PAPER *Special Issue on SISPAD'99*

Practical Inverse Modeling with SIESTA*

Rudolf STRASSER[†a)] ***and*** **Siegfried SELBERHERR**[††], ***Nonmembers***

SUMMARY We present a simulation system which meets the requirements for practical application of inverse modeling in a professional environment. A tool interface for the integration of arbitrary simulation tools at the user level is introduced and a methodology for the formation of simulation networks is described. A Levenberg-Marquardt optimizer automates the inverse modeling procedure. Strategies for the efficient execution of simulation tools are discussed. An example demonstrates the extraction of doping profile information on the basis of electrical measurements.

key words: *calibration, inverse modeling, parameter identification, tool integration, parallel and distributed computation*

1. Introduction

TCAD simulation tools are widely used throughout the semiconductor industry. However, their efficient application requires a sound calibration of the involved models. Their parameters need to be tuned in order to achieve an acceptable accuracy of the simulation results. Moreover, if used for inverse modeling purposes, TCAD models offer an attractive alternative to measurements of doping profiles. Recently inverse modeling is emerging as a standard methodology. In [2] as well as in [4] the authors describe a calibration methodology based on inverse modeling. The extraction of various structural data (channel and source/drain profiles) of MOSFET devices based on electrical measurements is reported in [3]. However, the complexity of the involved models inhibits a manual procedure and, therefore, an optimizer must be employed to automate the procedure. The simulation environment SIESTA [6] can be utilized to solve generic inverse modeling problems.

Figure 1 illustrates how SIESTA's optimizer is searching for sets of model parameters which deliver an optimal fit between simulation results and measurements: A simulation model is evaluated with a given set of parameters and its results are compared to measurements; the optimizer receives the difference between simulation and measurement, and uses it to compute improved sets of parameters.

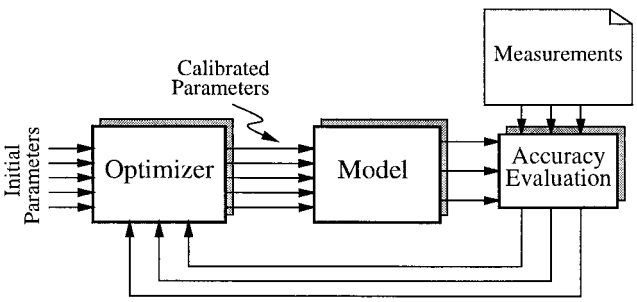

Fig. 1 An automated inverse modeling system. The simulation model can represent process, or device simulations, or their combinations. The optimizer receives the differences between simulated and measured quantities as feedback to its attempts.

2. A General Purpose Inverse Modeling Setup

As the rigorous calibration of simulation models can be a cumbersome procedure, models are often poorly calibrated based on a very limited number of measurements. Thus, the models deliver satisfying results only as long as they are used in the vicinity of the conditions of the calibration. Due to such a restricted calibration within small subspaces of the model's parameter ranges, it can occur that models are *miscalibrated* which means that differences between measurements and simulation are fitted by changing a parameter which had better been kept untouched. Although the model accuracy can be improved slightly for the measurements taken into account for that kind of calibration, the accuracy might even have worsened for operating regions which remained unconsidered by the calibration procedure.

2.1 Measurement Data Integration

To minimize the risk of miscalibration, it is of utmost importance to include as many measurements in the calibration procedure as available such as depicted in Fig. 2. This increases the probability that the overall accuracy really improves due to calibration and a physically sound set of parameters can be found. However, time restrictions of TCAD engineers will inhibit rigorous calibrations unless the calibration procedure is automated as far as possible, and a methodology

Manuscript received November 24, 1999.
Manuscript revised February 25, 2000.

[†]The author is with Infineon Technologies, SIM, Otto-Hahn-Ring 6, D-81739 Munich, Germany.
[††]The author is with the Institute for Microelectronics, TU Vienna, Gusshausstr, 27-29, Vienna, Austria.
a) E-mail: rudolf.strasser@infineon.com

*This paper was presented at SISPAD 1999, Kyoto, Japan.

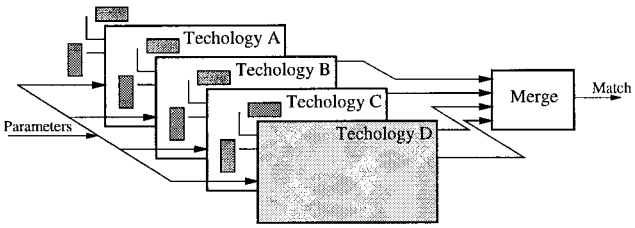

Fig. 2 Benchmarking of models based on multiple technologies or devices.

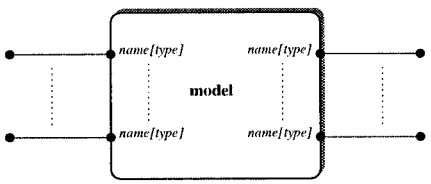

Fig. 3 A generic model is represented by a black box with well defined input and output ports.

enables the efficient description of modeling problems which includes a variety of measurements into the procedure. Moreover, these calibrations can require significant amounts of simulation time which raises demand for efficient processing of the evaluations of the simulation model.

3. Simulation Tool Integration

An open simulation tool interface enables the cooperation between SIESTA and arbitrary simulation tools. This interface imposes no restrictions on simulation tools and, in particular, it does not require their modification. The simulation tool interface creates an abstraction of a simulation tool by encapsulating it. Thus, from the optimizers point of view it is a black box with several well defined input parameters (ranges, default values) and output data such as depicted in Fig. 3.

3.1 Simulator Control and Result Management

Figure 4 illustrates how SIESTA controls simulators within these black boxes. Dedicated parts of the simulators input deck (e.g. *vt-dose, vt-energy*) or parts of its command line are marked by pairs of "< (" and ") >" which are controllable by the simulation environment. Arbitrary results of simulation tools are registered with SIESTA and are accessible for subsequent simulation tools. Users are able to form sequences of simulation tools and link these tools such as depicted in Fig. 5 in order to create so called simulation-flow-models which serve as an encapsulation of that sequence. Such a sequence could for example be a mask generation tool followed by a process simulator and a device simulator. Each of these tools has access to the output of one of its predecessors, and to the inputs ports of the simulation-flow-model.

```
defop VTIMPLANT() {
  comment(text : "VT Implant")
  implant(species : bf2,
          dose : <(vt-dose)> /cm2,
          energy : <(vt-energy)> keV,
          side : front,
          type : &Th_Adjust_Impl)
}
```

Fig. 4 SIESTA replaces symbols of input deck templates, marked by "< (" and ") >", by their actual values. A simulation tool can use an unlimited number of such input deck templates.

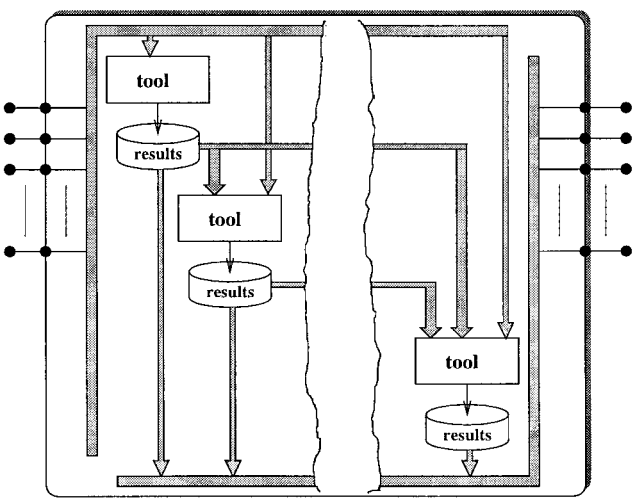

Fig. 5 A simulation-flow-model encapsulates a sequence of simulation tools. It manages their input files and their execution as well as their output files.

3.2 Evaluation Networks

Additionally, SIESTA offers the capability to create simulation networks of individual models (which themselves encapsulate simulation tools). Figure 6 illustrates how this feature is utilized in order to include several measurements into the inverse modeling procedure. Each part of this network evaluates the accuracy of the simulation model with respect to specific measurements, and returns a vector of float values quantifying the accuracy. Finally, these vectors are concatenated into a vector which represents an overall measure of accuracy for all measurements under consideration. This vector will be fed back to a Levenberg-Marquardt optimizer. Thus, we are able to enhance the confidence of the inverse modeling procedure by including as many measurements as possible. Additionally, one could expect that the number of local minima is kept as low as possible by including any available knowledge into the optimization procedure.

As far as parts of an evaluation network do not depend on each other they are evaluated concurrently. In other words, if a simulation networks contains for example a process simulation followed by several device characterizations, SIESTA evaluates the devices simu-

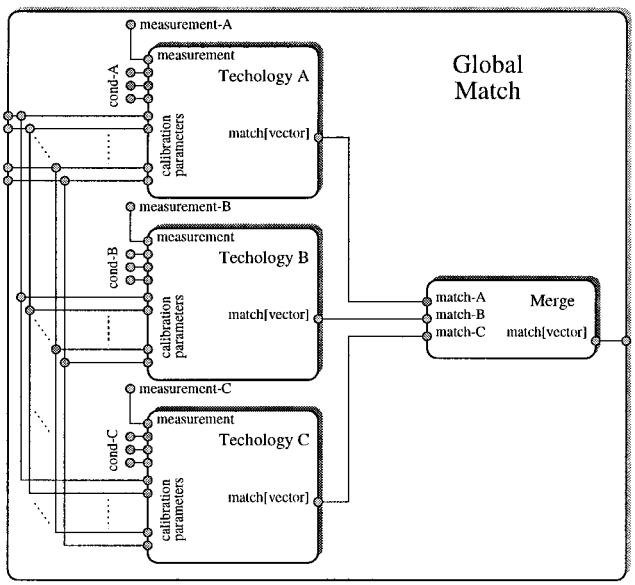

Fig. 6 Evaluation networks enable the integration of several simulation models.

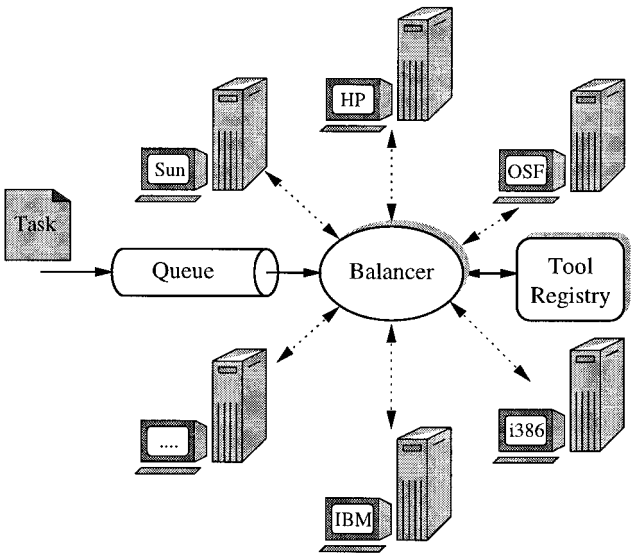

Fig. 7 The job farming infrastructure. A load balancing mechanism distributes workload to a heterogenous cluster of workstations.

lations concurrently which means that simulators are executed in parallel. Consequently, this leads to a significant reduction of simulation time, given that the simulation workload can be distributed to a cluster of workstations.

4. Parallel Computation

Inverse modeling requires a considerable computational effort. Optimizers typically evaluate the Jacobian of the simulation model with respect to the parameters to be optimized. This means that the modeling network described above has to be evaluated at least once for each parameter per gradient, and thus numerous evaluations are necessary. Individual evaluations during gradient computation are independent from each other and can therefore be carried out concurrently. This means that in combination with SIESTA's job farming capabilities (Fig. 7) which can handle parallel and distributed executions of simulation tools on a heterogeneous cluster of workstations [7], we are able to reduce simulation time considerably. A dynamic load balancing mechanism optimizes the utilization of the available computer hardware and a tool management mechanism handles simulation tools and their licenses.

4.1 Load Balancing

Whenever parallel computation is employed to reduce the real time for a given computation, the completion of the whole computation depends on the very last part to finish. This means that load balancing is necessary in order to optimize computation efficiency. Load balancing means that faster machines should get a bigger share of the whole workload and, on the other hand, slower machines should obtain less of it. Thus, since the faster machines do more work than the slower ones do, the former will have to wait less until the latter finish. The ultimate case occurs when workload is optimally balanced and all parts finish simultaneously.

The operating system loads (e.g. measure deliverd by the `uptime` command) of each computation hosts are continuously monitored. Based on actual load values simulation jobs are submitted to hosts according to the following criteria:

- The system command to be invoked must be available on a host.
- A host has to be reachable and its load must not exceed a certain limit *after* the job has been submitted.
- Finally the estimated performance of the selected host has to be superior compared to the remaining hosts.

4.2 License Management

Each tool's registration introduces a list of computation hosts where it is potentially available. Simulators might be restricted to specific hosts due to the existence of so called node-locked licenses which require that it is executed on a certain machine. Another reason for the restriction to a subset of the available hosts can be different computer architectures or operating systems. It is easily possible that a simulation tool is only available for some operating system. Furthermore, memory considerations might force a user to launch a simulation tool only on hosts where sufficient memory for a proper operation of the tool is available. It should be stressed that a registration is only necessary for tools

which either are not available on every host, or for tools which have a limited number of available licenses.

Additionally to the host at which a tool is available, the number of licenses available for that tool can be of importance. These licenses are usually a resource which is shared among concurrent users and, therefore, need to be managed carefully. Otherwise, situations occur where the automated occupation of tool licenses as it happens in SIESTA always grabs unused licenses before interactive users of simulation tools are able to do it. Therefore, the SIESTA tool registry offers a way to define how many licenses of a tool can be occupied by SIESTA.

4.2.1 Host Validation and Ranking

Each host is registered with SIESTA by defining a performance metric w_i of its CPUs, the number of CPUs n_i^{cpu}, and the desired maximum load l_i^{lim}. A host is considered to be available if its current load does not exceed

$$l_i^{max} = \left(l_i^{lim} + l_{base}\right) + 1,$$

where l^{base} denotes an amount of workload by which the limit of each individual host is increased. For each available host a ranking

$$p_i = \frac{\max\left(1\ ,\ \frac{l_i^{eff}+1}{n_i^{cpu}}\right)}{w_i} \tag{1}$$

is computed, which is an estimate for the performance that could be obtained if a job were executed under the hosts current operating conditions. Out of all hosts that have been identified to be suitable for a simulation tool, the one with the smallest value of p_i is selected for computation. The setting of l_{base} can be utilized to increase the load limits of all hosts simultaneously in situations where none of the hosts is below its load limit which might be caused by jobs of other users.

Since the memory which is required for a specific simulation job is not known a priori, and it cannot even be estimated somehow, (depends very much on in input deck settings etc), the machines memory equipment cannot be taken into account. However, users are able to lock specific tools to a subset of the available machines, and can therefore circumvent problems arising from excess memory consumption.

5. Performance Estimation

To illustrate the benefits arising from job-farming let us consider a rigorous calibration of a device simulator. For the estimation of the required simulation time let us assume the following: Transfer curves (I_D/V_G) with the overall number of N operating points are available, M parameters have to be calibrated, I optimizer iterations are necessary, W workstations are available for computation, and the typical computation time required per operating point is T.

Given that each optimization iteration consists of gradient computation and evaluation, the overall computation time is roughly $(M+1) \times I \times T \times N$. Parallel evaluation of transfer curves reduces this time to $(M+1) \times I \times T \times \frac{N}{W}$. For $N = 30$, $M = 4$, $W = 15$, $I = 100$, and $T = 1\,\text{min}$, this means that job farming is able to reduce the time compared to operation on a single workstation from approximately 10 days to 16 hours.

6. Inverse Modeling of Doping Profiles

During technology development device simulations are sometimes required at stages where neither SIMS data of doping profiles, nor calibrated device simulations are available. As long as electrical measurements are the only input to TCAD simulations, these measurements can be used to search for doping profiles which are related to I/V-curves. This extraction is non trivial since usually a doping profile has to be found which delivers a satisfactory fit for a couple of geometric variants. Hence, simulation offers a comparably cheap and fast alternative to measurements in this situation. Several attempts in this direction can be found in the literature [2]–[4]. The following example demonstrates how the extraction of the channel profile of MOSFET devices can be performed with SIESTA. Electrical measurements (transfer and output characteristics) of MOS devices with gate lengths of 0.18 μm, 0.25 μm, and 0.5 μm are used to extract their doping profiles.

6.1 Modeling the NMOS Structure

Although we could produce the electrical device by means of a full process simulation, we are for simplicity using MAKEDEVICE [1] to create a synthetic NMOS device. Figure 8 shows a schematic view of a semiconductor device produced. It contains several elements which build up the doping profile. They are either profiles of Gaussian, Pearson, or constant shape. Each Pearson shaped profiles contribution uses five parameters for dose, projected range, standard deviation, skewness, and kurtosis. There are doping elements for the source and drain regions, for the lightly doped drain, for the threshold adjust implant, and for the punch through implant.

MAKEDEVICE allows control of these doping elements by means of parameters of its input deck. Figure 9 shows a simulation-flow-model named *gendevice* which manages the invocation of the MAKEDEVICE program. Figure 10 lists the input deck which is used to describe the doping profiles of the NMOS device. This input deck contains template symbols which refer to input ports of the simulation-flow-model. Thus, one

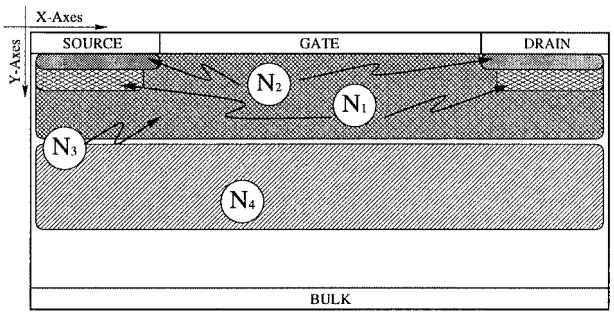

Fig. 8 A schematic view of the NMOS device which is modeled by means of analytical doping profiles. N_1 and N_2 model the source/drain wells and the lightly doped source/drain extensions, respectively. N_3 and N_4 represent the threshold-adjust implant, and the anti-punch-through implant, respectively.

```
(simulation-flow-model gendevice
  (inputs
      (gatelength      float)
      (deltaLg         bound-float   0.      -0.005   0.005)
      (tox             bound-float    3.4e-3  0.0028  0.0040)
      (nSubDonor       bound-float   14.65   14.      15.)
      (nSubAcceptor    bound-float    5.      0.      15.)
      ;; source/drain peak
      (n1              bound-float   20.42   20.3     20.8)
      (xpos1           bound-float   0.00    -0.02    0.02)
      (xsigma1         bound-float   0.0053  0.004    0.05)
      (ypos1           bound-float   0.00    -0.005   0.02)
      (ysigma1         bound-float   0.01    0.01     0.1)
      (gamma1          bound-float   0.01    -1.0     1.0)
      (betas1          bound-float   1.0     0.1      100.)
      ;; ldd peak
      (n2              bound-float   19.51   19.4     19.18)
      (xpos2           bound-float   0.001   -0.02    0.01)
          .
          .
          .
      (ypos4           bound-float    0.72   0.5      0.8)
      (ysigma4         bound-float    0.8    0.5      1.0))
  (outputs             (device filename))
  (output-mapping      (device synthesize.device))
  (tool-flow
    (synthesize   (tool
                    (controls
                      (command   "mkdev")
                      (cmdline   "ipd")
                      (ipd       (file "mkdev.ipd")))
                    (results
                      (device  "makedevice.pbf"  filename))))
))
```

Fig. 9 The model description needed for interaction of SIESTA and the device generator MAKEDEVICE.

is able to specify the parameters of the doping profile at the model's input ports. A sample device produced by this model is depicted in Fig. 11.

Figure 12 depicts the simulation network which evaluates a device with a given gate length and an analytical doping profile defined by several parameters. The model named *gendevice* generates an NMOS device according to the input settings. In the following two models named *mmnt-idvg* and *mmnt-idvd* evaluate the

```
.....
<Geometry>
{
  gateLength = <(gatelength)> + (<(deltaLg)>);
  oxideThickness = <(tox)>;
  .....
}
<Doping>
{
  ........
  // substrate concentrations
  NsubAcceptor = <(nSubAcceptor)>;
  NsubDonor    = <(nSubDonor)>;    // substrate donor doping
  .......
  <Peak>
  {
     <Peak1>  // Source
     {
         mode = 3;                      // Pearson mode
         switchAccDon = 1;              // Donor Type
         N = <(n1)>;
         x = 0.;
         xLength = SW+SG+(<(xpos1)>);
         y = <(ypos1)>;
         yLength = 0;
         xSigma= <(xsigma1)>; // lateral
         ySigma= <(ysigma1)>; // vertical
         gamma = <(gamma1)>;
         beta  = <(betas1)>
     }
     <Peak2>  // Drain
     {
      .....
```

Fig. 10 Elements of the synthetic doping profile are customized in the input deck of MAKEDEVICE.

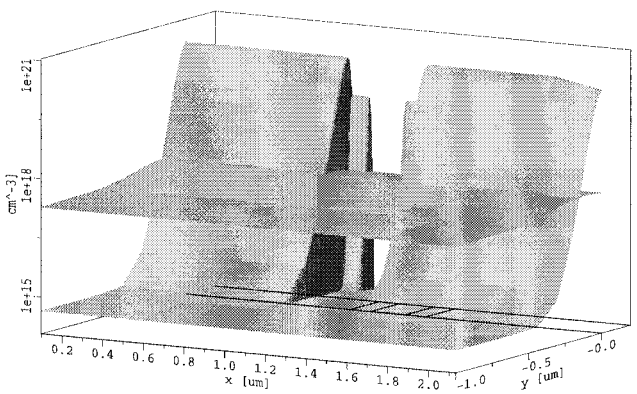

Fig. 11 A sample device with the initial profile which was synthesized by analytical doping profiles using MAKEDEVICE.

transfer and the output characteristics, respectively, of that device using the device simulator MINIMOS-NT [1]. Device simulation is based on a calibrated drift diffusion model and its parameters are not modified.The models named *compare-idvg* and *compare-idvd* compare the results of device simulation to measurements, and deliver a vector of floating point numbers which give a measure of *match* for each operating point under consideration according to

$$m = \begin{cases} \frac{I_{meas}-I_{sim}}{I_{meas}} \cdot 100\% & I_{meas} > I_{sim} \\ \frac{I_{sim}-I_{meas}}{I_{sim}} \cdot 100\% & I_{sim} > I_{meas} \end{cases}$$

Finally, these match vectors for transfer and output curves are concatenated and delivered as the result of the whole model. An evaluation network as depicted

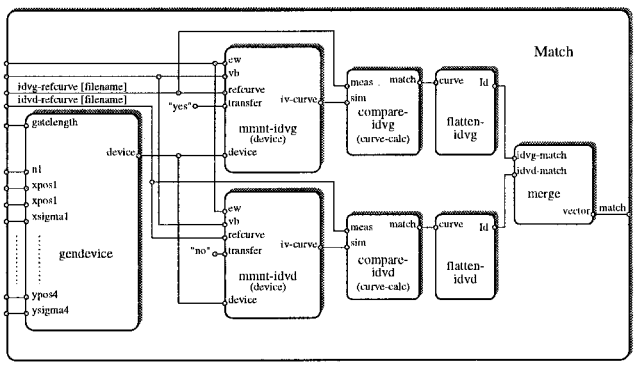

Fig. 12 A simulation network for the evaluation of a given device structure defined by the input settings. Beginning from left the device structure is generated at first followed by two device simulations evaluating output- and transfer characteristics. Finally a match metric is formed based on the results of the device simulations and the measurements which are given at the model's input ports.

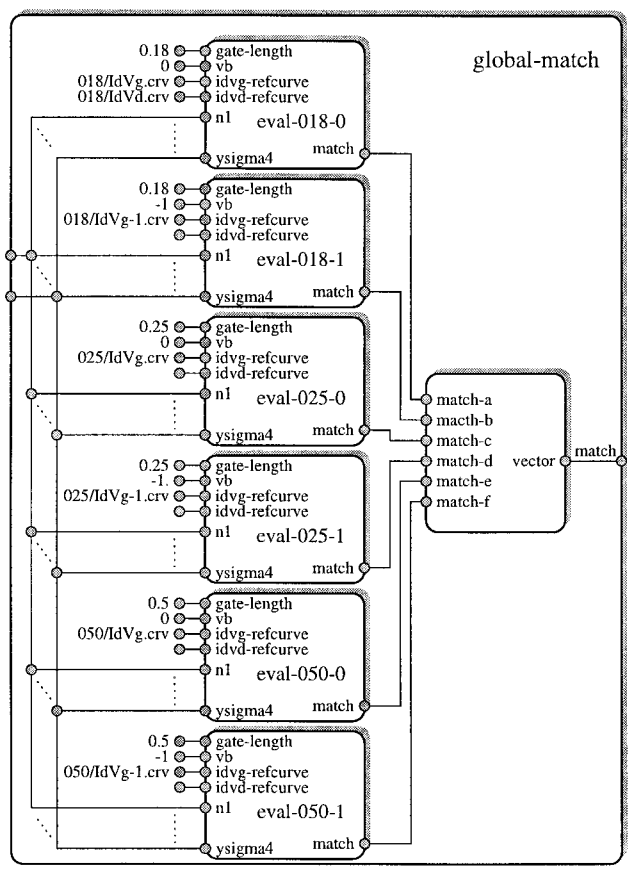

Fig. 13 Each part of this model represents a comparison of a device model's (with a specific gate length which is operating at a given bulk bias) I/V characteristics against corresponding measurements. The difference for each operating point is a scalar component in the vector of the model's output.

in Fig. 13 is designed on the basis of the model depicted in Fig. 12 in order to evaluate devices with different gate dimensions simultaneously.

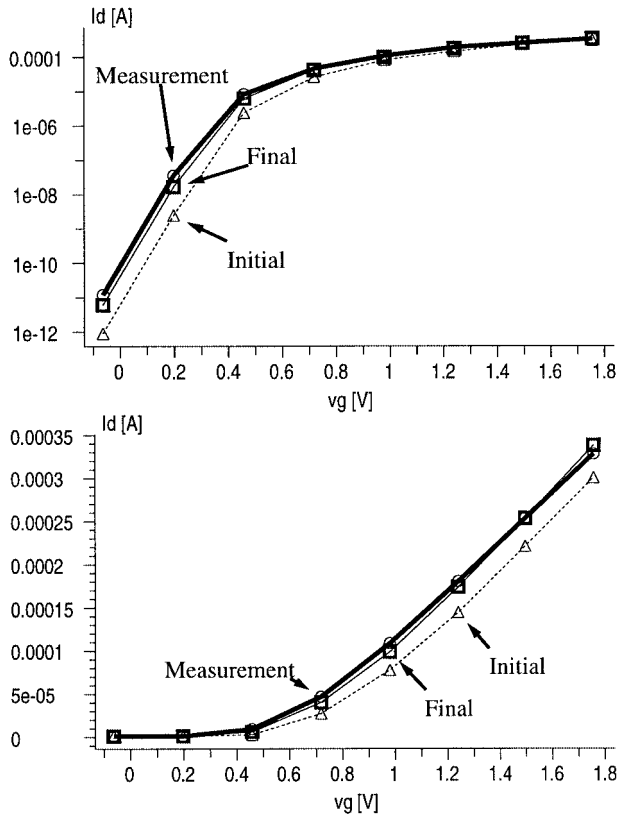

Fig. 14 The drain current for $V_d = 1.5\,\mathrm{V}$ of the $0.5\,\mu\mathrm{m}$ device as measured, for the initial doping profile, and for the final doping profile.

Figure 14 shows the measured transfer curves, the simulated results corresponding to the initial channel doping, and the transfer curves obtained from the channel profile which resulted from inverse modeling. As can be seen from Fig. 14 the system is able to identify a doping profile which delivers an excellent fit. Figure 15 depicts the dose of the two vertical profiles as well as their standard deviation. It also shows the evolution (individual settings for each iteration cycle) during the optimization procedure. The example can easily be extended in order to include additional measurements (e.g. different potentials at the bulk contact etc.). Despite the enormous effort which is related to these device simulations during the optimization procedure, the experiment takes no more than a couple of hours on a cluster of twenty workstations. A graphical user interface assists users during the inverse modeling experiment (Fig. 16). The history of the model's parameters and its accuracy can be browsed graphically.

7. Conclusion

The presented system has successfully been applied to calibration problems, parameter identification/extraction, and to the extraction of doping profiles on the basis of electrical measurements. In practice the

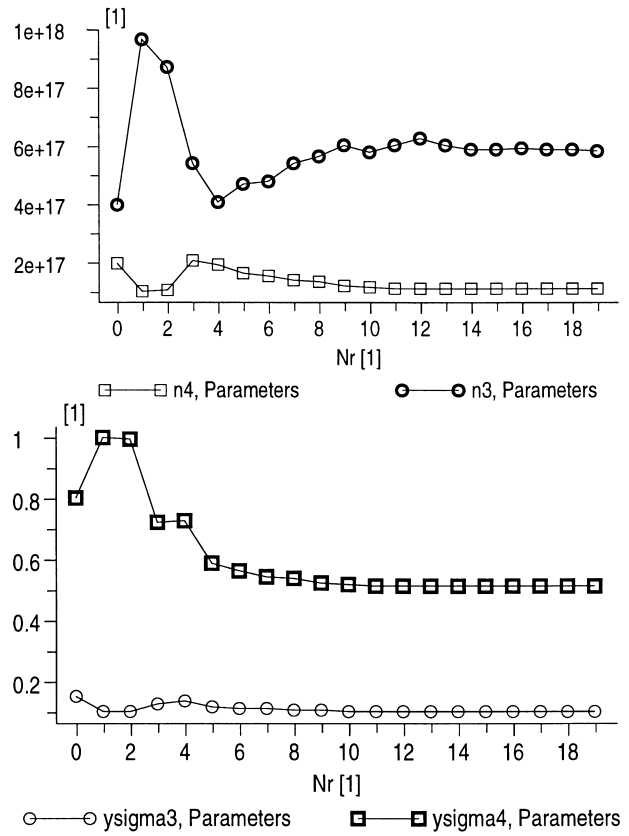

Fig. 15 The parameters of the vertical Pearson profiles during optimization.

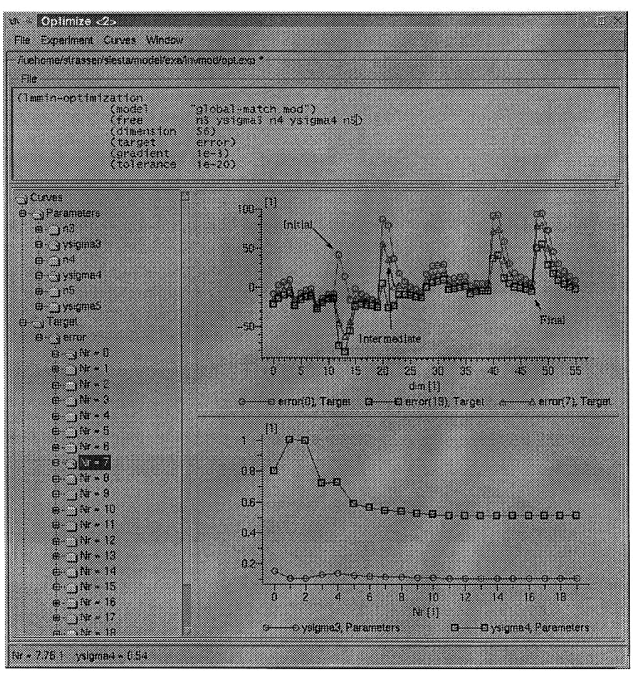

Fig. 16 A graphical user interface enables SIESTA's users to track the optimization progress.

open tool interface has proven to be extremely valuable since arbitrary simulation tools can either be calibrated or they can be utilized to extract doping profiles. State of the art TCAD tools (PROMIS, DIOS, TSUPREM, MINIMOS, DESSIS, and MEDICI) have been used in conjunction with SIESTA. The performance as well as the robustness of the system encourage SIESTA's usage for routine tasks in an industrial environment. Due to parallel computation the overall computation time is no longer a severe constraint for TCAD experiments of this kind.

Acknowledgment

Major parts of this work were carried out with the support of the "Christian Doppler Forschungsgesellschaft," Vienna, Austria, Austria Mikro Systeme, Unterpremstätten, Austria, and Sony Corporation, Atsugi, Japan.

References

[1] T. Binder, K. Dragosits, T. Grasser, R. Klima, M. Knaipp, H. Kosina, R. Mlekus, V. Palankovski, M. Rottinger, G. Schrom, S. Selberherr, and M. Stockinger, MINIMOS-NT Users's Guide, Institut für Mikrolektronik, TU Wien, 1.0 ed., 1998.

[2] A. Das, D. Newmark, I. Clejan, M. Foisy, M. Sharma, S. Venkatesan, S. Veeraraghavan, V. Misra, B. Gadepally, and L. Parrillo, "An advanced MOSFET design approach and a calibration methodology using inverse modeling that accurately predicts device characteristics," Int. Electron Devices Meeting, pp.687–690, 1997.

[3] N. Khalil, "ULSI characterization with technology computer-aided design," Dissertation, Technische Universität Wien, 1995.
`http://www.iue.tuwien.ac.at/diss/khalil/diss/diss.html`

[4] Z.K. Lee, M.B. McIlrath, and D.A. Antoniadi, "Inverse modeling of MOSFETs using I-V characteristics in the subthreshold region," Int. Electron Devices Meeting, pp.683–686, 1997.

[5] M. Rottinger, N. Seifert, and S. Selberherr, "Analysis of AVC measurements," eds. A. Touboul, Y. Danto, J.-P. Klein, and H. Grünbacher, 28th European Solid-State Device Research Conference, pp.344–347, Bordeaux, France, Editions Frontieres, 1998.

[6] R. Strasser, "Rigorous TCAD investigations on semiconductor fabrication technology," Dissertation, Technische Universität Wien, 1999.
`http://www.iue.tuwien.ac.at/diss/rudolf_strasser/diss/diss.html`

[7] R. Strasser and S. Selberherr, "Parallel and distributed TCAD simulations using dynamic load balancing," in Simulation of Semiconductor Processes and Devices, eds. K. De Meyer and S. Biesemans, pp.89–92, Springer, Wien, New York, 1998.

Rudolf Strasser was born in Ried im Innkreis, Austria, in 1970. He received the 'Diplomingenieur' degree in electrical engineering and the Ph.D. degree from the 'Technische Universität Wien' in 1995 and 1999, respectively. From spring 1992 to autumn 1993 he held a research position at the campus-based Engineering Center of Digital Equipment Corporation, Vienna, Austria. He joined the 'Institut für Mikroelektronik' in April 1995. In summer 1996 Dr. Strasser was with the Advanced Products Research and Development Laboratory at Motorola, Austin, USA. Since June 1999 he is with Infineon Technologies, Munich, Germany. His scientific interests include semiconductor technology, grid generation, and software engineering.

Siegfried Selberherr was born in Klosterneuburg, Austria, in 1955. He received the degree of 'Diplomingenieur' in electrical engineering and the doctoral degree in technical sciences from the 'Technische Universität Wien' in 1978 and 1981, respectively. Dr. Selberherr has been holding the 'venia docendi' on 'Computer-Aided Design' since 1984. Since 1988 he has been the head of the 'Institut für Mikroelektronik' and since 1999 he is dean of the 'Fakultät für Elektrotechnik'. His current topics are modeling and simulation of problems for microelectronics engineering.