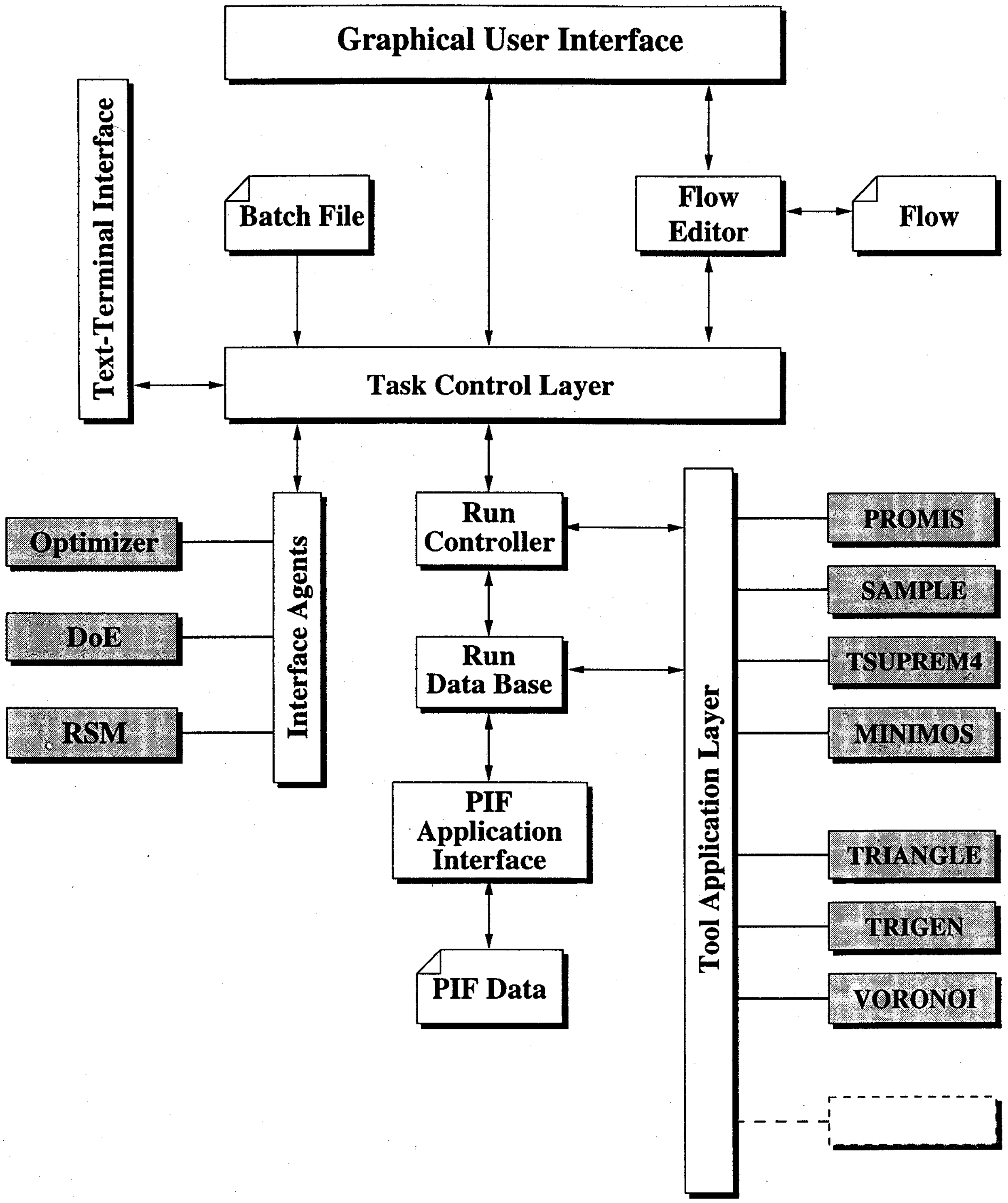
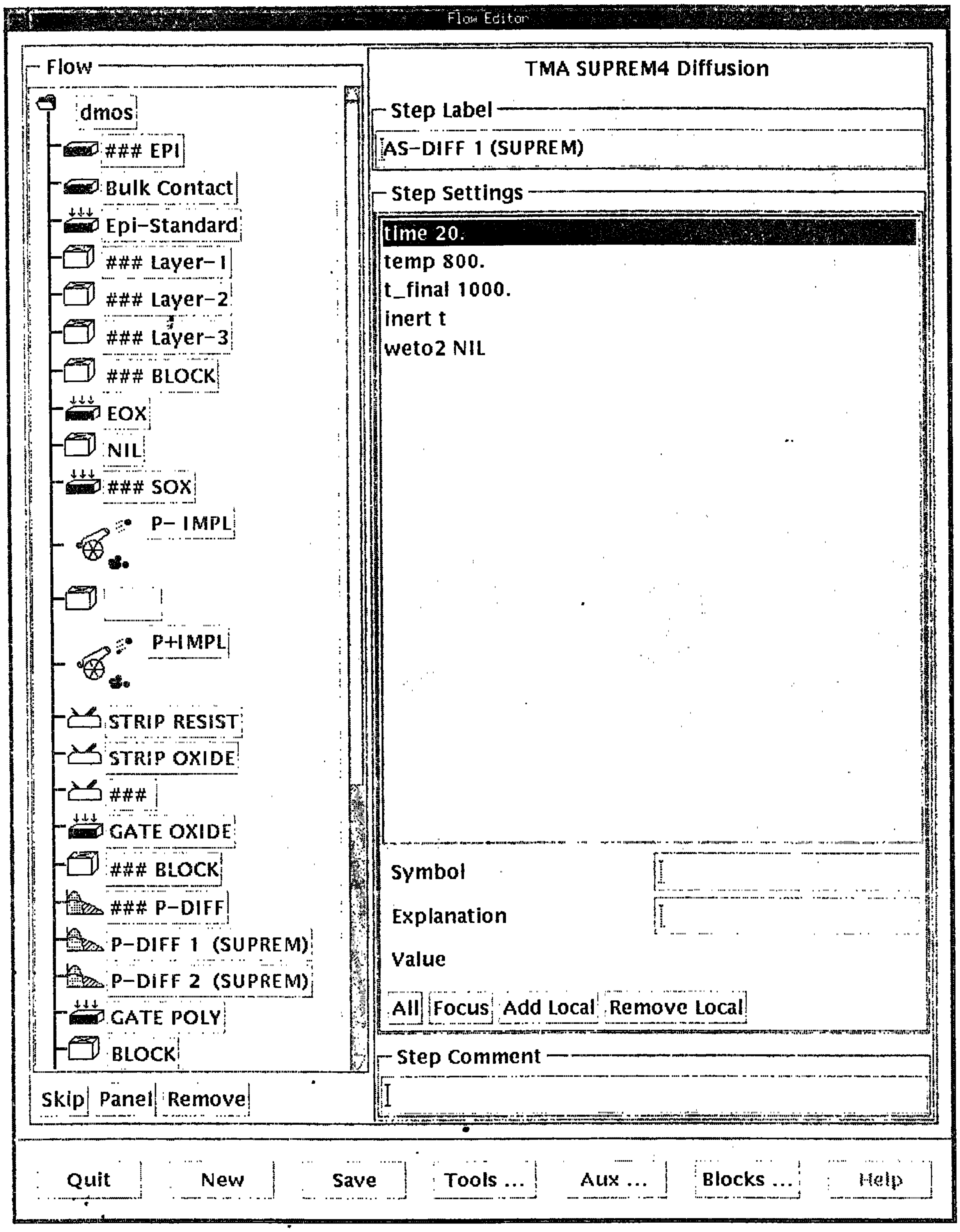


Figure 1: Main components of the VISTA/SFC TCAD environment. Dark rectangles indicate external executables.

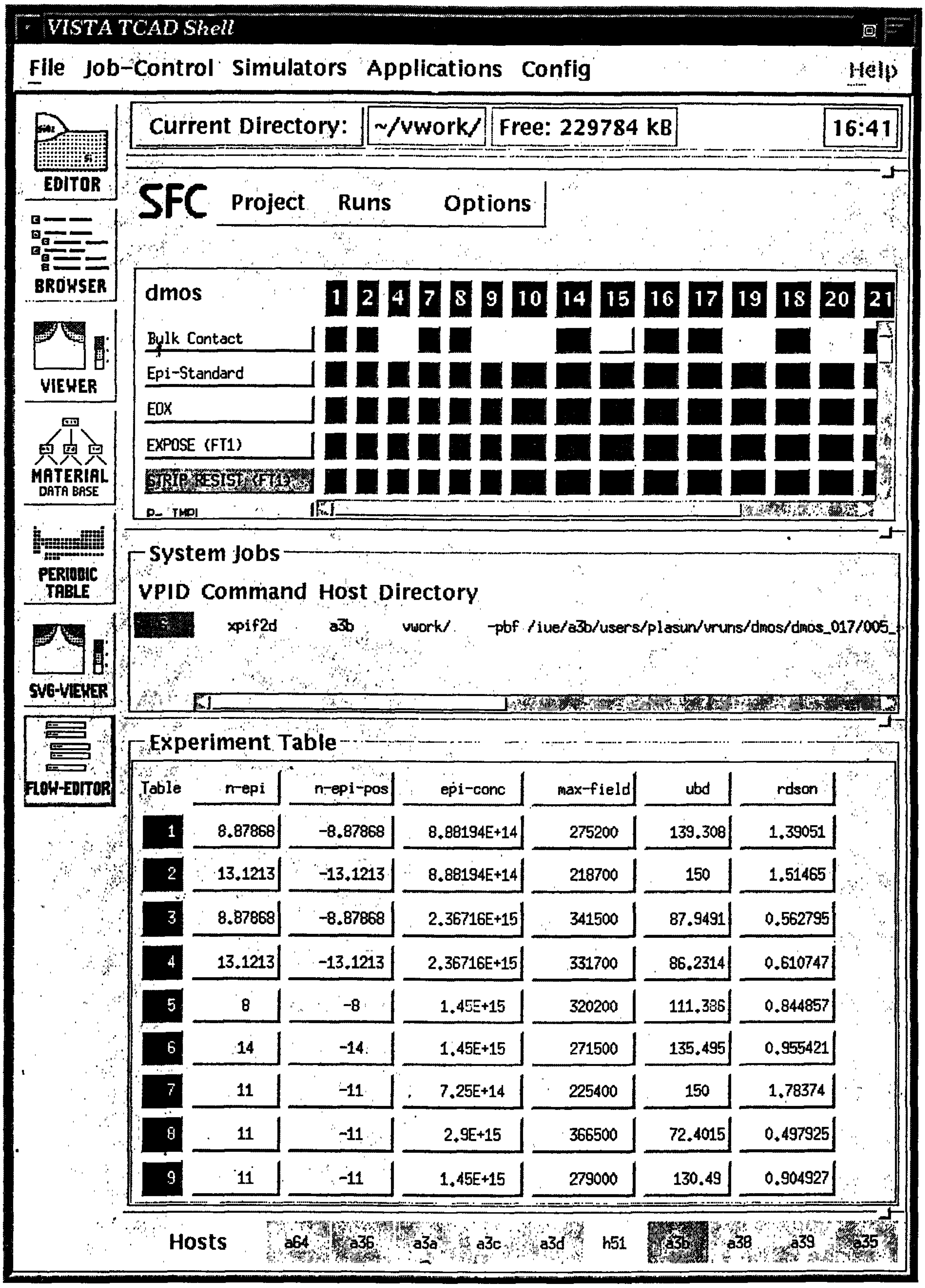


dmos
EPI
Bulk Contact
Epi-Standard
Layer-1
Layer-2
Layer-3
BLOCK
EOX
NIL
SOX
P- IMPL
P+IMPL
STRIP RESIST
STRIP OXIDE
###
GATE OXIDE
BLOCK
P-DIFF
P-DIFF 1 (SUPREM)
P-DIFF 2 (SUPREM)
GATE POLY
BLOCK

AS-DIFF 1 (SUPREM)

```
time 20.
temp 800.
t_final 1000.
inert t
weto2 NIL
```

Figure 2: The process simulation flow for the VDMOS transistor. Individual steps are clearly shown and can be easily modified.



Current Directory: ~/vwork/ Free: 229784 kB 16:41

dmos	1	2	4	7	8	9	10	14	15	16	17	19	18	20	21
Bulk Contact	■	■		■	■			■	□	■	■		■		■
Epi-Standard	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EOX	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EXPOSE (FT1)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
STRIP RESIST (FT1)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

System Jobs

```
VPID Command Host Directory
     xpif2d  a3b  vwork/  -pbf /iue/a3b/users/plasun/vruns/dmos/dmos_017/005_
```

Experiment Table

Table	n-epi	n-epi-pos	epi-conc	max-field	ubd	rdson
1	8.87868	-8.87868	8.88194E+14	275200	139.308	1.39051
2	13.1213	-13.1213	8.88194E+14	218700	150	1.51465
3	8.87868	-8.87868	2.36716E+15	341500	87.9491	0.562795
4	13.1213	-13.1213	2.36716E+15	331700	86.2314	0.610747
5	8	-8	1.45E+15	320200	111.386	0.844857
6	14	-14	1.45E+15	271500	135.495	0.955421
7	11	-11	7.25E+14	225400	150	1.78374
8	11	-11	2.9E+15	366500	72.4015	0.497925
9	11	-11	1.45E+15	279000	130.49	0.904927

Hosts a64 a36 a3a a3c a3d h51 a3b a38 a39 a35

Figure 3: VISTA/SFC graphical user interface.

regarding the invocation of simulators, the precise format of input decks, the proper interpretation of generated files and return values, etc. To this end, the VISTA/SFC simulation environment provides a class of objects that encapsulate all evaluation tasks on the task level and establish a uniform interface between task-level modules such as optimizers, response surface models, database queries, etc.

2.3 Evaluable Entities

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 more complex, user-defined tasks. For example, the minimization of the on-resistance of a VDMOS transistor for a given process can be encapsulated in an object that is evaluated for a set of initial value vectors for the optimizer generated by a DoE module or by a simple LISP loop. Figure 4 shows the basic idea of an EVE object representing a process flow simulation task. In order to avoid any ambiguity, the left side of the EVE object in Figure 4 is called *client side*, the right one *server side*.

After specifying those process parameters and measurements which are to be used as input variables (controls) and output variables (responses), respectively, for subsequent analysis tasks, the EVE object hides all evaluation details of the underlying simulation.

An EVE object offers a set of basic functions that can be invoked uniformly across all classes of EVE objects. Table 1 gives a summary of their names together with short explanations.

The `define-control` and `define-response` methods of an EVE object usually are invoked only once to establish a link with the server-side module. In general, a single control variable can have any number of connections to the server side. This mechanism is particularly useful when encapsulating process flows to map a single control variable to all appearances of a certain class of process or simulation parameter.

3 High Level Optimization Capabilities

Optimization problems can be divided into two groups:

- The first one contains the optimization of extracted physical parameters in a given set of input parameter space. Direct optimization methods are not very well suited for this problem because in order to calculate one set of input and output values a whole process flow and usually several device tests must be simulated. To save CPU-time, methods like Design of Experiments (DoE) and Response Surface Methodology (RSM) have to be used.

  The second class of problems are concerned with the calibration of simulator modules. However, in this case only a single simulator – eventually with pre- and postprocessing – needs to be executed. Direct optimization methods work very well for this task.

This article will concentrate on the first type of problem.

3.1 Design of Experiments

The EVE object can hold additional information for each process parameter, e.g., default values and value ranges. For dependence analysis of controls and responses these data are used by the DoE module to generate a set of experiments [5]. The type of the experimental design can be chosen out of a large number of available types (Table 2). For the generation of a RSM a *Central Composite* design is very common or for a simpler analysis a *Screening Analysis* can be used. To probe the parameter space in the most efficient way, transformations for the parameters can be defined; see Section 3.3 for details. From the results of the DoE module — the *design matrix* — the SFC generates the split tree so that no duplicated process steps are calculated.

After simulating these runs the control and the response values are written to the *experiment table* and can be processed by other tools.

Table 2: Experimental designs

NOM	Nominal Design
SA	Screening Analysis
FUL	Full Factorial Design
CCF	Central Composite Facecentered Design
CCC	Central Composite Circumscribed Design
CCI	Central Composite Inscribed Design
RAN	Random Design
DIA	Diagonal Design
GRI	2D - Grid Design
LAT	Latin Hypercube Design
FRA	Fractional Factorial Design
PLA	Plackett-Burman Design
OME	Orthogonal Main Effect Design
SUP	Supplementary Design

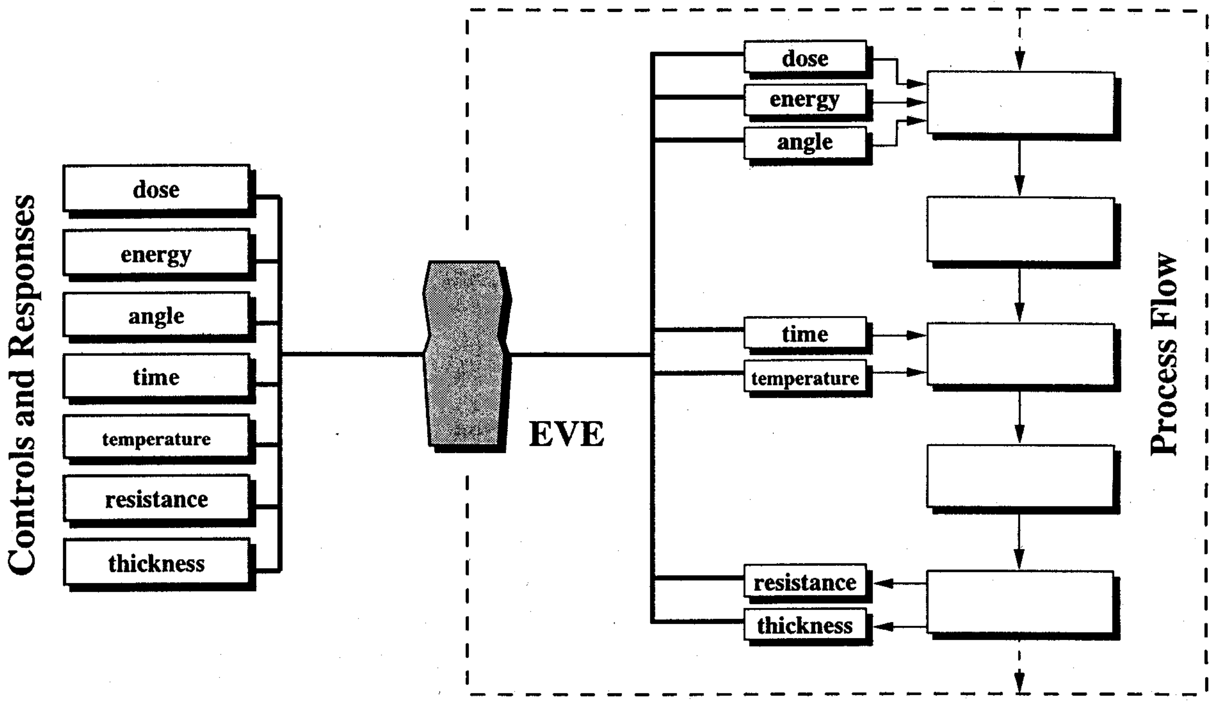


Figure 4: Process flow encapsulation with evaluable entity (EVE) objects.

```
(Eve-Define-Control "alpha")

(Eve-Define-Control "beta" :eval-expr '(if (plusp alpha) (sqrt alpha) 0.)
                           :internal T)

(Eve-Define-Control "gamma" :conversion '(string gamma))
```

Figure 5: Three examples of the command `Define-Control` for defining control variables. `alpha` is declared as a control variable. `beta`'s value is derived from a LISP expression using `alpha`. The `internal` attribute marks the `beta` to be inaccessible to the client side. `gamma`'s value is converted to a string on the server side, but not on the client side.

Table 1: Summary of basic EVE functionality. For the sake of conciseness, operations that are equivalent for both controls and responses are not listed separately.

Symbolic Name	Description
`define-control`	Defines a new control variable. All variables are identified by case-insensitive names that are unique in the context of an EVE object. Used to define the position of a process parameter in a process flow or to select one out of a number of available parameters.
`set-control`	Defines a default value and ranges for a control variable. Additionally, a control variable may be marked as `internal` if it is not to be accessed from the client side.
`set-control-expression`	Defines a LISP expression used to derive a control variable's value from other control variables (Figure 5).
`set-control-conversion`	Defines a LISP expression used to convert a control variable's value to a different representation before handing down to the server side.
`eval`	Request the evaluation of the underlying model for a given set of control values. If the set of control values is not complete, missing values are taken from the default values of the respective control variables. After termination of the evaluation, the responses are returned in a callback.

3.2 Response Surface Methodology

Given a set of simulated data points in the *experiment table*, the RSM module [6] is used to generate polynomial functions that provide an analytical representation of the data. As in the DoE module, additional transformations for the controls and the responses can be added; see Section 3.3 for details.

With an RSM-viewer the fitted function can be visualized as two- or three-dimensional graphics (e.g., Figure 7). Sliders for the independent variables are available to set parameter values and to help the user in understanding the influence of the variables on the response.

3.3 Transformations

To accurately model the system behavior, both the DoE and RSM modules make use of transformations of the parameter space to linearize the dependence of the output variables on the transformed input parameters. Subdivision of the parameter space as well as fitting of the response surfaces takes place in transformed space.

For each input parameter, a transformation function can be selected from a set of well-known transformations. If the transformation function needs parameters (*transformation parameters*), these parameters may either be specified explicitly - e.g., in the case when a physical formula has been established, or they may be determined automatically from a set of sample points. Additionally, it is also possible to select the best one of a given set of transformation functions for a given set of sample points. Thus, the user does not need to specify the transformation.

It is important to note that all transformation functions have to be defined by specifying code for both the forward and reverse directions and assigning a reference name to the transformation before they can be used. All information on transformations is stored centrally and accessed exclusively by the reference name. E.g., for the example shown in the application section a transformation `epi-conc` is defined which analytically reflects the logarithmic dependence of the on-resistance on the epi-doping concentration and, thus, linearize the problem for DoE and RSM.

3.4 Optimizer

For optimizing device performance parameters over a given input variable space a number of optimization algorithms have been implemented. The communication between optimization program and framework is handled by an optimizer agent. This agent handles a simple standardized communication protocol so other external optimizers can be integrated easily.

For standard applications a constrained optimizer with sequential quadratic approximations is used. The gradient is calculated by evaluating finite differences and the hessian is built by a BFGS update. The optimizer minimizes the target function which is constructed by the optimizer agent regarding the specified input and output value sets.

An optimizer for nonlinear calibration problems is also available. It is based on the Levenberg Marquardt algorithm [7]. Again, the gradient is calculated by finite differences and the hessian is built by BFGS updates.

3.5 Architecture

The DoE module, the RSM module and the optimizer are external tools. The optimizer and the RSM module are concurrent programs that run simultaneously with the framework, so special interface agents handle communication between the framework and the tool. These agents are implemented as object classes; by subclassing, new classes can be derived from existing ones to simplify the integration of additional tools. Thus, the existing implementation of a tool acts as an template and only a few settings specifically for the tool to integrate must be changed or added.

4 Application

In this section the processing of a vertical DMOS (VDMOS) transistor is described; it is shown how all the functions mentioned above work together. VDMOS transistors are commonly used power devices for smart power technology for voltages up to 100 V. Low on-resistance as well as smaller lateral size make them superior to lateral DMOS transistors and make up the more complex processing. Figure 8 depicts the schematic structure of a VDMOS transistor.

At first, as described in the previous section, a simulation flow representation is created using the SFE. Regarding to a certain process step one or more simulation steps must be chosen. For each simulation step the desired parameters are set. The SFE panel with the simulation flow representation for the VDMOS transistor process is given in Figure 2. The left column of the panel shows the tree representation of the simulation flow. The actually selected step is the first diffusion of the source contact doping. The parameter settings are shown in "Step Settings" window. The annealing time is 20 min. where the temperature is linearly ramped up from 800 to 1000° C. Figure 6 depicts the topmost part of the transistor structure and the netdoping as the result of a typical run.

The goal of the example is to make the on-resistance as low as possible, but hold the breakdown voltage above a fixed value (> 105 V). The control variables are the epi-doping and epi-layer thickness (Table 3).

Table 3: Range of the control variables.

Parameter	min.	max.	Unit
Epi doping	0.725e15	2.9e15	cm^{-3}
Epi thickness	8	14	μm

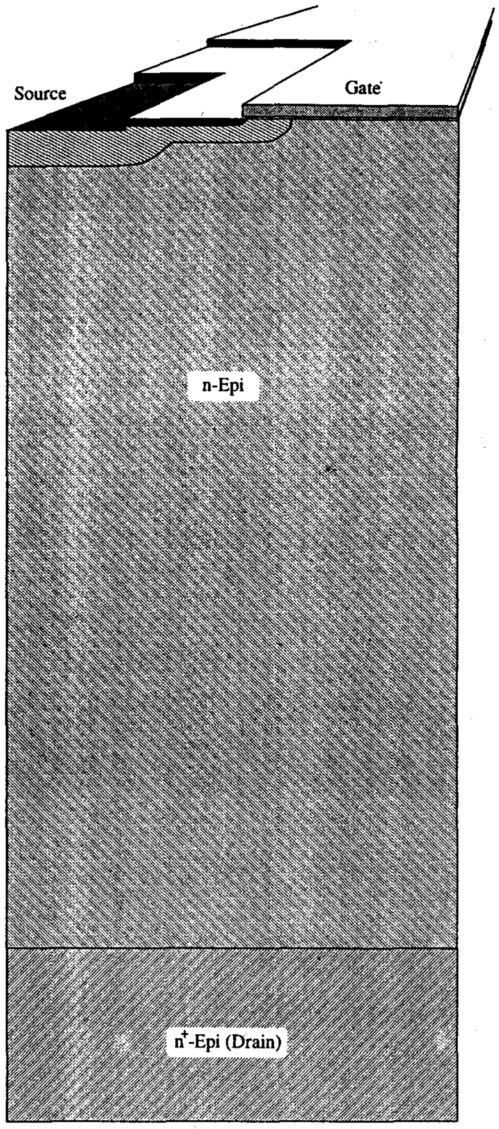


Figure 8: Structure of vertical DMOS transistor (source metalization and gate oxide are not shown)

For simulating the diffusion processes the commercial simulator TSUPREM4 [8], for the simulation of breakdown voltage and on-resistance MINIMOS-NT [9] were used.

After the generation of a Central Composite Inscribed design and calculation of 9 runs (Figure 3), a response surface model is generated and the optimization is started. The optimizer solves the constrained problems by querying the RSM-tool for results of the fitted surface. The surface is defined

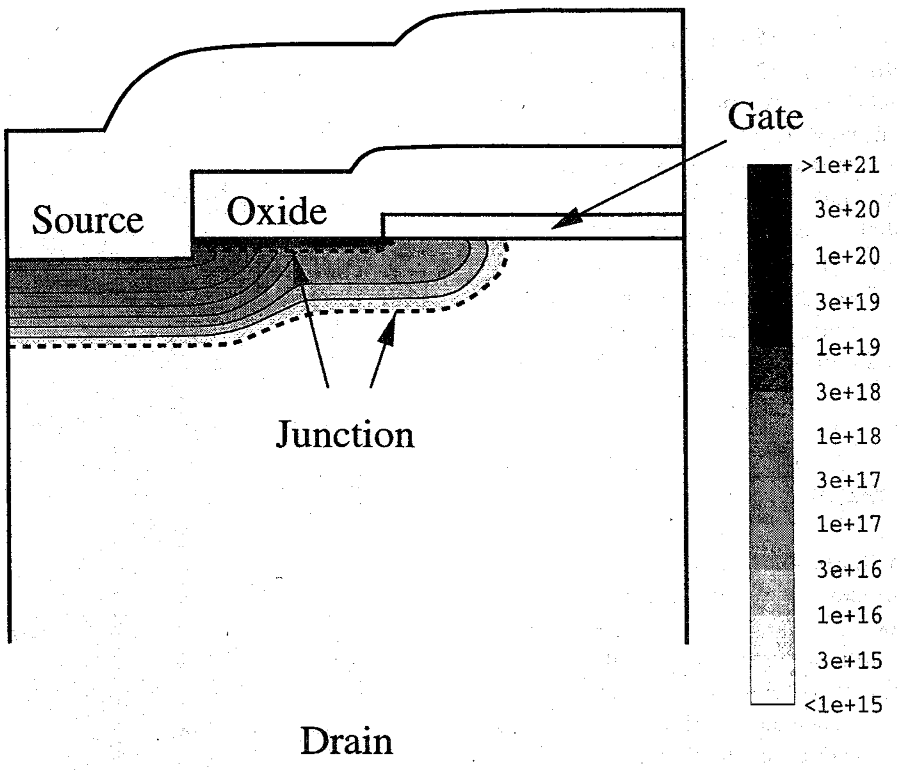


Figure 6: The final VDMOS transistor. The top of the structure is shown along with the netdoping concentration (cm^{-3}).

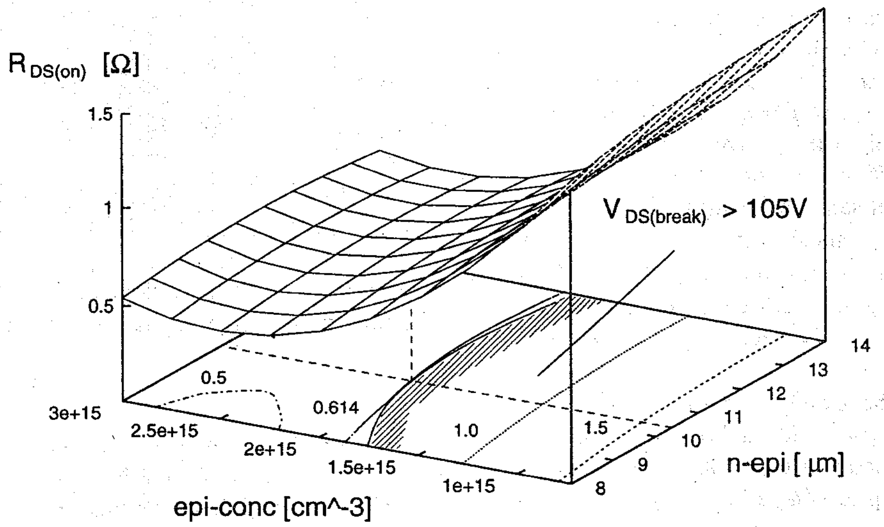


Figure 7: Response Surface Methodology plot of the target function (on-resistance) versus the epi-doping concentration (`epi-conc`) and the epi-layer thickness (`n-epi`). The optimal value is found on the breakdown voltage iso-line for the lower limit of 105 V.

as the on-resistance depending on the two control variables epi doping and epi thickness spanning the plane for the on-resistance surface (see Figure 7). The specified limits for the two control variables and the breakdown voltage constraint define the range of allowed values for the control variables.

The minimum value of the on-resistance within this range of valid parameter values is found on the boundary where the breakdown voltage exhibits the allowed minimum of 105 V (Figure 7). Thus, the optimal process parameters are found with `n-epi` $= 10.37\mu\mathrm{m}$, `epi-conc` $= 1.9e15 cm^{-3}$, and $R_{DS(on)} = 0.614\Omega$. Table 4 comprehends the optimal process parameters and on-resistance.

Table 4: Optimal process parameters and corresponding on-resistance.

Epi doping	1.9e15 cm^{-3}
Epi thickness	10.37 μm
On-resistance	0.614 Ω

Thus, the simulation gives an on-resistance which is reduced by 20% in respect to the nominal value. However, for this optimum value the breakdown voltage exhibits its lower limit of 105 V. Hence, the latitude for statistical process variations and other deviations from the nominal specifications is very small. Regarding statistical variations and three-dimensional effects (as only two-dimensional simulations were performed) the lower limit for the breakdown voltage should be slightly higher which then will end up with an on-resistance corresponding to the nominal value of 0.9Ω.

5 Conclusion

It has be shown that TCAD is suitable for simulating and optimizing processes for smart power technology. As an example a vertical DMOS transistor has been analyzed and the obtained results compare very well to real device specifications. The capabilities of split-lot experiments, response-surface-modeling, design-of-experiment, and optimization make VISTA a powerful instrument not only for VLSI technology but also for smart power technology. To alleviate the problems of disk space and computation time requirements new concepts are under development which include a more efficient database management, powerful gridding tools, and faster solver algorithms. Thus, the next generation of Technology CAD capable to deal with three-dimensional and complex structures is to appear in the near future.

References

[1] S. Halama, F. Fasching, C. Fischer, H. Kosina, E. Leitner, C. Pichler, H. Pimingstorfer, H. Puchner, G. Rieger, G. Schrom, T. Simlinger, M. Stiftinger, H. Stippel, E. Strasser, W. Tuppa, K. Wimmer, and S. Selberherr, "The Viennese Integrated System for Technology CAD Applications," in *Technology CAD Systems* (F. Fasching, S. Halama, and S. Selberherr, eds.), (Wien), pp. 197–236, Springer, 1993.

[2] Institute for Microelectronics, Technical University Vienna, *VISTA Documentation 1.3-1, VLISP Manual*, Jan. 1996.

[3] F. Fasching, W. Tuppa, and S. Selberherr, "VISTA-The Data Level," *IEEE Trans.Computer-Aided Design*, vol. 13, no. 1, pp. 72–81, 1994.

[4] S. Duvall, "An Interchange Format for Process and Device Simulation," *IEEE Trans.Computer-Aided Design*, vol. 7, no. 7, pp. 741–754, 1988.

[5] T. Lorenzen and V. Anderson, *Design of Experiments*. Marcel Dekker, 1991.

[6] G. Box and N. Draper, *Empirical Model-Building and Response Surfaces*. Wiley, 1987.

[7] N. Khalil, *ULSI Characterization with Technology Computer-Aided Design*. Dissertation, Technische Universität Wien, 1995.

[8] Technology Modeling Associates, Inc., Palo Alto, CA, *TMA TSUPREM-4, Two-Dimensional Process Simulation Program, Version 6.2*, June 1995.

[9] T. Simlinger, H. Kosina, M. Rottinger, and S. Selberherr, "MINIMOS-NT: A Generic Simulator for Complex Semiconductor Devices," in *ESSDERC'95 - 25th European Solid State Device Research Conference* (H. de Graaff and H. van Kranenburg, eds.), (Gif-sur-Yvette Cedex, France), pp. 83–86, Editions Frontieres, 1995.

Acknowledgment

This work is supported by Austria Mikro Systeme AG, Unterpremstätten, Austria; Siemens Corporation at Munich, Germany; and Christian Doppler Gesellschaft, Vienna, Austria.