



## VISTA Status Report December 1999

T. Grasser, A. Hössinger, R. Kosik, R. Mlekus, W. Pyka,  
M. Stockinger, S. Selberherr



**Institute for Microelectronics  
Technical University Vienna  
Gusshausstrasse 27-29  
A-1040 Vienna, Austria**



# Contents

<b>1</b>	<b>Mixed-Mode Device Simulation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Circuit Simulation . . . . .	2
1.3	Thermal Simulation . . . . .	2
1.4	Device Simulation . . . . .	3
1.5	Mixed Mode Simulation . . . . .	3
1.6	Convergence . . . . .	4
1.7	Examples . . . . .	5
1.7.1	Five-Stage CMOS Ring Oscillator . . . . .	5
1.7.2	Five-Stage CML Ring Oscillator . . . . .	6
1.7.3	Electro-Thermal Analysis of a Complete OpAmp . . . . .	6
<b>2</b>	<b>Closed-Loop CMOS Gate Delay Time Optimization</b>	<b>16</b>
2.1	Introduction . . . . .	16
2.2	Optimization Procedure . . . . .	16
2.3	Doping Profile Optimization . . . . .	18
2.3.1	Two-Dimensional . . . . .	18
2.3.2	Implantation Models . . . . .	18
2.4	Discussion . . . . .	20
<b>3</b>	<b>Modeling of High-Pressure Chemical Vapour Deposition</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	CVD-Model . . . . .	23
3.3	Applications and Results . . . . .	25
<b>4</b>	<b>Parallelization of a Monte-Carlo Ion Implantation Simulator</b>	<b>27</b>
4.1	Introduction . . . . .	27
4.2	Parallelization Strategy . . . . .	27
4.3	Simulation Flow . . . . .	28
4.4	Results . . . . .	29

4.5	Conclusion . . . . .	30
4.6	Acknowledgment . . . . .	30
<b>5</b>	<b>PROMIS-NT</b>	<b>31</b>
5.1	Features of PROMIS-NT . . . . .	31
5.2	Supported Model Structures . . . . .	31
5.2.1	Volume Models . . . . .	31
5.2.2	Boundary Models . . . . .	32
5.2.3	Process Temperature Modeling . . . . .	33
5.3	Quantity Management and VISTA Integration . . . . .	34
5.3.1	Charge States . . . . .	34
5.4	The Structure of PROMIS-NT . . . . .	34
5.5	The PROMIS-NT Input Deck . . . . .	35
5.5.1	Logical Structure of the PROMIS-NT Input Deck . . . . .	35
5.5.2	PROMIS-NT Process Flow . . . . .	38
5.6	PROMIS-NT Application Examples . . . . .	41
5.6.1	Five Species Phosphorus Pair-Diffusion Model . . . . .	41
5.6.2	Performance Considerations . . . . .	44
<b>6</b>	<b>Comparison of Finite Element and Finite Box Discretization</b>	<b>49</b>
6.1	Introduction . . . . .	49
6.2	Discretization using AMIGOS . . . . .	49
6.3	Numerical Experiments . . . . .	50
6.4	Impact on Meshing Strategies . . . . .	50
6.5	Conclusion . . . . .	50
6.6	Acknowledgments . . . . .	51

# 1 Mixed-Mode Device Simulation

## 1.1 Introduction

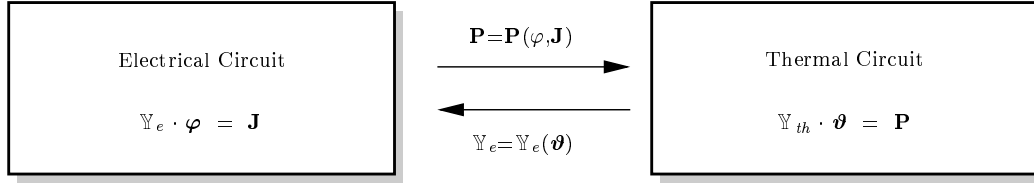
Over the last decades numerous powerful circuit simulation programs have been developed. Amongst those are general purpose programs which have been designed to cope with all different kinds of circuits and special purpose programs which provide highly optimized algorithms for, e.g., filter design. General purpose programs can be divided into two categories. Programs belonging to the first category offer a modeling language which can be used to define fairly arbitrary dependences between the circuit elements. The most prominent member of this category is **ASTAP** [?] which was developed by IBM in the 1970s. To provide the user with a maximum of flexibility, **ASTAP** generates **FORTRAN** source files which need to be compiled before execution. The other category consists of programs which only allow for a predefined set of circuit elements and dependences. Although the flexibility is strongly diminished, this approach allows for a much faster execution and a compact, highly optimized simulator kernel. The most prominent member of this category is **Spice** which was developed at the University of Berkeley [?].

Circuit simulation programs have in common that the electrical behavior of the devices is modeled by means of a compact model, that is an analytical expression describing the device behavior. Once a suitable compact model is found, it can be evaluated in a very efficient way. However, this task is far from being trivial and many complicated models have been developed. Even if the behavior of the device under consideration can be mapped onto one of the existing compact models, the parameters of this compact model need to be extracted. For example, in the case of the **BSIM3v3** model [?] for short-channel MOS transistors more than 100 parameters are available for calibration purposes, the identification of which is obviously a cumbersome task. Similar arguments hold for other available MOS transistor models as the **EKV** model [?, ?] and the **Philips MM9** model [?]. If the device design is known and not modified, these parameters need to be extracted only once and can be used for circuit design provided the accuracy of the models is sufficient. When there is need to optimize a device using modified geometries and doping profiles the compact model parameters have to be extracted for each different layout as many of these parameters are mere fit parameters without any physical meaning.

The electrical behavior of the devices can either be measured or simulated. When performing a device optimization, fabricating and measuring each optimization step would be very expensive. Hence, device simulators became more and more popular, e.g., **DESSIS** [?], **GALENE** [?], **MEDICI** [?], **MINIMOS 6** [?], and **PISCES** [?]. These device simulators solve the transport equations for a device with given doping profiles and a given geometry. The transport equations form a highly nonlinear partial differential equation system which cannot be solved analytically. Numerical methods have to be used to calculate a solution by discretizing the equations on a suitable simulation grid. The data obtained from these simulations can be used to extract the parameters of the compact model.

Altogether, this subsequent use of different simulators and extraction tools is cumbersome and error-prone. To overcome these problems several solutions have been published where a device simulator was coupled to **Spice** [?, ?]. This is again problematic when considering the communication between two completely different simulators. On the other hand some solutions were presented where circuit simulation capabilities were added to a device simulator [?]. However, the restrictions imposed are so severe that circuits containing more than a few distributed devices cannot be properly dealt with.

The examples in this paper were simulated using **MINIMOS-NT**, a device simulator which has been equipped with full circuit simulation capabilities with the only limitation being the amount of available computer resources. **MINIMOS-NT** is a general purpose device simulator developed as the successor of **MINIMOS 6** [?].



**Figure 1:** Interaction of the coupled electrical and thermal circuits.

With mixed-mode capabilities at hand devices can be characterized by their performance in a circuit as a function of transport models, doping profiles, mobility models, etc. This is of fundamental importance when investigating the behavior of modern submicron devices and non-mainstream devices like Heterostructure-Bipolar-Transistors (HBTs) [?] or High-Electron-Mobility-Transistors (HEMTs) [?, ?] where compact models are not so far developed. Furthermore, when the devices are scaled down, non-local effects become more and more pronounced which can alter the device behavior significantly. This cannot be handled by scaling the parameters of compact models.

## 1.2 Circuit Simulation

Several different methods have been published for the description of the circuit equations. However, nearly all circuit simulators employ methods based either on the nodal approach (NA) [?, ?, ?] or the tableau approach [?]. Methods based on the NA enjoy large popularity due to its ease of use. However, the basic NA only allows for current-defined branches. Voltage-defined branches can be introduced without extending the formulation by the use of gyrators [?, ?]. To properly account for voltage-defined branches the modified nodal approach (MNA) has been proposed which allows for the introduction of arbitrary branch currents [?].

## 1.3 Thermal Simulation

The standard way of treating temperature effects in semiconductor devices and circuits is based on the assumption of a constant device temperature which can be obtained by *a priori* estimates on the dissipated power or by measurements. However, in general this *a priori* assumed dissipated power is not in accordance with the resulting dissipated power. Furthermore, devices may be thermally coupled resulting in completely different temperatures than would be expected from individual self-heating effects alone. This is of special importance as many circuit layouts rely on this effect, e.g., current mirrors and differential pairs [?]. Therefore, the temperature must not be considered a constant parameter, but must be introduced as an additional solution variable [?, ?, ?, ?].

Thermal coupling can be modeled by a thermal circuit [?, ?] (cf. Fig. ??). The topological equations describing a thermal circuit are similar in form to Kirchhoff's equations and the branch relations map to familiar electrical branch relations. The electrical compact models have been extended to provide the device temperature as an external node. For distributed devices MINIMOS-NT solves the lattice heat flow equation [?] to account for self-heating effects. This is of course far more accurate than assuming a spatially constant temperature in the device and estimating the dissipated power by Joule-heat terms alone as is done for the compact models. To provide a connection to an external thermal circuit arbitrary thermal contacts are defined.

## 1.4 Device Simulation

The vast majority of today's routinely performed device simulations are based on a numerical solution of the basic semiconductor equations which include drift-diffusion current relations [?, ?, ?, ?]. The efficiency of this numerical device model allows its extensive use in device optimization.

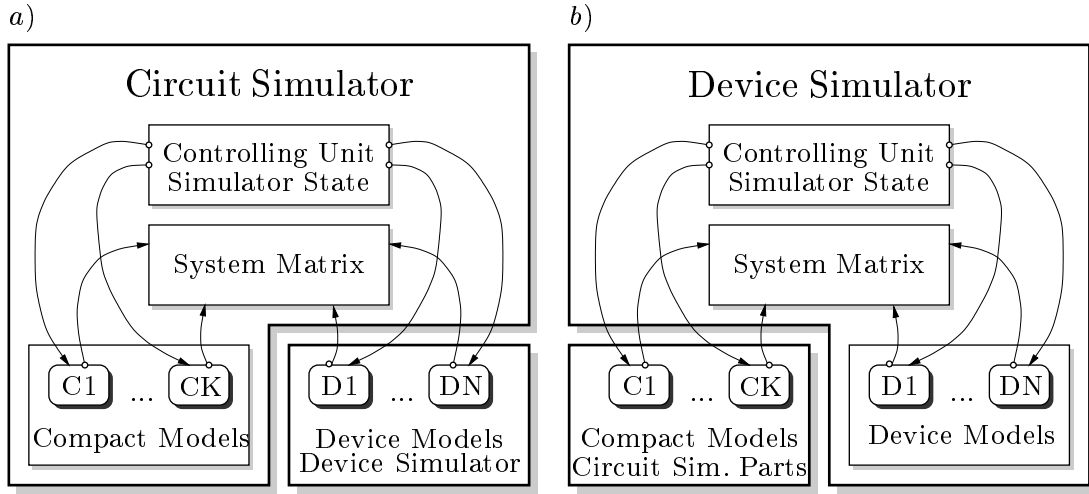
A device of a modern ULSI circuit is characterized by large electric fields in conjunction with steep gradients of the electric field and of the carrier concentrations. Under these conditions, the accuracy of the widely used drift-diffusion model becomes questionable. More sophisticated device models, such as the hydrodynamic transport model [?, ?, ?, ?, ?, ?, ?] overcome these limitations. However, the increased physical rigor of a model comes at the expense of increased computation times. This fact prevented wide spread application of the hydrodynamic model in the past, and probably in the near future. This is especially true for mixed-mode simulations which inherently suffer from large simulation times and poor convergence properties. Thus, the necessity of using the hydrodynamic model should be checked by comparison with drift-diffusion simulation results. However, for this comparison to deliver useful results, several prerequisites must be met, the most important of them being that both transport models must deliver similar results under homogeneous situations [?, ?].

## 1.5 Mixed Mode Simulation

Several works dealing with circuit simulation using distributed devices have been published so far [?, ?, ?, ?]. Most publications deal with the coupling of device simulators to *Spice*. This results in a two-level Newton algorithm since the device and circuit equations are handled subsequently. Each solution of the circuit equations gives a new operating point for the distributed devices. The device simulator is then invoked to calculate the resulting currents and the derivatives of these currents with respect to the contact voltages. In [?] a method was proposed which was termed full-Newton algorithm. However, this approach is very similar to the two-level method proposed in the same paper thus it will be termed "quasi" full-Newton. The difference to the two-level Newton lies in the fact, that the device simulator only performs the first step of the Newton iteration and returns the result to the circuit simulator. Both approaches are easy to implement as only marginal changes in both simulators are required. The circuit simulator acts as a server which controls the device simulator. At each Newton iteration of the circuit, an input deck for the device simulator has to be generated and the device simulator has to be called to calculate currents and conductances. The main advantage of this approach is that the device and circuit simulator are decoupled and special device simulators may be used for different problems.

The other approach is called full-Newton algorithm as it combines the device and circuit equations within one single equation system. This equation system is then solved applying Newton's algorithm. In contrast to the two-level Newton and the quasi full-Newton algorithm where the device and circuit unknowns are solved in a decoupled manner, here the complete set of unknowns is solved simultaneously. In MINIMOS-NT an approach similar to [?] is used. The capability to solve circuit equations was added to the simulator kernel. This allowed for assembling the circuit *and* the device equations into one system matrix which results in a real full-Newton method. There is no need to explicitly calculate the derivatives of the contact currents with respect to the contact voltages as the contact currents are solution variables which simply gives  $\pm 1$  as a derivative in the constitutive relations.

However, the benefits gained from using the numerous existing *Spice* compact models must not be neglected. As *Spice* has a well defined and documented interface, it is, in principle, straight-forward to implement a similar interface in the combined circuit-device simulator.



**Figure 2:** Comparison of the two different strategies: a) Device simulator as client.  
 b) Device simulator as server

A comparison of these different architectures is shown in Fig. ?? In Fig. ??a the device simulator acts as a client to the circuit simulator whereas in Fig. ??b the device simulator is extended with circuit simulator capabilities and can reuse circuit simulator models on demand.

## 1.6 Convergence

The system of equations which has to be solved for mixed-mode device simulation is non-linear and extremely sensitive to small changes in the solution variables. While the semiconductor equations are difficult to solve themselves the situation becomes even worse when using dynamic mixed-mode boundary conditions. To solve these equations the Newton method is used which is known to have quadratic convergence properties for an initial-guess sufficiently close to the final solution. However, such an initial-guess is hard to construct for both the distributed quantities inside the device and the circuit equations. Hence methods have to be found to enlarge the region of convergence to succeed even with a poor initial-guess. This is achieved by applying suitable damping schemes. One of the most popular damping schemes has been published by Bank and Rose [?, ?]. In MINIMOS-NT a purely heuristic method is used which takes the exponential relation between the potential and the carrier concentration into consideration [?]. This method provides similar convergence properties to the method of Bank and Rose without costly evaluation of damping parameters.

Especially important is a reliable method to obtain a DC operating point which is needed as a starting point for a subsequent transient analysis or a static transfer characteristic. Transient simulations are far better conditioned as the time derivatives provide main-diagonal entries and act as a natural damping. As the solution of the last timestep provides a good initial-guess it is normally possible to obtain convergence for a sufficiently small timestep. Although the conditioning of the equation system does not change for DC transfer analysis the last solution again provides a good initial-guess. In case the system fails to converge for a given step the step can normally be reduced in such a way to obtain convergence. Hence the following discussion will focus solely on DC operating point calculation.

To the best knowledge of the authors no useful damping scheme for mixed-mode has been published so far. Only in [?] it was stated that the change of the node voltages was limited to a user-specified value which is in the range of  $2 \cdot V_T$ . This is, as pointed out in the very same paper, far from being optimal as



it requires a large number of iterations for larger supply voltages. E.g., for the OpAmp circuit simulated in the examples section the supply voltages are  $\pm 15$  V, hence it takes at least  $15/0.05 = 300$  iterations to build up the supply voltages without even considering the effect of non-linearities. Furthermore it is stated in [?] that a solution can only be obtained for an initial-guess as close to the solution as  $\pm 0.2$  V for forward-biased junctions.

These restrictions of mixed-mode simulations seem to be generally accepted nowadays. Experiments with a new method delivered promising results for small circuits, the main field of application of mixed-mode simulations. This method is based on the idea, that the distributed devices should be carefully embedded into the rest of the circuit during evolution of the operating point. Similar observations were made by Ho *et al.* [?] for FET circuits using compact models. They proposed to shunt a resistor of 3 k $\Omega$  at the source and drain during the first three Newton iterations to stabilize the coupled system and to slightly decouple the device from the circuit equations. This approach has been extended by introducing an iteration dependent conductance  $G_S^k$  between each device node and ground. The following purely empirical expression for  $G_S^k$  delivered very satisfying results

$$G_0 = 10^{-2} S \quad (1)$$

$$G_{min} = 10^{-12} S \quad (2)$$

$$G_S^k = \max(G_{min}, G_0 \cdot 10^{-k/\kappa}) \quad (3)$$

$$\kappa = 1.0 \dots 4.0 \quad (4)$$

with  $k$  being the iteration counter. It is worthwhile to note that the algorithm worked equally well with  $G_{min} = 0$  for the simulated circuits. However, this expression is purely empirical but unfortunately any attempt to use a more rigorous expression based on norms of the quantities did not work satisfactory.

Using this new technique, solutions could be found for several typical analog and digital circuits starting from the zero initial-guess for the node voltages and charge neutrality assumptions for the semiconductor devices within 20–50 iterations which is a comparable effort to **Spice** which uses compact models.

## 1.7 Examples

### 1.7.1 Five-Stage CMOS Ring Oscillator

A five-stage CMOS ring oscillator circuit is shown in Fig. ?? . For both the NMOS and the PMOS transistors a device width of  $W = 1 \mu\text{m}$  was assumed. Normally, to achieve equal noise margins, a ratio of  $W_p/W_n \approx 2.5$  is used to compensate for the poorer performance of the PMOS transistor [?]. To model the influence of the interconnect circuitry, an additional load capacity of 5 fF was used. To force the circuit into a predefined initial state, the input voltage  $\varphi_{in}$  of the first inverter was set to zero during operating point calculation.

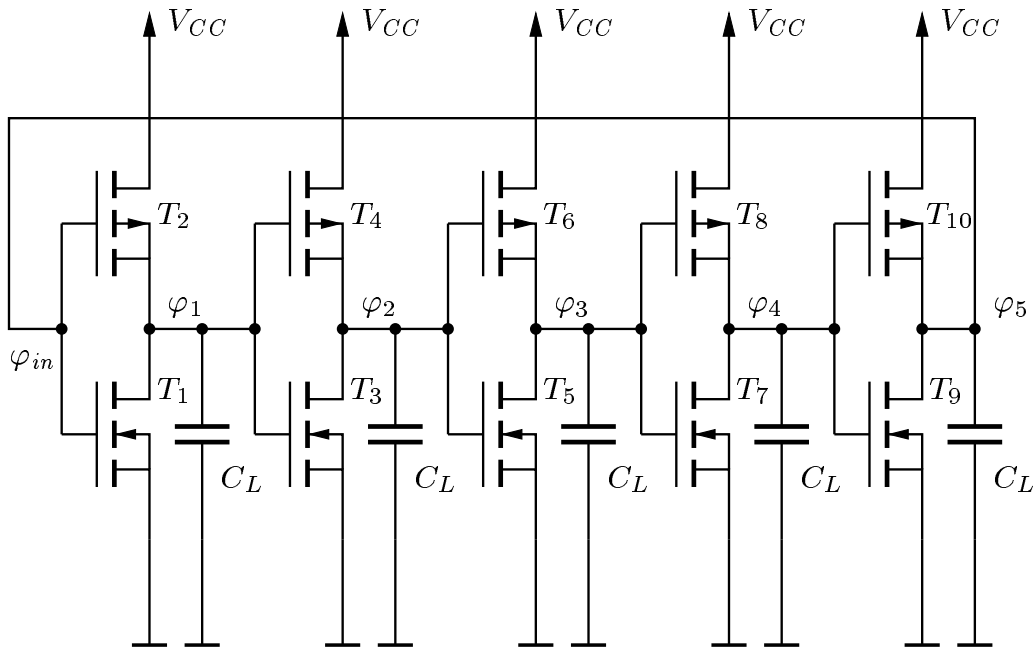
Two different ring oscillators have been simulated, one with long-channel transistors ( $L_G = 2 \mu\text{m}$ ), the other one with short-channel transistors ( $L_G = 0.2 \mu\text{m}$ ). For the long-channel transistors, the simulation results obtained with the drift-diffusion and hydrodynamic transport models agree so closely, that in the graph no differences are visible (cf. Fig. ??). The simulation results for the short-channel devices are shown in Fig. ?? . Here, the differences between the transport models are significant. This is due to the larger currents resulting from the hydrodynamic transport model as the charging and discharging times of an inverter chain are inversely proportional to the drain currents. The simulated inverter delay times are  $\tau_{DD} \approx 30$  ns and  $\tau_{HD} \approx 26$  ns giving a difference of about 15 %. For single devices the hydrodynamic currents are approximately 30 % and 5 % higher for the NMOS and the PMOS transistor, respectively. The average of these values (17.5 %) closely corresponds to the simulated delay time difference of 15 %.

### 1.7.2 Five-Stage CML Ring Oscillator

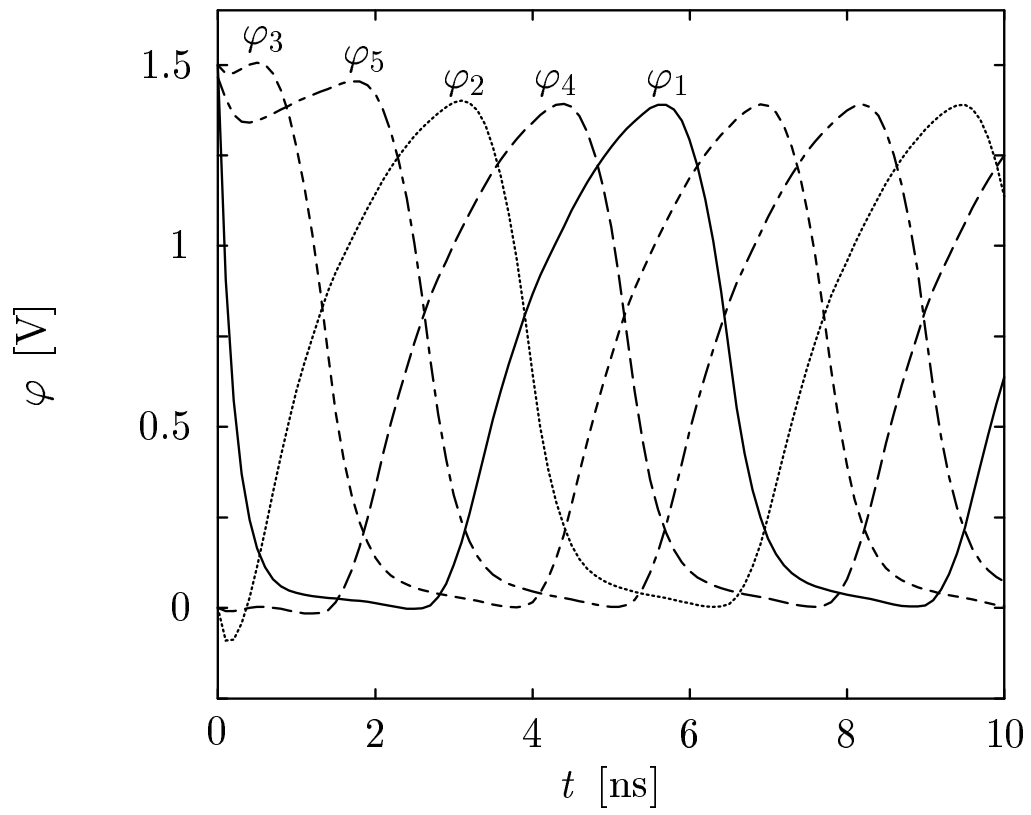
A current mode logic (CML) gate is an emitter coupled logic (ECL) gate stripped of the emitter-follower [?, ?]. The gain of a single stage without load can be approximated by assuming a simple Ebers-Moll model for the transistors [?] to be approximately  $-5$ . When considering an inverter chain consisting of 5 CML inverters as shown in Fig. ?? the total gain occurring at the last output node is  $(-5)^5 = 3125$ . With such a high gain, the circuit is too sensitive to the voltage changes occurring during iteration such that no solution can be found without a proper initial-guess using conventional techniques. However, using the shunt conductance technique with  $\kappa = 4$  a DC operating point was easily obtained with only 34 iterations. As for the CMOS ring oscillator there is no unique operating point for the closed-loop and one of the node voltages had to be fixed to force the circuit into an initial state from which oscillations can start. Oscillations start immediately with a frequency  $f_{DD} = 6.8$  GHz for the drift-diffusion and  $f_{HD} = 10.6$  GHz for the hydrodynamic model which gives a relative difference of 36% for the drift-diffusion model (Fig. ??). This is due to the velocity overshoot which occurs in the base-collector space charge region which cannot be modeled using a drift-diffusion transport model. The current levels are approximately equal in both cases.

### 1.7.3 Electro-Thermal Analysis of a Complete OpAmp

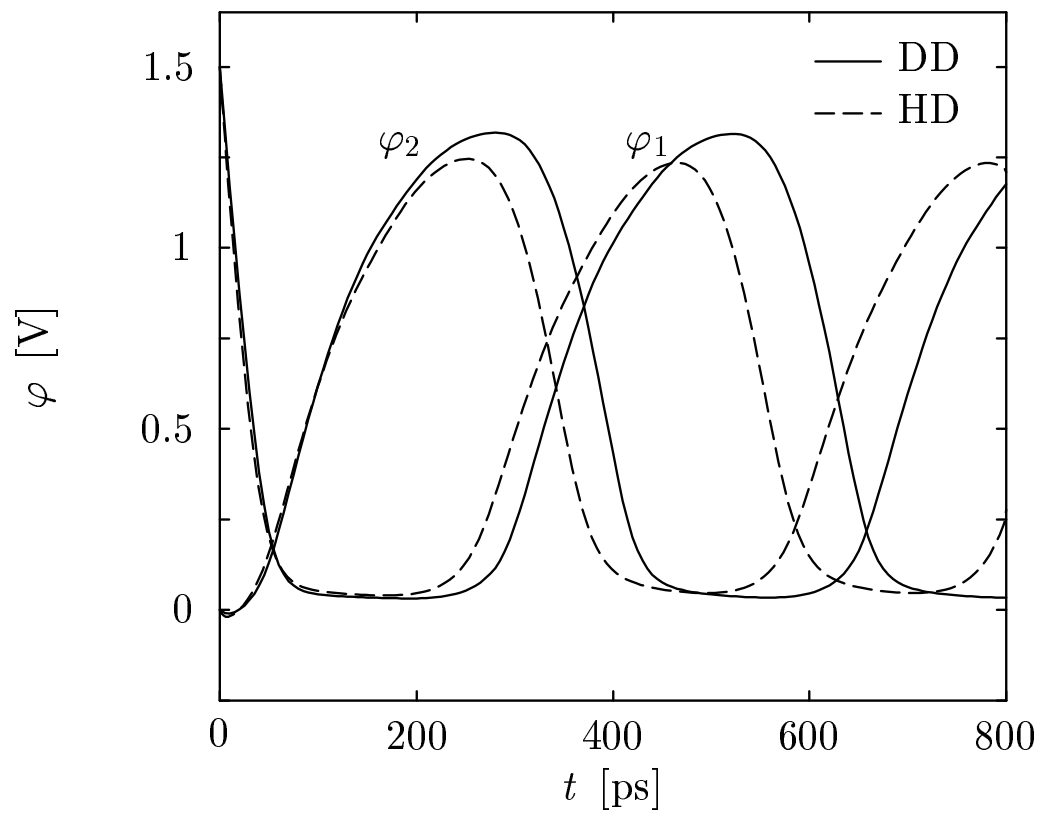
Thermal effects are of fundamental importance for the chip design of integrated circuits. Typical operational amplifiers (OpAmps) can deliver powers of 50–100 mW to a load, and as the output stage internally dissipates similar power levels the temperature of the chip rises in proportion to the dissipated output power [?, ?]. As the transistors are very densely packed, self-heating of the output stage will affect all other transistors. This is especially true as silicon is a good thermal conductor, so the whole chip tends to rise to the same temperature as the output stage. However, small temperature gradients develop across the chip with the output stage being the heat source. The temperature coefficient of the junction voltage for



**Figure 3:** Five-stage CMOS ring oscillator



**Figure 4:** Node voltages of the long-channel five-stage CMOS ring oscillator. Drift-diffusion and hydrodynamic simulation results match perfectly.

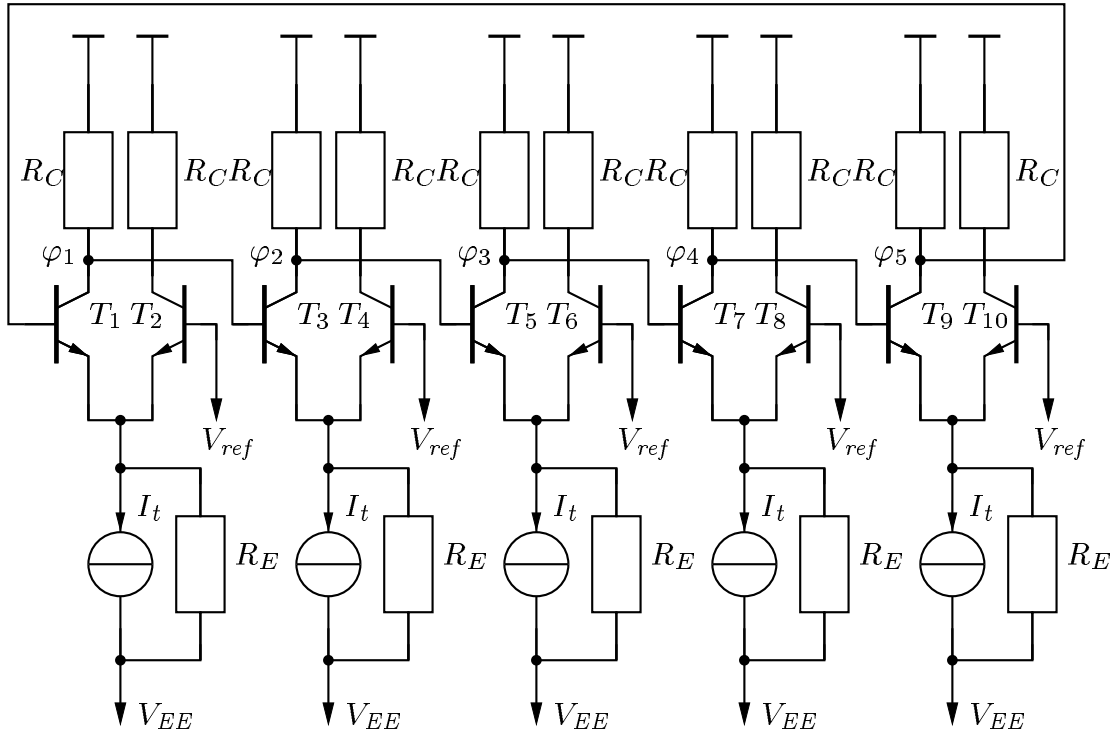


**Figure 5:** Node voltages  $\varphi_1$  and  $\varphi_2$  of the short-channel five-stage CMOS ring oscillator for drift-diffusion and hydrodynamic simulation.

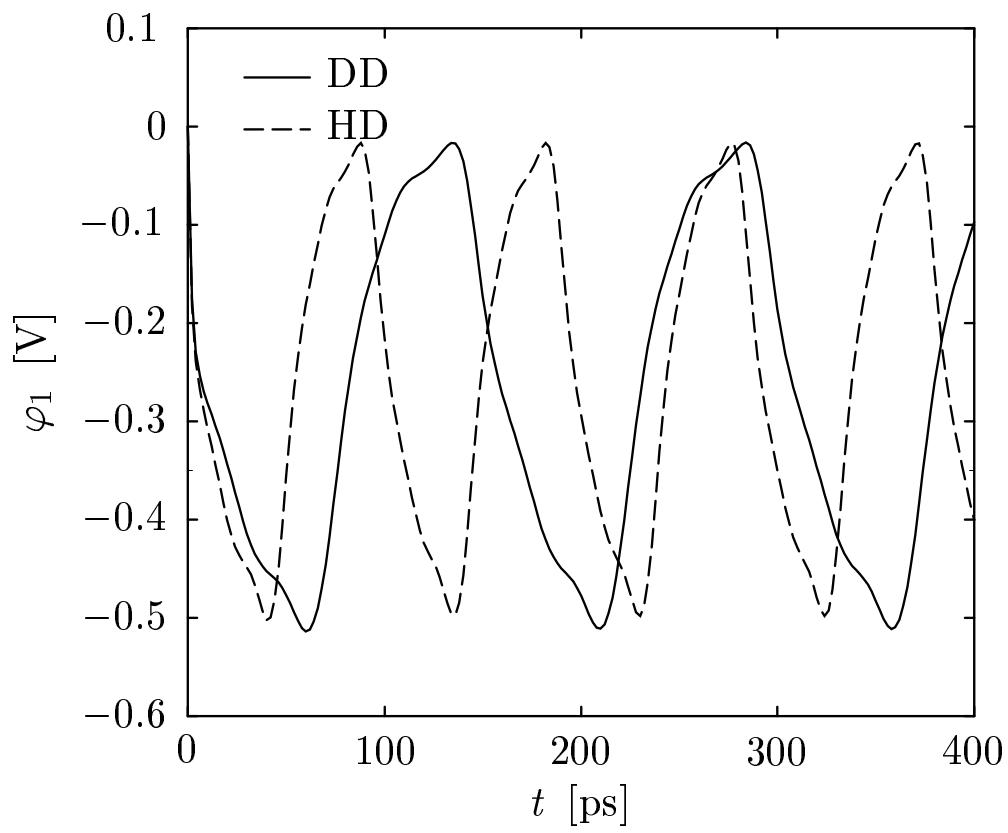
forward-biased pn-junctions is known to be approximately  $-2 \text{ mV/K}$ , that is to obtain the same current a smaller junction voltage is needed. These temperature gradients appear across the input components of the OpAmp and induce an additional input voltage difference which is proportional to the output dissipated power.

The complete  $\mu\text{A709}$  [?, ?] as shown in Fig. ?? has been simulated considering thermal interaction between the input and the output stage. This circuit is of special interest as it is one of the **Spice** benchmark circuits given in [?] (without thermal feedback). The DC transfer characteristic has been calculated with and without thermal interaction. Consideration of thermal interaction was done by solving the lattice heat flow equation for the transistors  $T_1, T_2, T_9$  and  $T_{15}$  and by assuming a thermal network which provides for the thermal coupling of the devices as shown in Fig. ?. The thermal conductances were assumed to be  $G_1 = G_2 = 2 \text{ mW/K}$  and  $G_9 = G_{15} = 10 \text{ mW/K}$  while the coupling mismatch was modeled by  $G_{1,9} = G_{1,15} = G_k = 10 \text{ mW/K}$  and  $G_{2,9} = G_{2,15} = G_k \cdot (1 - \Delta)$  with  $\Delta = 0.9$  being the mismatch parameter which is proportional to the temperature gradient across the input transistors [?].

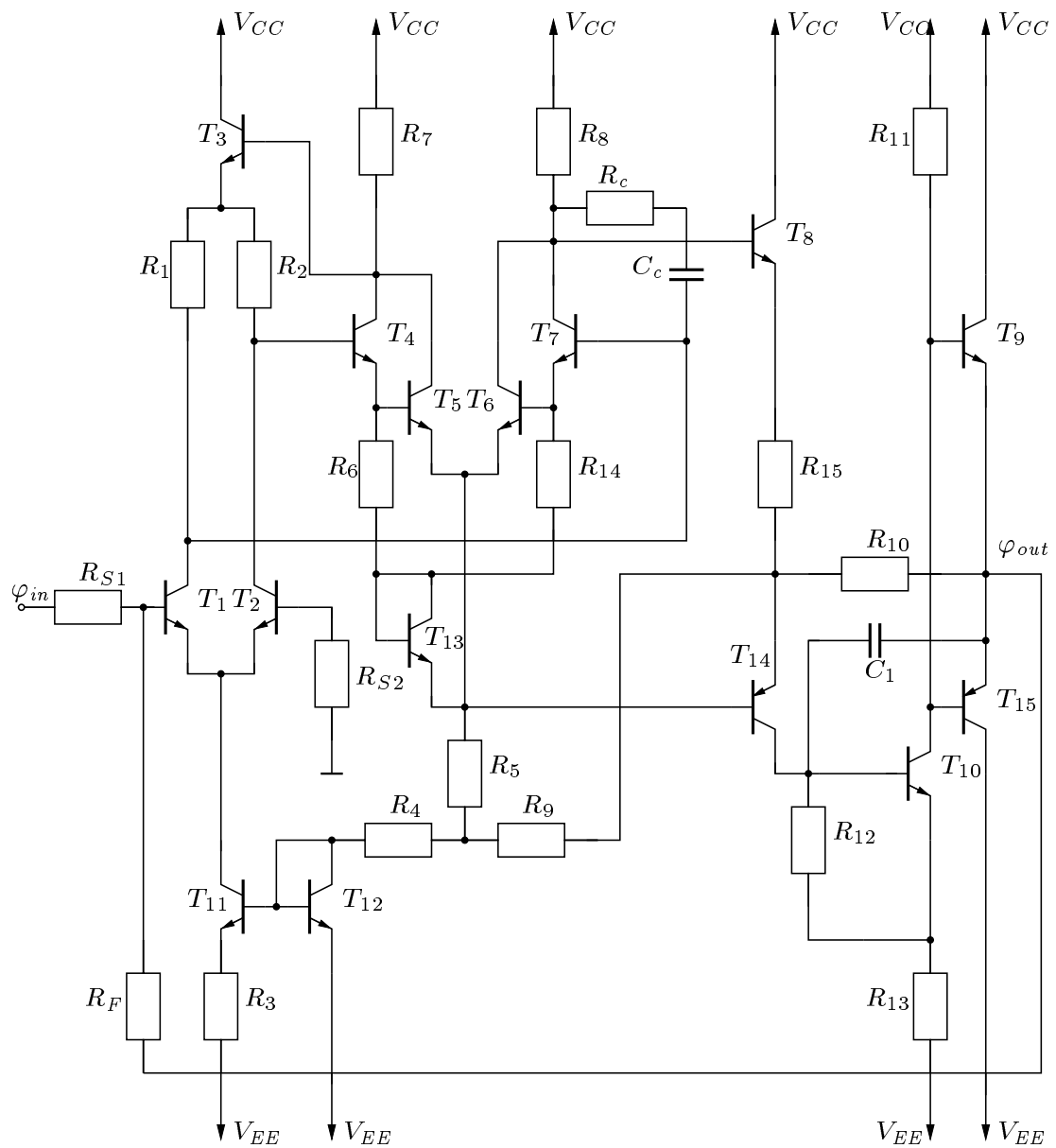
The solution of the fully coupled equation system is possible with a proper iteration scheme. A small change in the output voltage during iteration causes a large change in the collector current of the conducting output transistor. The dissipated power changes which influences the temperature distribution inside the output transistor. This modified power alters the base-emitter voltages of the input transistors which produces a change in the base-emitter voltages of the output transistors. As all these coupling mechanisms are highly non-linear a special iteration scheme is used. In the first block the thermal quantities were ignored until an electrical solution was found. In the second block, the lattice temperature was added to the solution vector without considering the coupling effects caused by the node temperatures. This was also found to be advantageous when stepping through the DC transfer curve hence this block was also used for the consecutive steps.



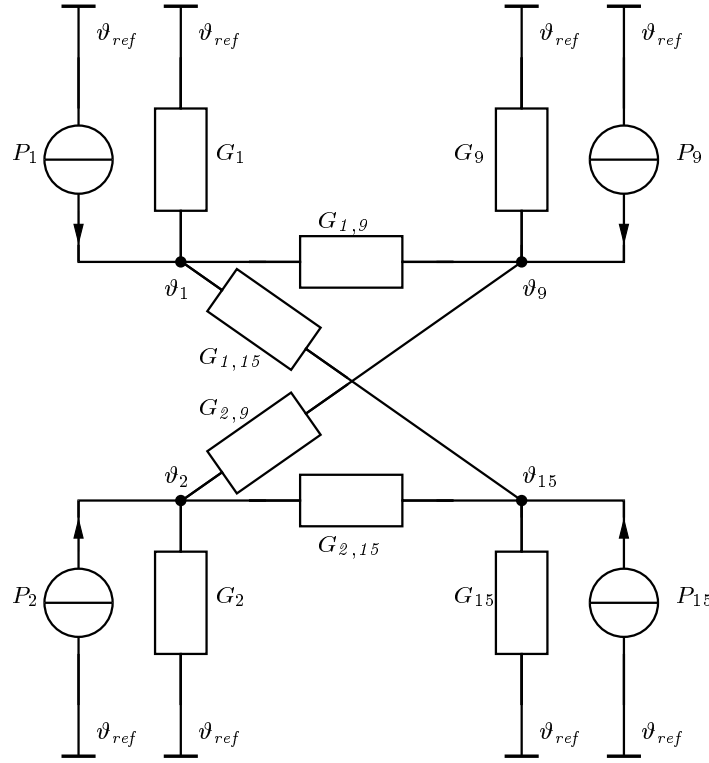
**Figure 6:** Five-stage CML ring oscillator



**Figure 7:** Oscillation of node voltage  $\varphi_1$  of the five-stage CML ring oscillator. Large discrepancies between drift-diffusion and hydrodynamic simulations are observed.



**Figure 8:** Schematic of the  $\mu A709$  OpAmp.



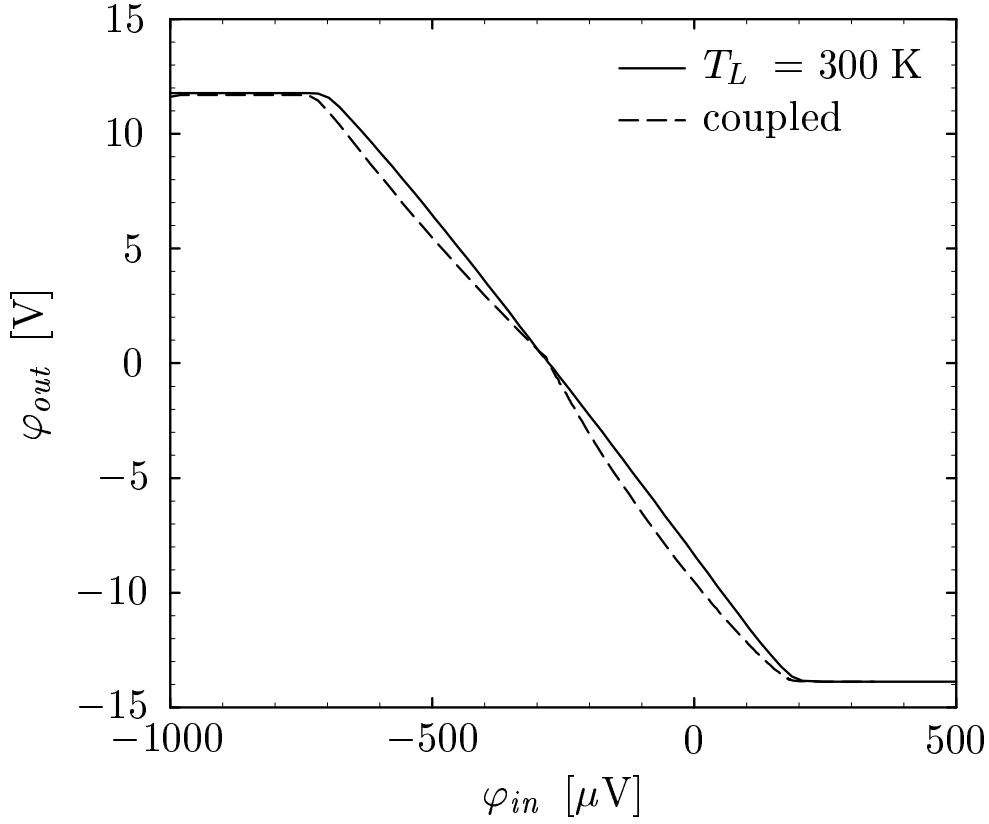
**Figure 9:** Thermal equivalent circuit used to simulate thermal interaction for the  $\mu\text{A709}$  OpAmp.

However, as the condition of the transient problem is much better, this block is not used for transient simulation. Only after a proper temperature distribution inside the devices has been established for the new voltage boundary conditions, can the complete equation system be used.

However, as the simulation failed very frequently for too large steps of the input voltage an additional failure criterion was added. When the step of the input voltage was too large it caused oscillations in the solution which, due to the strong non-linearities, blew up the lattice temperatures. This took approximately 30 iterations which were very expensive in computational terms as each iteration took approximately 20–200 seconds depending on the condition of the system matrix. So this event had to be detected as soon as possible. It was found that an abnormal behavior of the potential update norm  $E_\infty(\mathbf{u}_\psi)$  was a good indication of starting oscillations. Hence, whenever  $E_\infty(\mathbf{u}_\psi)$  was larger than approximately  $10^2 \cdot V_T$  after 10 iterations or whenever  $E_\infty(\mathbf{u}_\psi)$  exceeded  $10^5 \cdot V_T$  the iteration was canceled. Furthermore, the number of iterations was limited to 30.

The DC transfer characteristic was calculated by stepping  $\varphi_{in}$  from  $-1$  mV to  $1$  mV with  $\Delta\varphi_{in} = 20$   $\mu\text{V}$ . From Spice simulations the open-loop gain of the  $\mu\text{A709}$  was known to be approximately 35000 so for each step of  $\Delta\varphi_{in}$  a step of  $0.7$  V could be expected for  $\Delta\varphi_{out}$  which is quite large. However, no convergence problems occurred until  $\varphi_{out}$  approached  $0$  V. This was the most critical part of the simulation and several step reductions were necessary for both the pure electrical and the thermal simulation. The size of the system matrix was 37177 and 40449 for constant temperature and thermal simulation, respectively, and the simulation took 9 and 25 hours on a Linux Pentium II 350MHz workstation. For the thermal simulation the conditioning of the system matrix was found to be very poor and several step reductions were necessary.



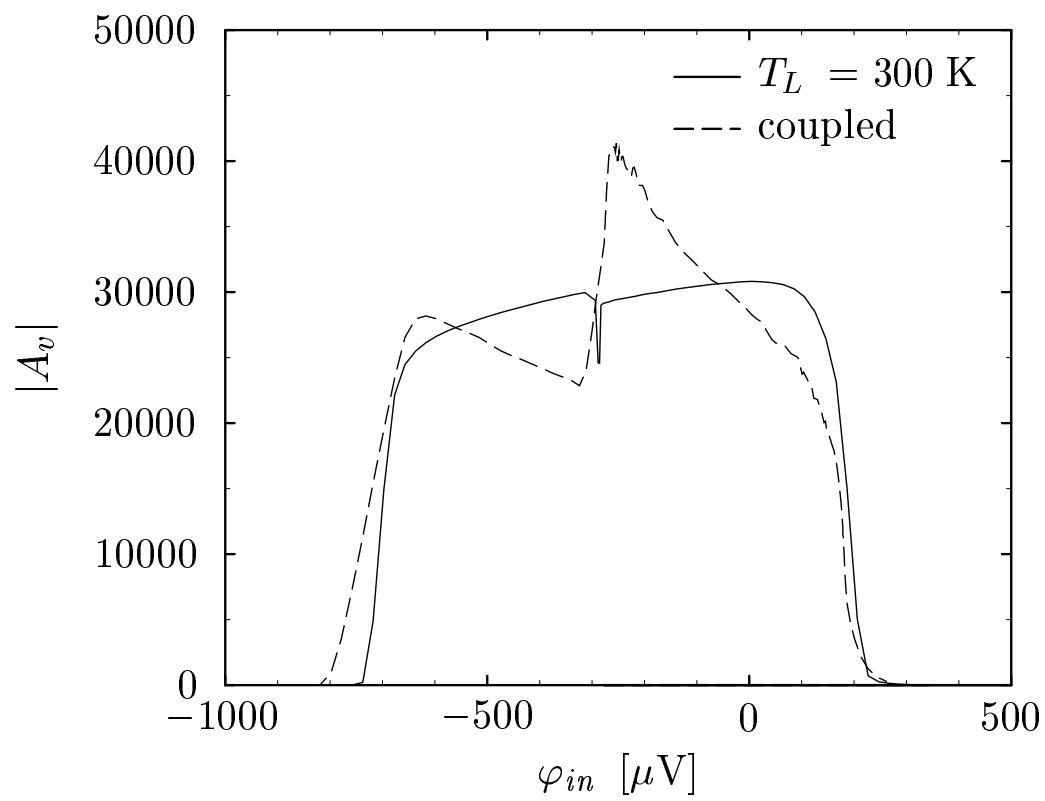


**Figure 10:** DC transfer characteristic of the  $\mu\text{A709}$  for constant lattice temperature and considering thermal coupling of the input and output stage.

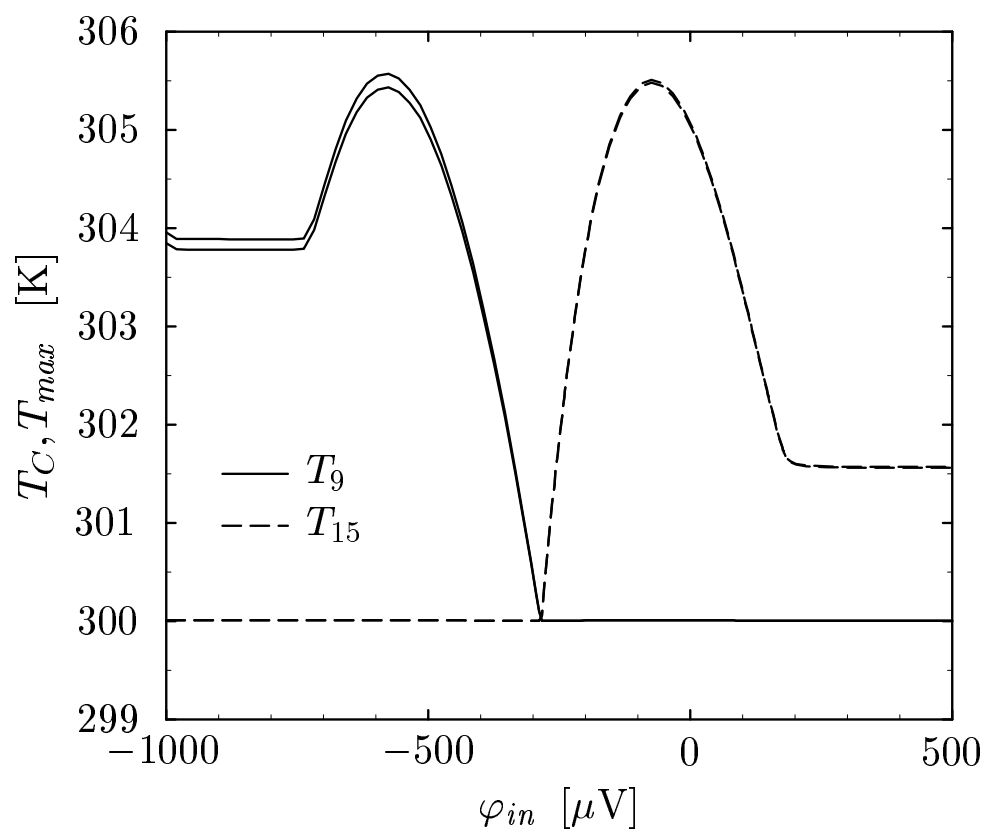
The DC transfer characteristic is shown in Fig. ?? with the obvious humps resulting from thermal feedback effects. In Fig. ?? the open-loop voltage gain  $A_v$  is shown and the dramatic impact of thermal coupling. The thermal conductances assumed in this simulation were very optimistic and an even stronger impact of thermal coupling has been published [?, ?]. For stronger coupling, even the sign of the open-loop voltage gain may change and cause the OpAmp to become unstable [?].

The maximum temperature and the contact temperature of the output stage are shown in Fig. ?. It is obvious that the self-heating inside the transistor plays only a minor role at these current levels. However, the power dissipated inside the device heats up the NPN transistor due to the resistive thermal boundary condition which obstructs the heat flow out of the transistor. This is in accordance to the commonly used assumption that the transistor can be modeled by a power source alone. The PNP transistor has only a  $\beta$  of approximately 10 and comparable current levels have been obtained by increasing the emitter area of the transistor ( $W_{PNP}/W_{NPN} = 5$ ). Hence the locally generated heat density  $H$  is even smaller than for the NPN transistor and the temperature drop inside the device is negligible.

A similar situation occurs for the input transistors  $T_1$  and  $T_2$ . As they are biased with  $I_C = 20 \mu\text{A}$  only self-heating is negligible and the contact temperature resembles the heat transferred from the output stage. As unsymmetric thermal conductivities have been assumed the temperature of  $T_1$  is always slightly higher than the temperature of  $T_2$ . The maximum temperature difference  $T_2 - T_1$  was found to be only  $-22 \text{ mK}$ . Even this small temperature difference has a strong impact on the output characteristic due to the high gain of the circuit.



**Figure 11:** Open-loop gain of the  $\mu\text{A709}$  for constant lattice temperature and considering thermal coupling of the input and output stage.

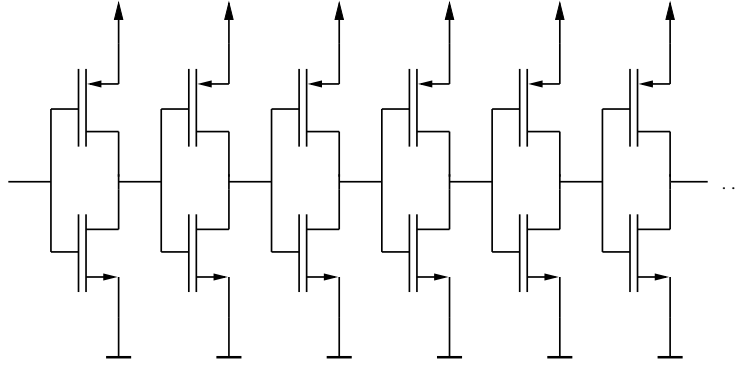


**Figure 12:** Maximum and contact temperature of the output transistors  $T_9$  and  $T_{15}$  during the DC transfer characteristic.

## 2 Closed-Loop CMOS Gate Delay Time Optimization

### 2.1 Introduction

Increasing speed and reducing standby power are the key challenges of the ever growing portable electronics market. Reducing the average gate delay of a CMOS inverter chain, as shown in Fig. ??, while keeping the leakage current low means increasing the speed of the whole technology without changing the standby power.



**Figure 13:** Infinite CMOS inverter chain

Usually, the device geometry and the supply voltage are fixed for a given technology, therefore the key challenge lies in an optimized doping profile which will be the scope of this work.

### 2.2 Optimization Procedure

In order to evaluate the average gate delay time of an infinite inverter chain, an adequate model for one single stage has to be found (Fig. ??). It consists of a CMOS inverter and a capacitive load  $C_L$  connected to the output which accounts for the gate capacitance of the following stage. Since this capacitance changes during transition, it is assumed to be voltage dependent. It can be calculated using the input current information of the succeeding stage.

$$C_L(V) = \left. \frac{I_{in}(t)}{dV_{in}(t)/dt} \right|_{V_{in}(t) = V} \quad (5)$$

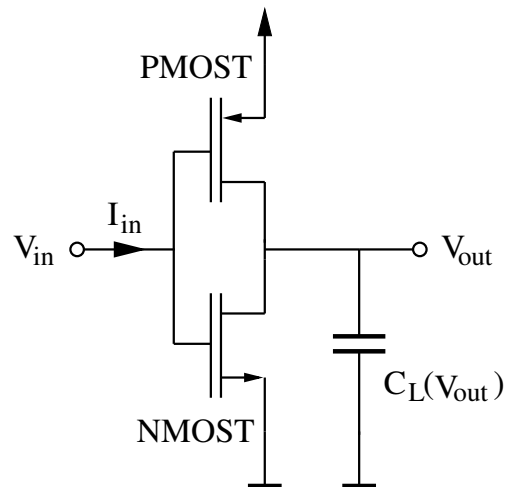
An optimizer drives the closed-loop optimization procedure [?]. The optimization target which will be minimized during optimization, is defined as the average inverter delay time for the on- and off-transitions:

$$\text{target} = \frac{(t_{d,on} + t_{d,off})}{2} \quad (6)$$

The optimization constraint which is kept above zero, guarantees that the average leakage current stays below 1pA:

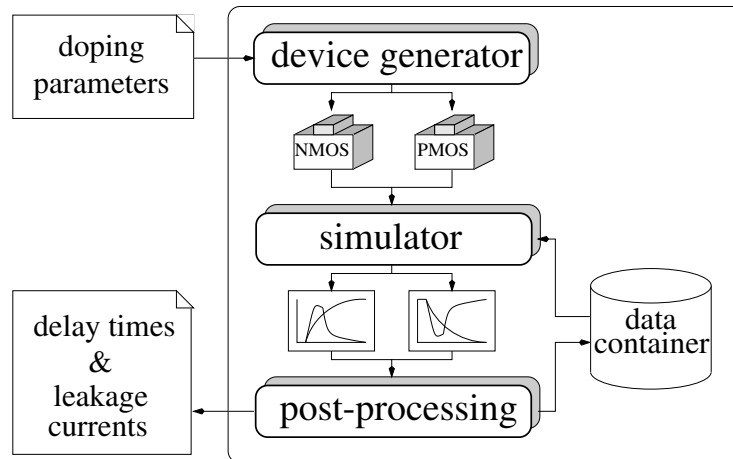
$$\text{constr.} = -\log \left( \frac{(I_{l,on} + I_{l,off})/2}{1 \text{ pA}} \right) \quad (7)$$

The model for the inverter delay times and the static leakage currents is shown in Fig. ?. After reading a given set of doping parameters, the device description of the NMOS and PMOS transistors are produced.



**Figure 14:** Single stage inverter model

Then the inverter model depicted in Fig. ?? is evaluated by transient simulations for both the on- and off-transitions. Additional input data for the simulator, besides the device descriptions, are the input  $V(t)$  curves and the  $C(V)$  curves of the capacitive load  $C_L$  which are taken from a data container. Using the resulting output  $V(t)$  and input  $I(t)$  curves of the inverter, the delay times and leakage currents are calculated. The processed input  $V(t)$  and  $C(V)$  curves for following model evaluations are stored in the data container.



**Figure 15:** Delay and leakage model

Fig. ?? shows the optimization sequence of this procedure. Any time a temporary minimum is found after a number of evaluation steps, a gradients calculation is launched to find the Jacobian matrix for the optimization parameters, and then the next temporary minimum is searched. After each temporary minimum step, the output curves are stored in a wait-state and will be transferred into the data container after the gradients calculation is finished. This permanent update of the data container provides a self-contained emulation of an infinite inverter chain, since output voltage curves and input currents will be used for input curves and load capacitance evaluations for the next steps, repeatedly.

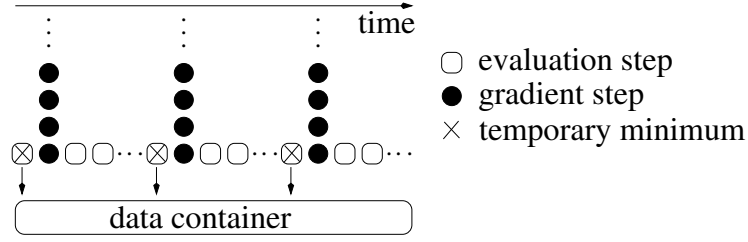


Figure 16: Optimization sequence

### 2.3 Doping Profile Optimization

The optimization procedure was performed on MOSFETs with  $0.25\mu\text{m}$  gate length,  $1\mu\text{m}$  gate width, and  $5\text{nm}$  gate oxide thickness for  $1.5\text{V}$  supply voltage. The source/drain doping profiles stayed fixed during optimization with a maximum doping of  $10^{20}\text{cm}^{-3}$  and about  $50\text{nm}$  junction depth.

Two different methods were used to obtain a set of optimization parameters which define the doping profiles in the active regions of the two transistors: A general two-dimensional approach using an optimization grid, and an approach with implantation models. The simulator MINIMOS-NT [?] was used for all simulation tasks since it supports the coupling between circuit simulation with compact models and numerical device simulations consistently.

#### 2.3.1 Two-Dimensional

An optimization grid was chosen with the shape of an inverted “T” to cover all regions which might influence the device behavior. Fig. ?? shows the optimization grid and the source/drain wells. The doping at each grid point is defined by one optimization parameter, therefore 124 parameters, 62 for each device, are required. Between the grid points, an interpolation method was used to provide a smooth two-dimensional doping profile. Outside the optimization region the substrate doping was kept at  $10^{15}\text{cm}^{-3}$ .

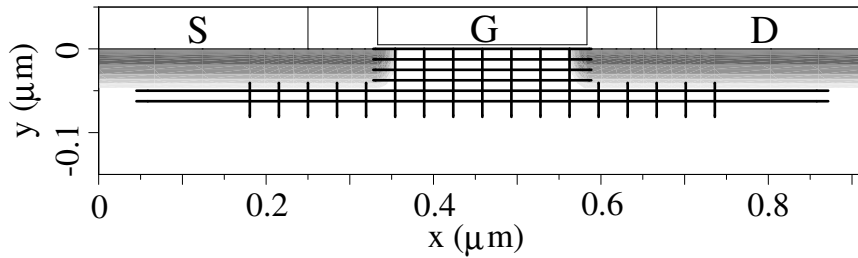
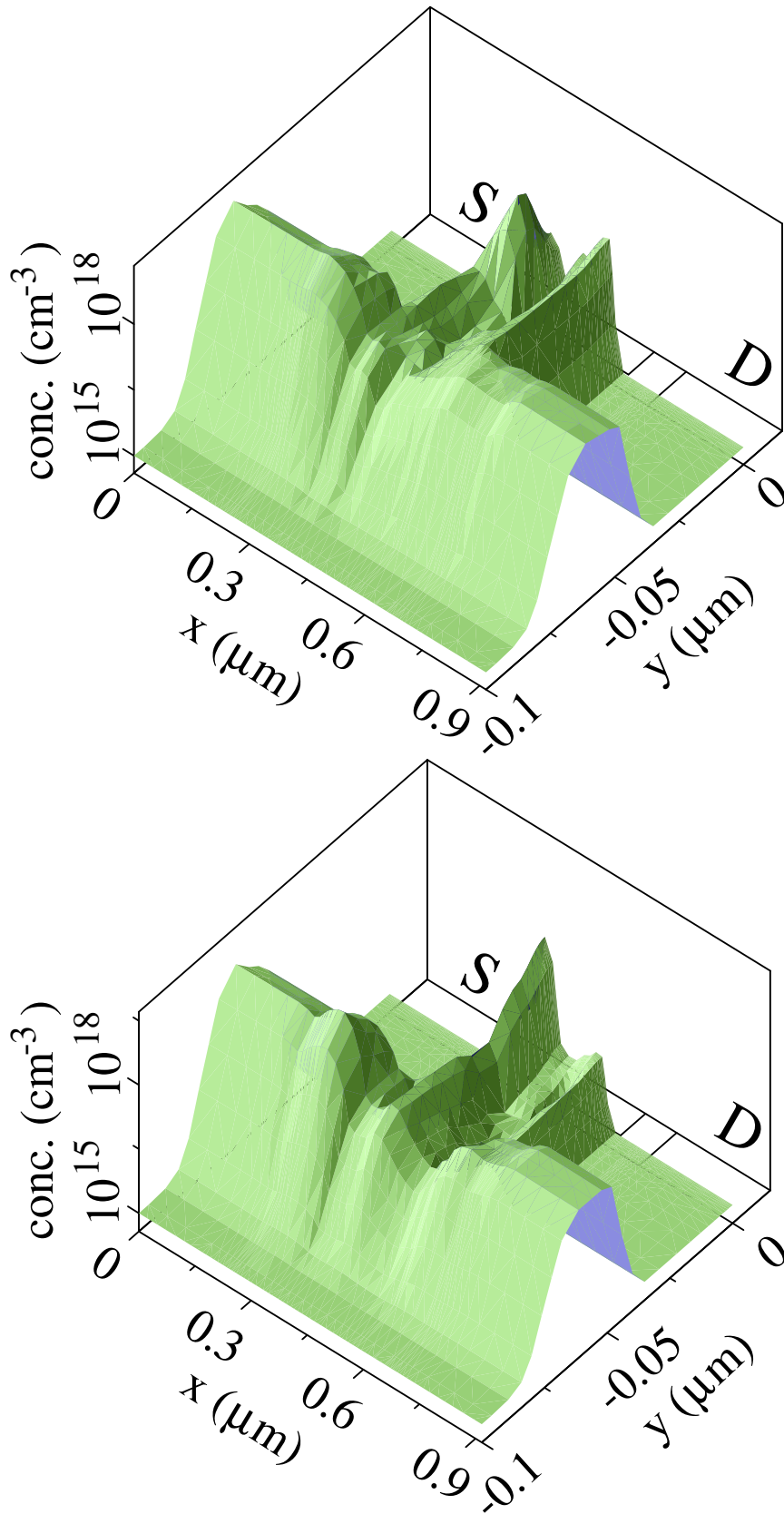


Figure 17: Optimization grid and S/D wells

Fig. ?? shows the resulting doping profiles. The average gate delay time was reduced by 58%, from  $82.1\text{ps}$  to  $34.8\text{ps}$  compared to the initial device with a uniformly doped “inverted-T” region.

#### 2.3.2 Implantation Models

The use of Gaussian implantation models allows for a considerable reduction of the number of doping parameters and, therefore, for a faster optimization procedure. Additionally, the results from the two-



**Figure 18:** Two-dimensional optimization results, top: NMOST, bottom: PMOST

dimensional approach, which look quite complex due to the numeric origin of the optimization procedure, can be tailored to more realistic profiles.

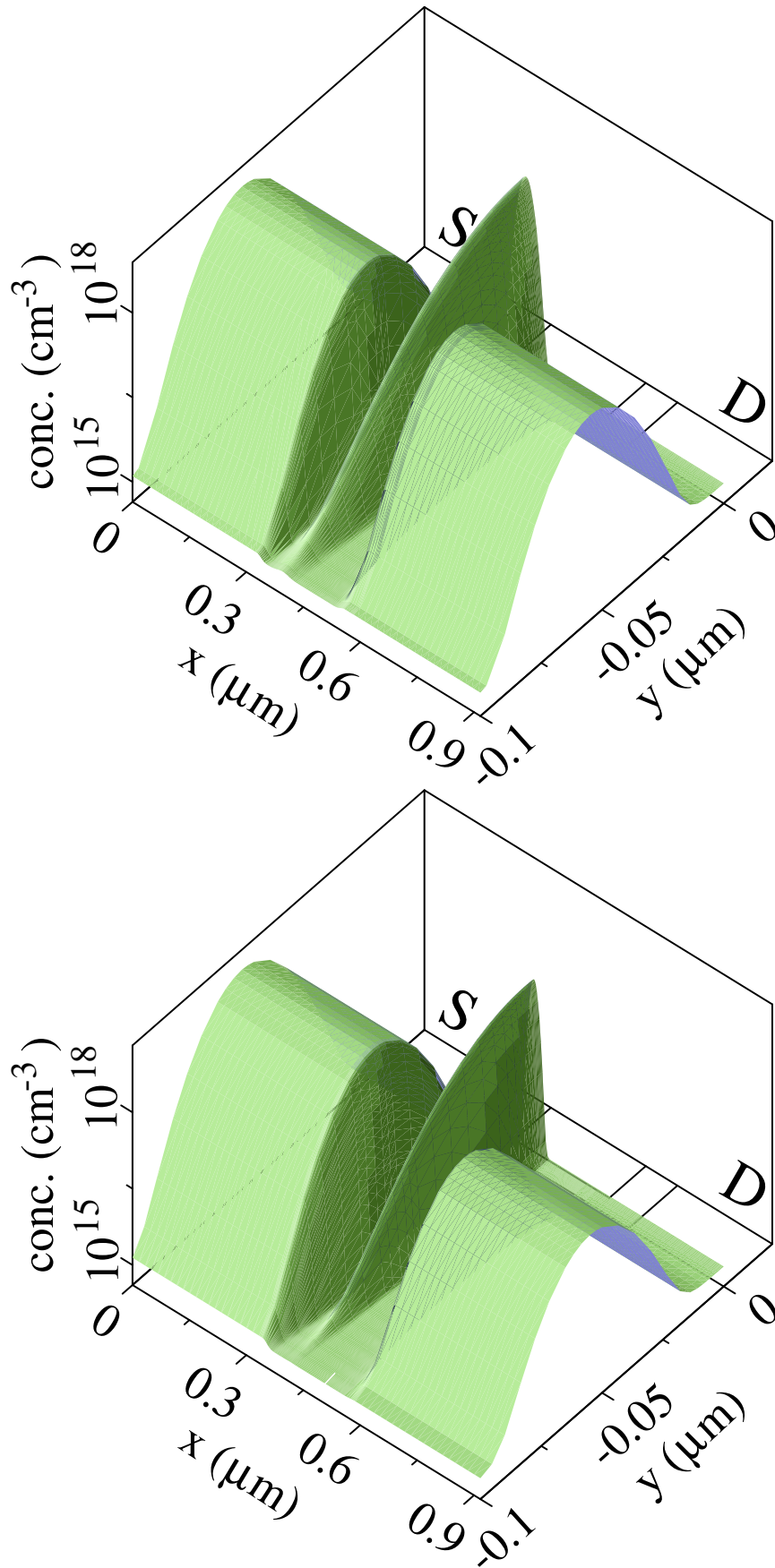
Three implantations were used for the channel region. The first one is located in the channel close to the source well. It sets the threshold voltage of the device and reduces the effective gate length. The second and third implants are located deeper, under the source and drain wells, respectively. They improve the short channel effects and work as a shield against deep punchthrough. Fig. ?? shows the resulting doping profiles. The delay-time reduction with this method is still 55%.

## 2.4 Discussion

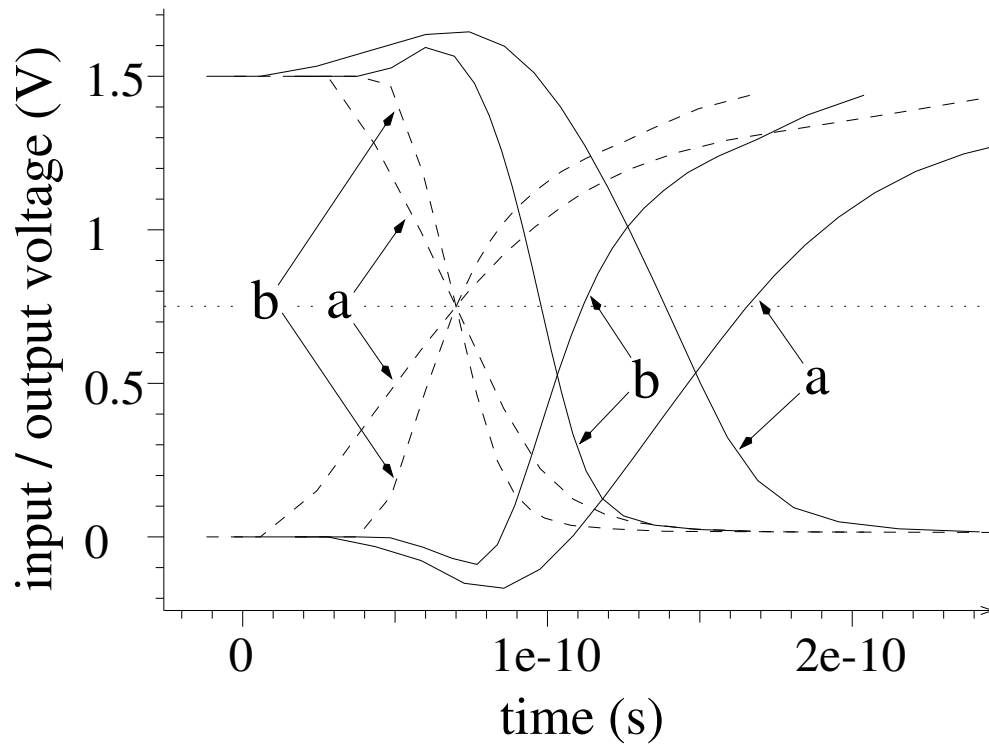
The optimized doping profiles are similar to the results obtained by previous work where only the drive current of a single NMOS transistor was optimized [?]. Now the regions under source/drain are of increased importance because of the source/drain well capacitances.

Fig. ?? shows the inverter input/output curves before and after optimization for both transition cases. For optimization, or rather simulation, reasons, the transition time point of the input curves, defined at 50% of the supply voltage, was kept constant. The delay time for the output-on transition is higher than for the output-off transition because the optimizer kept both leakage currents at about the same value of 1pA. Therefore, the PMOS transistor delivers a lower drive current due to the lower majority carrier mobility.





**Figure 19:** Implantation models optimization results, top: NMOST, bottom: PMOST



**Figure 20:** Input (dashed) and output (solid) curves before (a) and after (b) optimization

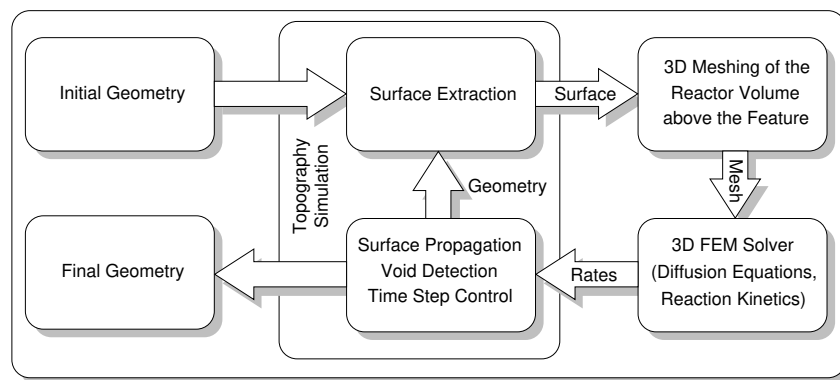
### 3 Modeling of High-Pressure Chemical Vapour Deposition

This section describes a three-dimensional model for the simulation of continuum transport and reaction determined high pressure CVD processes. Our approach allows simulations over arbitrary geometries such as structures resulting from non-uniform underlying PVD films. This enables the examination of film profile variations across the wafer for multi-step processes consisting of low and high pressure parts such as Ti/TiN/W plug-fills. Additionally the model allows a very flexible formulation of the involved chemistry and can easily be extended to arbitrary CVD processes including gas phase reactions of precursors as observed in the deposition of silicon dioxide from tetraethylorthosilicate (TEOS).

#### 3.1 Introduction

A variety of three-dimensional topography simulators provides facet motion, cellular, level-set or Monte-Carlo algorithms in combination with macroscopic models for the simulation of low pressure deposition processes determined by ballistic transport. However, for increasing process pressure leading to diffusion determined mass transfer, up to now only two-dimensional simulations are available [1].

To close the gap of missing high pressure models in three-dimensional topography simulation we have extended the two-dimensional continuum transport and reaction model and present a fully three-dimensional model for the simulation of arbitrary chemistry, multiple species high pressure chemical vapor deposition (CVD) processes. Together with the low pressure models for our simulator presented in [2] all tools for an integrated three-dimensional back-end process simulation including across wafer non uniformities of ballistic transport and diffusion determined processes are now available.



**Figure 21:** Flow diagram for the high pressure CVD model

#### 3.2 CVD-Model

As shown in Fig. 1 the model consists of a combination of specialized tools which are called automatically from a controlling instance. After extracting the surface of the initial geometry, a three-dimensional mesh of the gas domain above the considered structure is generated. The differential equations describing the mass transfer and the reaction kinetics are set up and evaluated with a general object-oriented solver which operates on the previously generated unstructured mesh. The resulting deposition rates are transferred to the topography simulator which works on a cellular material representation. The surface propagation for each time step is deduced by applying structuring elements to the actual surface [3]. The size of the structuring elements correspond to the deposition rates calculated with the continuum transport model.



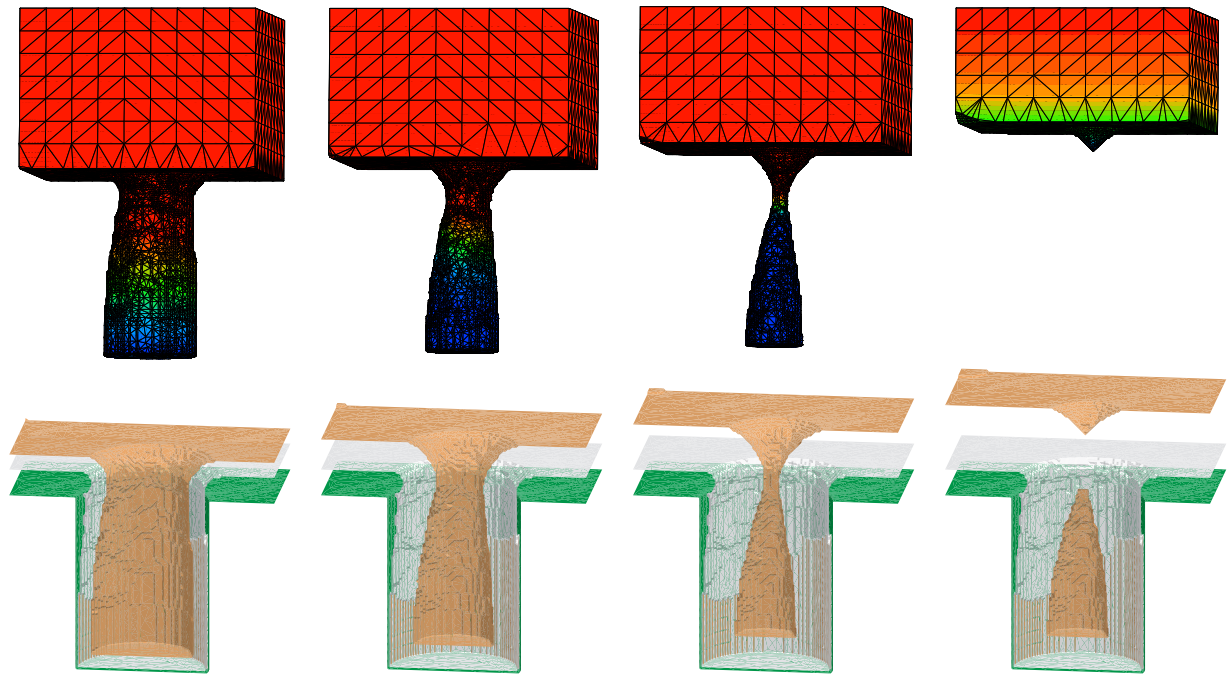
as input for the meshing tool. For the tetrahedralization the dimensions of the gas space above the feature have to be specified and additional points within the volume are inserted. The meshing tool uses a modified advancing front algorithm to generate a three-dimensional unstructured tetrahedral mesh [4].

The chemistry model is set up with AMIGOS [5] which provides an analytic interface for discretizing and solving differential equations. The governing principles for the CVD model are surface reactions  $-D_i \frac{\partial c_i}{\partial n} = R$  and diffusion of gas species in the plasma  $\frac{\partial c_i}{\partial t} = \nabla(D_i \nabla c_i)$ . For the CVD model we only calculate the time independent steady state with  $\frac{\partial c_i}{\partial t} = 0$ . Transient calculations have revealed that the steady state is reached after a few  $\mu s$  and that the steady state assumption for each single time step is correct.

### 3.3 Applications and Results

As shown in Fig. 3 this model is applied to the simulation of tungsten CVD used for a Ti/TiN/W plug-fill process [6]. The chemistry model used for the high pressure CVD process assumes that W is reduced from  $WF_6$  using  $H_2$  and forming HF as by-product. The three gas species diffuse in the feature and the reduction takes place at the feature surface. The assumed process conditions cause a depletion of  $WF_6$  in the feature and a characteristic overhang in the layer profile. The simulated structure is located at an off center position of the wafer. Thus the TiN layer, formed by sputter deposition prior to the tungsten CVD is strongly asymmetric requiring the rigorous three-dimensional simulation of the CVD film formation.

The required CPU time for this example, simulated with a DEC 600/333 workstation is approximately 10 min for each time step of the automatically controlled simulation sequence, including surface extraction, meshing, calculation of the deposition rates, time step control, void detection and surface propagation. Depending on the size of the structure between 10.000 and 30.000 tetrahedra were used for the continuum transport model.



**Figure 23:** The volume meshes used for the continuum transport model and the corresponding three-dimensional film profiles for a sequence of time steps of a Ti/TiN/W plug-fill process. The profiles in the lower row show from bottom to top the initial circular via, the TiN PVD layer formed by sputter deposition and the growing W CVD layer.

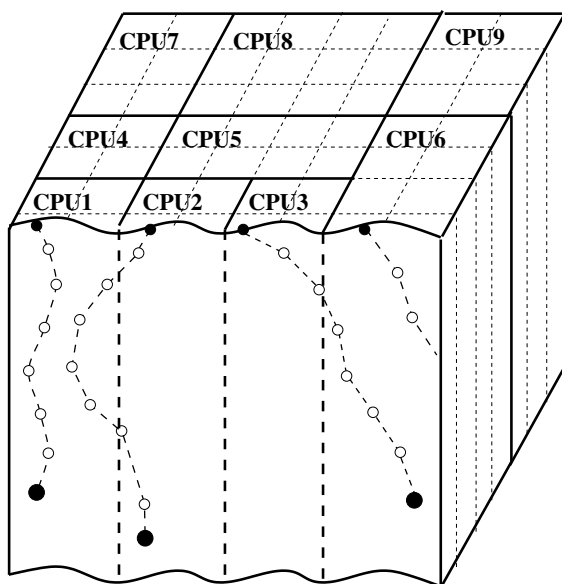
## 4 Parallelization of a Monte-Carlo Ion Implantation Simulator for Three-Dimensional Crystalline Structures

### 4.1 Introduction

When simulating semiconductor production processes, ion implantation is a very important, but also one of the most critical steps, concerning the simulation time. Due to the complicated structures and the small dimensions of modern semiconductor devices, Monte-Carlo simulation methods often have to be used to describe non-planarity effects, phenomena resulting from ion channeling and large tilt angles, and to provide accurate point-defect distributions for rapid thermal annealing processes. To reach the expected accuracy, three-dimensional simulations have to be performed with sophisticated models [?][?], especially for very shallow implantation conditions. By meeting all these requirements the simulation times exceed one night or even more on high-end workstations for large structures. Therefore the parallelization of the Monte-Carlo ion implantation simulation process step is desirable to avoid a bottleneck in the process simulation flow, because normally a cluster of workstations is available to perform process simulation and optimization. We present a parallelization method which allows a distributed simulation on a cluster of single processor workstations. Our parallelization strategy is based on MPI and it allows to reuse all sophisticated methods and models [?][?] developed for the single processor version without modification.

### 4.2 Parallelization Strategy

We use a master-slave strategy, where the master process provides all the I/O operations and controls and synchronizes the behavior of all slaves which perform the actual simulation. The basic concept of the Monte-Carlo ion implantation simulation method is that the trajectories through the simulation domain are calculated for a large number of ions. The final positions of calculated particles and the number of generated point-defects are stored in a histogram which is used to derive the particle and point-defect distributions. The parallelization is achieved by splitting the simulation domain into several rectangular



**Figure 24:** Schematic presentation of the split of the simulation domain into subdomains and of the distribution of the subdomains among several processors.

prismatic subdomains (Fig. ??). Each available CPU is responsible for several of these subdomains. This means that all particles moving through a certain region of the simulation domain are calculated by a certain CPU and that all simulation results are stored in the memory belonging to a certain CPU. Thereby

the trajectories calculated as well as the memory consumption are distributed among several workstations. When initializing the simulation the master process determines the distribution of the subdomains according to the number and the speeds of the available CPUs, considering the following conditions.

$$\frac{V_i}{CPU_i} \simeq const., \forall i \quad (8)$$

$$\sum_i \frac{O_i}{V_i} \rightarrow max \quad (9)$$

$V_i$ ,  $O_i$  are, respectively, the volume and the surface of a prismatic scope belonging to one processor, as denoted in Fig. ?? by the thick lines.  $CPU_i$  is the relative computing capability (e.g, floating point operations per second) of one processor. (??) ensures that every CPU gets a reasonable amount of trajectories for calculation, because a faster CPU can calculate more trajectories than a slower CPU. Due to the fact that the implanted ions are equally distributed over the device surface the number of trajectories that have to be calculated by one CPU is proportional to the volume of the prismatic scope belonging to this CPU, assuming that the same models are used throughout the whole simulation domain.

(??) guarantees a minimum of communication between the CPUs. It is possible that a particle moves to the prismatic scope of another CPU during its motion through the simulation domain. In that case the particle described by its physical and modeling properties has to be exchanged between two CPUs. Due to the fact that such communication events always limit the performance gain of a parallelized application it is desirable to minimize this communication. The probability that a particle leaves the scope of a CPU is all the lower the larger the volume  $V_i$  of the scope, and it is all the higher the larger the interface area  $O_i$  to other CPUs.

In order to take into account the influence of crystal damage on the trajectory of an ion a transient simulation is explicitly introduced. This is not necessary for a single processor version, because in that case the single trajectories are calculated one after the other while in the parallelized version the order of calculation is not deterministic. It is assumed that the ions belonging to the same time step do not influence each other and therefore the order of calculation is of no relevance within one time step. This requirement is met if the number of ions that are calculated per time step is significantly smaller then the total number of calculated ions.

### 4.3 Simulation Flow

First the master parses the command-line, reads the input files and initializes the physical properties of the simulation domain, the physical models and the implantation conditions. The initialization data are sent to all slaves. Then the master creates the subdomains and evaluates a distribution scheme. The subdomains and the distribution scheme are sent to all slaves. Thereby all processes are informed about the scope of responsibility of all other processes too. Whenever a particle trajectory leaves the scope of responsibility of a slave, this slave knows where to send the particle.

After this initialization the master process calculates the initial conditions of the implanted ions and prepares them for being sent to the slaves. When all ions in all subdomains belonging to one time step are prepared one package of ions is sent to each slave. Then the initial conditions of the ions of the next time step are prepared for sending before the master process enters a wait-loop where it determines if all slaves have finished the last time step. The sending, packing and waiting is repeated until the total number of simulated ions is calculated. Afterwards the master process sends a 'Simulation Finished' request to all slaves, collects the information about all simulated ion trajectories, performs the statistical analysis for the resulting doping and point-defect distributions, prepares the generation of the output and writes the output files.



The slaves enter a wait-loop immediately after the initialization process, where they are waiting for a request either from the master or from a slave. Five types of requests are managed.

**(a) Ion Package Request:**

The trajectories of the ions in the ion package which is sent either from the master or from another slave are calculated. During the trajectory calculation simulation data can be stored in or received from the area of responsibility of another slave. Moreover a simulated particle can be moved to another slave by sending an appropriate request.

**(b) Store Simulation Data Request:**

Simulation data that have to be stored in the local memory are received from another slave. Information about the type and the position of the data is received before storing the data.

**(c) Send Simulation Data Request:**

Information about the type and the position of required data is received, and the corresponding simulation data are sent to another slave.

**(d) New Time Step Request:**

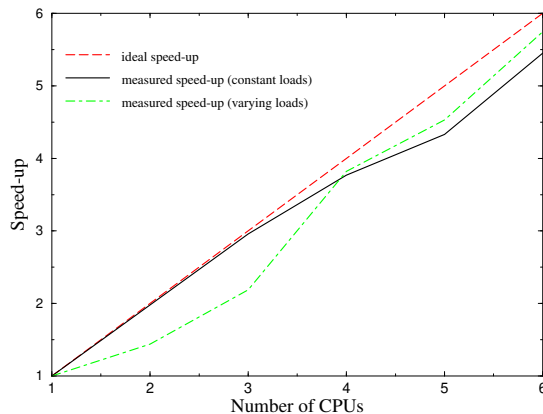
The initialization procedure for a new time step is called.

**(e) Simulation Finished Request:**

While the slave returns to the wait-loop when he has finished one of the above requests he leaves the loop in case of a 'Simulation Finished' request. All simulation results are sent to the master before the slave terminates his operation.

## 4.4 Results

By measuring the performance gain on a network of workstations it has turned out that this parallelization strategy delivers an almost linear performance gain (Fig. ??), but only if the processor load is constant on all CPUs throughout the simulation. In case of varying processor loads the performance gain can decrease dramatically, because the calculation time for all trajectories at one time step is not constant anymore on all CPUs. Thereby one slave always holds up all other slaves due to synchronization at the end of each time step. Additionally it has to be mentioned that the size of one subdomain must be larger than the lateral range of the implanted ions to keep the communication overhead low. This limits the number of available subdomains and thereby the number of CPUs that can be used for a parallel simulation. However for three-dimensional application the number of subdomains is larger than 1000 which normally exceeds the number of available processors.



**Figure 25:** Measured speed-up compared to the ideal speed-up for two different load situations.

## **4.5 Conclusion**

We have presented a method for parallelizing a Monte-Carlo ion implantation simulator with a minimum of communication overhead and therefore an almost linear speed-up. For strongly varying processor loads a dynamic load balancing should be implemented. Besides a tremendous reduction of the simulation time a splitting of the memory requirement is achieved, which allows to utilize several small workstations for the simulation of large three-dimensional problems.

## **4.6 Acknowledgment**

This work has been carried out within the SFB project AURORA, funded by the Austrian Science Fund (FWF).

## 5 PROMIS-NT

PROMIS-NT is a two-dimensional diffusion simulator which is designed to serve as a development platform for diffusion models in the field of TCAD process simulations. The simulator is derived from the diffusion simulator DIFFUS [?] by replacing all previously hard coded functions concerning the diffusion modeling by interfaces to the Algorithm Library. Thereby an external user interface to diffusion modeling within a simulator which is fully integrated into the VISTA TCAD environment [?, ?] has been generated.

### 5.1 Features of PROMIS-NT

Diffusion simulation can be done on arbitrary two-dimensional geometries by using orthogonal or unstructured triangular simulation grids. Orthogonal grids are restricted to geometries with Manhattan type geometries. Built-in grid refinement algorithms are applied to the input data grids to form the initial simulation grids ensuring that some numerical quality criteria for the grids are fulfilled. With user defined algorithms one can induce additional quality criteria when needed.

The geometry of the simulated device can be partitioned into several segments connected by one-dimensional boundaries. The number of segments and boundaries is only limited by the memory resources of the computer used for the simulation. For each of the segments and boundaries different diffusion models can be defined.

The simulator is fully integrated into the VISTA TCAD environment. The input and output format of the device descriptions is the Profile Interchange Format PIF [?, ?]. Therefore simulations with PROMIS-NT can be integrated into a chain of other two-dimensional process simulations within the VISTA environment, or by using the *tifwrap* program [?], also into TMA Suprem-4 [?, ?] process simulation flows.

The predefined set of quantities available for the diffusion simulation which is compatible to the standard set of quantities available within TMA Suprem-4 or VISTA process simulations, can be replaced or extended by an arbitrary number of user defined quantities.

Algorithms determining the initial distribution of each quantity can be defined for each segment of the geometry. Analogously there is the possibility to define algorithms to postprocess the quantity distributions before writing the simulation results into the resulting PIF file.

The profile of the process temperature can be controlled by user defined MDL functions.

### 5.2 Supported Model Structures

To liberate the developer of the physical diffusion model from the task to discretize the corresponding partial differential equation system, PROMIS-NT has been equipped with built-in algorithms to discretize the transport equations (??) and (??) and boundary models (??) and (??), respectively. Thereby the task of implementing new diffusion models is reduced to the development of algorithms which compute appropriate values and derivatives for the needed coefficients of (??) – (??).

#### 5.2.1 Volume Models

Equation (??) shows the structure of the partial differential equation system describing the transport of  $N$  quantities  $W_j$  within a segment of the device geometry.

$$\sum_{j=1}^N \alpha_{ij} \cdot \frac{\partial W_j}{\partial t} + \text{div} \vec{J}_i + \gamma_i = 0, \quad i = 1 \dots N \quad (10)$$

$$\vec{J}_i = \sum_{j=1}^N (a_{ij} \cdot \text{grad } W_j + b_{ij} \cdot W_j \cdot \text{grad } \psi + \vec{c}_{ij} \cdot W_j) + \vec{d}_i \quad (11)$$

$W_j$  denotes the dependent variables which are the values of the affected quantities and  $N$  is the number of equations.  $\psi$  is either the electrostatic potential resulting from (??) or can be chosen to be one of the dependent variables  $W_j$  by a corresponding statement in the input deck.

For a system of  $N$  impurities the total net concentration  $C_{net}$  of all impurities is given by equation (??).  $z_i$  denotes the charge state of the impurity  $C_i$  (0 for neutral impurities, +1 for singly ionized acceptors, and -1 for singly ionized donors).

$$C_{net} = \sum_{i=1}^N z_i \cdot C_i \quad (12)$$

The electrostatic potential  $\psi$  is determined by the Poisson equation (??)

$$\text{div}(\text{grad } \psi) = \frac{q}{\epsilon} \cdot (n - p - C_{net}) \quad (13)$$

By assuming vanishing space charge and the applicability of Boltzmann statistics the Poisson equation can be solved explicitly (??).

$$\psi = \frac{k \cdot T}{q} \cdot \text{asinh} \left( \frac{C_{net}}{2 \cdot n_i} \right) \quad (14)$$

### 5.2.2 Boundary Models

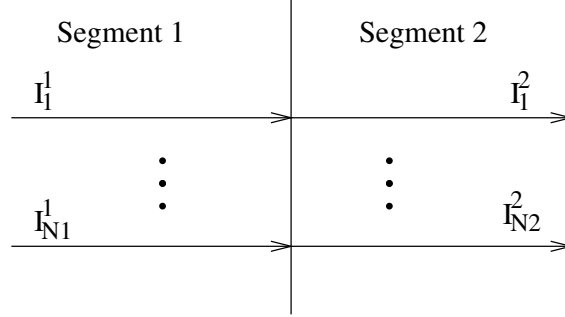
Analogously to the situation for the volume model equations, a generalized formulation of models describing the flux of quantities across the interface between two adjacent segments has been discretized, leaving the user the possibility to provide models determining the coefficients of the underlying equation system. In the following equations  $J_j^1$  denotes the flux of the quantity  $j$  leaving segment 1 whereas  $J_j^2$  gives the flux of quantity  $j$  entering segment 2 (Fig. ??).

The default boundary condition (??) for all quantities is a homogeneous Neumann type condition which defines a zero flux over the boundary.

$$J_j^1 = J_j^2 = 0 \quad (15)$$

These default boundary conditions can be replaced for any combination and number of quantities on both of the adjoining segments by either defining the flux over the interface (Section ??) or defining the actual concentration of the respective quantities by adding Dirichlet type boundary equations as described in Section ??.

**Boundary Flux Equation System** Under the assumption of  $N_1$  and  $N_2$  quantities on the segments 1 and 2, respectively, (??) and (??) represent a linear equation system for the partial current densities of all involved quantities crossing the interface in normal direction. In these equations



**Figure 26:** PROMIS-NT interface model scheme

$$\sum_{j=1}^{N_1} (\beta_{i,j}^{11} \cdot J_j^1) + \sum_{j=1}^{N_2} (\beta_{i,j}^{12} \cdot J_j^2) + \phi_i^1 = 0, \quad i = 1 \dots N_1 \quad (16)$$

$$\sum_{j=1}^{N_1} (\beta_{i,j}^{21} \cdot J_j^1) + \sum_{j=1}^{N_2} (\beta_{i,j}^{22} \cdot J_j^2) + \phi_i^2 = 0, \quad i = 1 \dots N_2 \quad (17)$$

All coefficients  $\beta^{11}$ ,  $\beta^{12}$ ,  $\beta^{21}$ , and  $\beta^{22}$  are functions of the simulation time  $t$ , the process temperature  $T$ , and the spatial coordinates  $x$  and  $y$ . The coefficients  $\phi^1$  and  $\phi^2$  may depend on the concentrations of the quantities on both sides of the interface. The model developer has to provide MDL classes which determine the actual values of the coefficients and their derivatives to the quantity concentrations which are the dependent variables in the PROMIS-NT internal equation system.

**Dirichlet Type Boundary Conditions** Dirichlet type boundary conditions (??) – (??) can be specified for quantities which are not contained in the equation system (??) – (??). The resulting values of the quantity on both sides of the interface can be functions of the simulation time  $t$ , the process temperature  $T$ , the spatial coordinates  $x$  and  $y$  and the values of all available quantities on both sides of the interface.

$$C_{i1}^1 = f(x, y, t, T, C_{j1}^1, C_{j2}^2), \quad i1, j1 = 1 \dots N_1 \quad (18)$$

$$C_{i2}^2 = f(x, y, t, T, C_{j1}^1, C_{j2}^2), \quad i2, j2 = 1 \dots N_2 \quad (19)$$

### 5.2.3 Process Temperature Modeling

PROMIS-NT supports the arbitrary user defined functions for the definition of the lattice temperature  $T$  (??). The process temperature can be a function of the simulation time  $t$ , the simulation start time  $t_s$  and the simulation end time  $t_e$ . The function is evaluated at discrete dates resulting from the built-in time step control algorithms.

$$T = f(t, t_s, t_e) \quad (20)$$

### 5.3 Quantity Management and VISTA Integration

PROMIS-NT provides a default set of quantities available for diffusion simulations which is coordinated with other process simulators within the VISTA TCAD environment. It is supported to define any number of additional quantities including their charge states in different materials and their representation within PIF files. Table ?? depicts an excerpt of the quantities contained in the PROMIS-NT default quantity set. The complete and up-to-date list can be examined by looking into the PROMIS-NT default setup file `promis-defaults.mdl` which is part of the PROMIS-NT distribution.

quantity	MDL name	description
Boron	B	substitutional boron
Boron	B_active	interstitial boron
Boron	B_gb	grain boundary boron (poly silicon)
Arsenic	As	substitutional arsenic
Arsenic	As_active	interstitial arsenic
Phosphorus	P	substitutional phosphorus
Phosphorus	P_active	interstitial phosphorus
Silicon	Si	silicon
Silicon	Si_interstitial	interstitial silicon
Silicon	Si_clustered	clustered silicon
Silicon	Si_vacancy	silicon atom vacancy
Grain Size	Grain_Size	poly silicon grain size
Grain Orientation	Grain_Orientation	poly silicon grain orientation
Stress	Sxx	stress tensor element xx
Stress	Sxy	stress tensor element xy
Stress	Syy	stress tensor element yy
Velocity	vx	velocity lateral component
Velocity	vy	velocity vertical component

**Table 1:** PROMIS-NT quantities

It is recommended to leave these default definitions unchanged if input or output files of PROMIS-NT should be exchanged with other simulators of the VISTA framework.

#### 5.3.1 Charge States

For the computation of the built-in potential (??) it is required that the charge states of quantities in the surrounding material segments are defined. Again PROMIS-NT provides a set of predefined materials (Table ??) for which the charge states of the default quantities Table ?? are properly defined.

The explicit computation of the built-in potential in other materials can be ensured by adding charge state definitions for the required quantities.

### 5.4 The Structure of PROMIS-NT

The diffusion modeling framework PROMIS-NT has been built on top of the basic input/output, equation solving and transient time step control modules of the diffusion simulator DIFFUS. Therefore the

material	MDL name
silicon	Si
germanium	Ge
silicon-germanium	SiGe
gallium arsenide	GaAs
silicon dioxide	SiO2
Nitride	Si3N4
resist	Resist
aluminum oxide	Al2O3
poly crystalline silicon	Poly

**Table 2:** PROMIS-NT segment materials

discretization function of a number of built-in diffusion models were replaced by the discretization functions for the differential equation systems (??) – (??). Interface structures to the Algorithm Library have been introduced within these discretization functions as well as for the initialization and post-processing of quantities, the process temperatures modeling, quantity definition and other strategic algorithms for the diffusion modeling. A set of PROMIS-NT specific MDL **Parameter** classes, operators, functions, and **Model** classes forms the development platform for user defined diffusion models collected in external libraries of MDL extension libraries and MDL input decks (Fig. ??).

## 5.5 The PROMIS-NT Input Deck

The simulation flow of PROMIS-NT is completely controlled by statements given on the input deck. These are responsible for the general simulator setup including information about input and output files, the simulation time, and process temperature progression as well as for the definition of the entire set of diffusion modeling equations (??) – (??).

PROMIS-NT is designed to allow for the development of new diffusion models by solely understanding the structure and the sequence of **Model** evaluations defined on this input deck. Therefore the following sections provide a detailed description of this interface both for the purpose of documentation and for demonstrating the extent of application specific knowledge still required for extending a simulator to such a degree.

It should be noted in advance that within the input deck all indices contained in (??) – (??) are replaced by the according MDL names of the regarding materials and quantities placed inside the MDL array subscription operator ' [ ] '. Furthermore the contents of the following tables reflect the documentation and default values associated with the regarding **Parameters** and **Models**, which will be forwarded to the user by the Algorithm Library in case that missing or erroneous definitions are found on the input deck.

### 5.5.1 Logical Structure of the PROMIS-NT Input Deck

As depicted in Fig. ?? a PROMIS-NT input deck forms a hierarchically organized tree of MDL model instances. Starting from the root by iterating to the outer leaves these model instances are responsible for the more and more detailed description of the diffusion simulation to be computed.

The optional **Model** instances *StartUp* and *ShutDown* build the flanks of the entire input deck structure both in terms of the logical structure and in terms of the simulation flow as they give the user the possibility to define MDL classes which will be evaluated upon startup and shut down of PROMIS-NT.

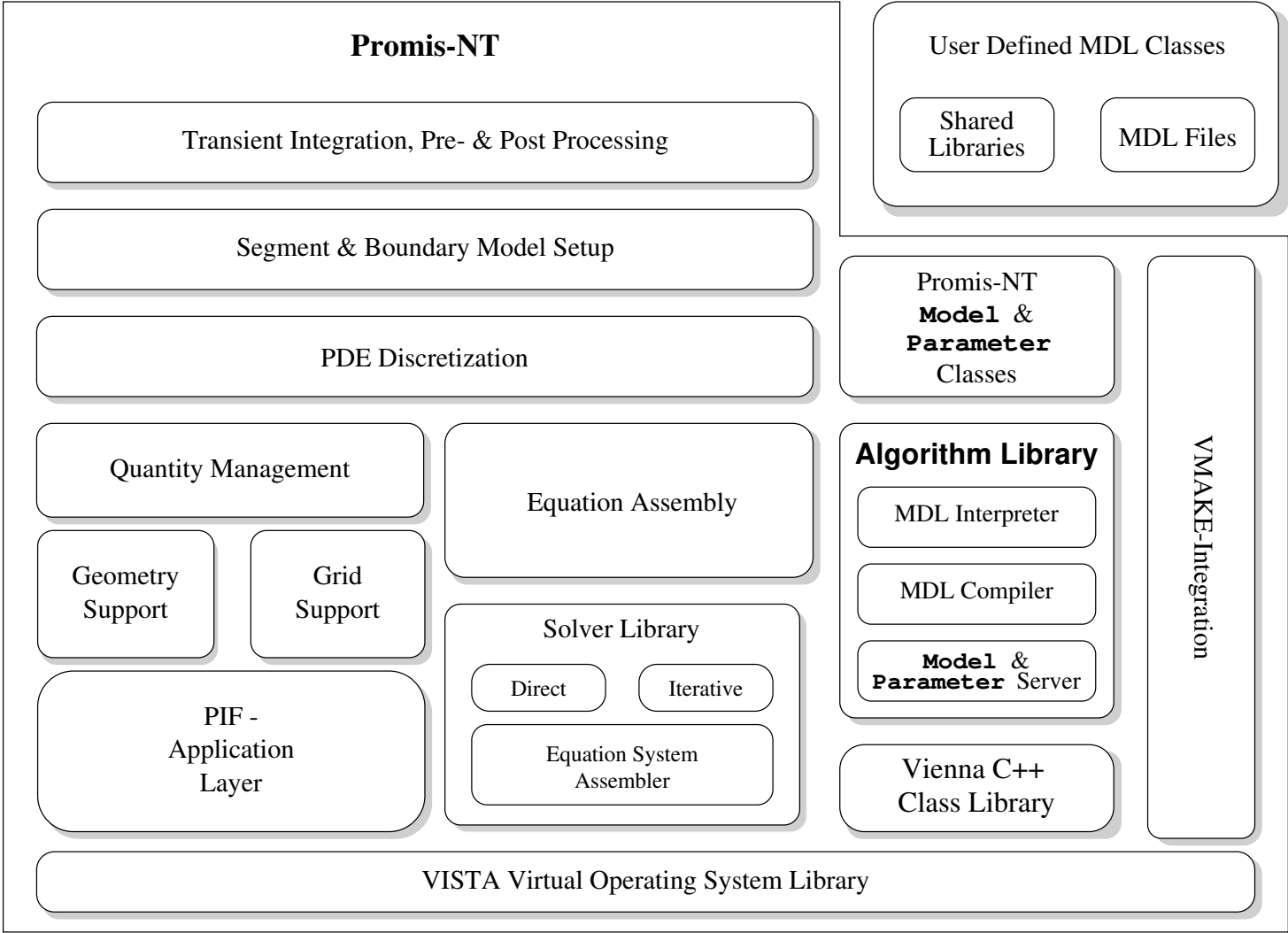


Figure 27: Structure of PROMIS-NT

The root of this tree is given by the definition of the `PromisNTSetup` Model Instance responsible for the general setup of the simulation models and parameters concerning various details of the simulation control.

**Instance** `PromisNTSetup = Instance_Name;`



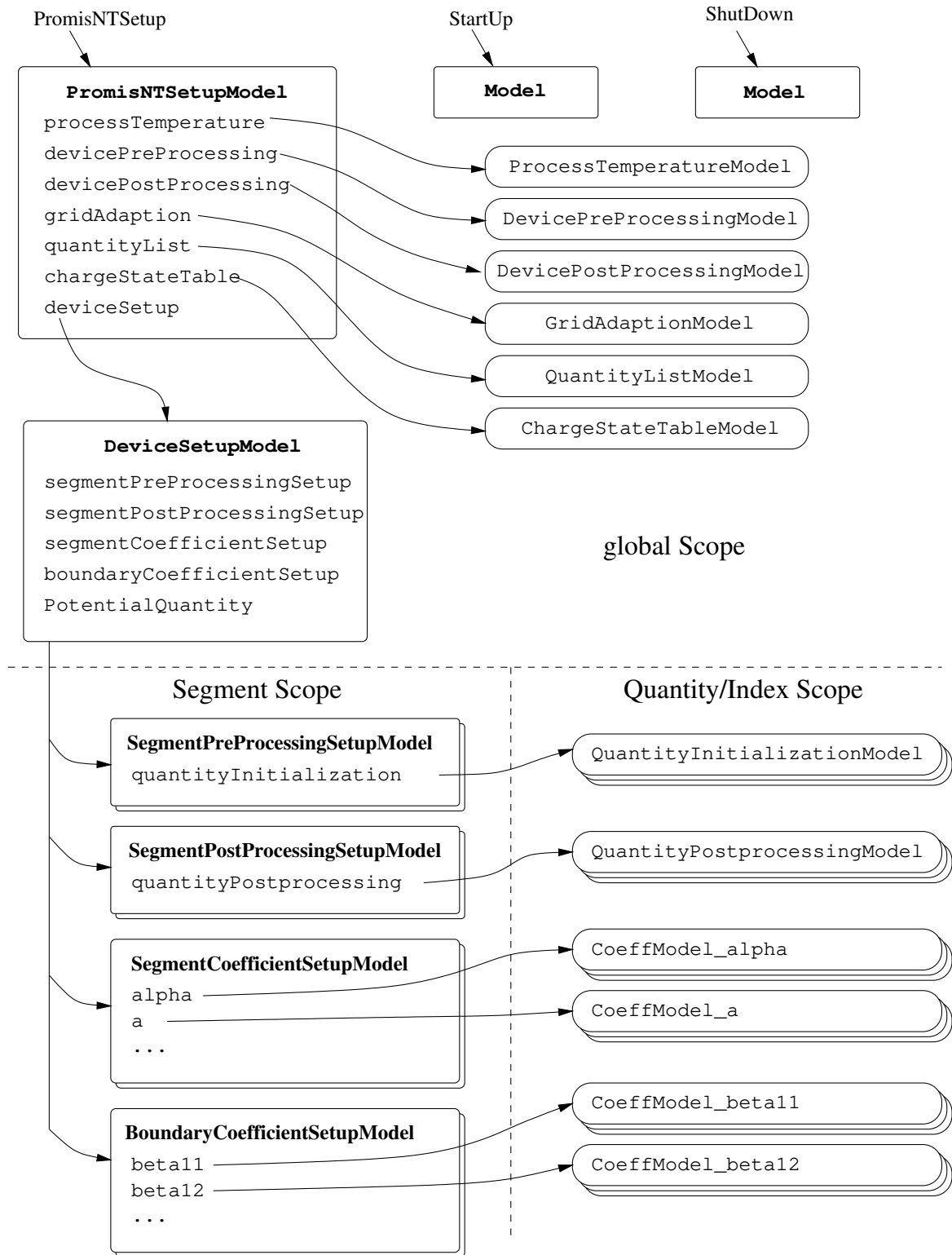


Figure 28: Structure of the PROMIS-NT input deck

The second task for the PromisNTSetup Model Instance is to specify MDL classes which are responsible for

- the process temperature modeling
- optional algorithms which have to be evaluated after all quantities have been initialized to their starting conditions
- optional algorithms which have to be evaluated after the post processing of all quantities
- the optional grid adaption algorithm
- the setup of the quantity list
- the setup of the charge states of the quantities
- the setup of MDL models which in turn are responsible to determine all information associated with the whole device.

The above mentioned `DeviceSetupModel` instance associates with each material segment MDL **Model** classes which are used to setup all segment specific information:

- The `segmentPreProcessingSetupModel` instances are responsible for the specification of **Model** classes which in term are used to initialize all quantity distributions contained in their dedicated segments.
- Likewise the `segmentPostProcessingSetupModel` instances determine the **Model** classes responsible for the quantity distribution post processing.
- The `segmentCoefficientSetupModels` administer associative lists and arrays determining the **Model** classes which will be employed to compute the coefficient values of the volume models (??) – (??).
- Analogously the `boundaryCoefficientSetupModels` contain information about the **Model** classes determining the values of the various boundary model coefficients in (??) – (??).
- The optional string value `PotentialQuantity` can be used to specify the MDL name of the quantity which computes the value of the built-in potential  $\psi$  used in (??), thus replacing (??).

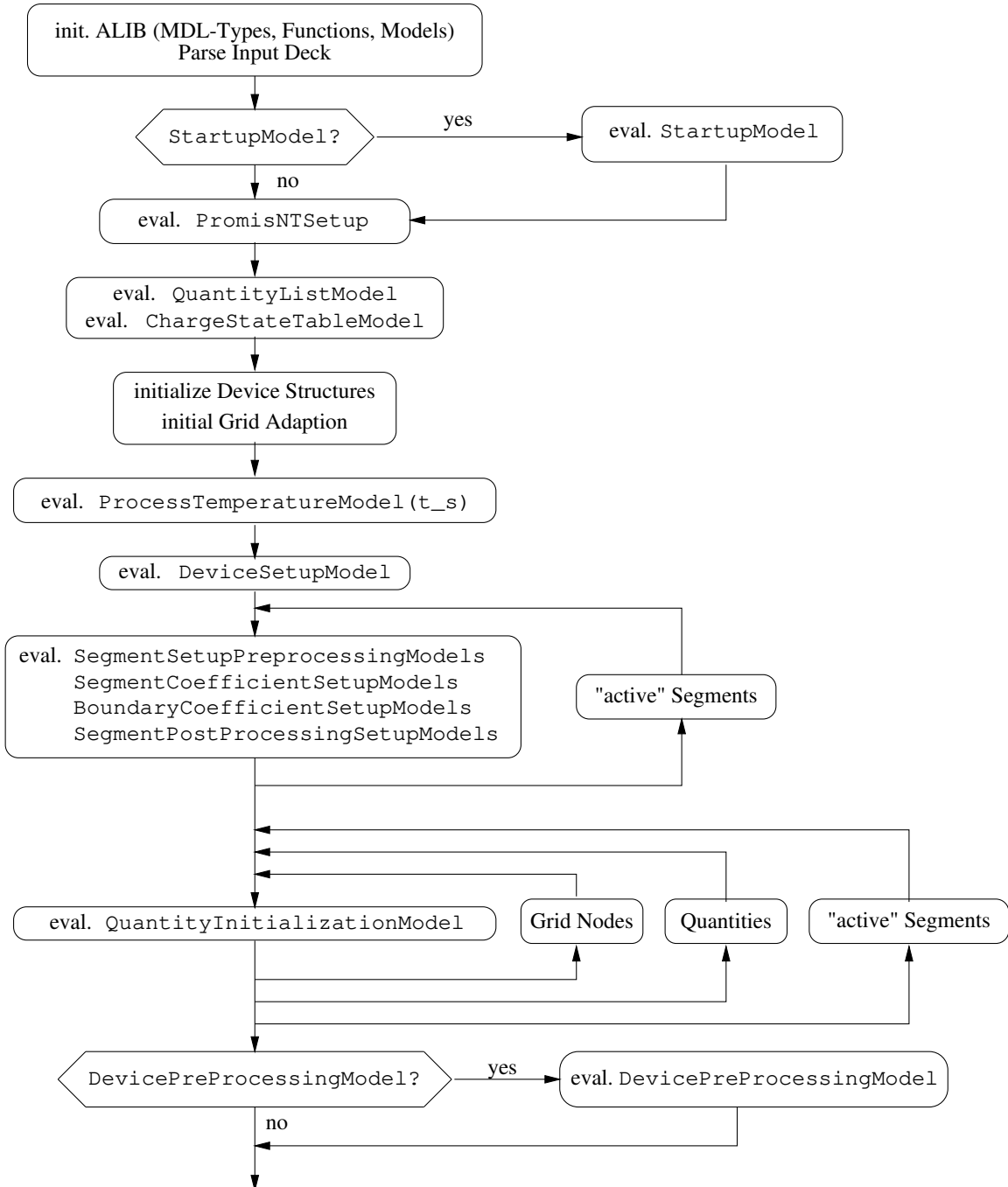
The next hierarchy level of the input deck is formed by **Model** instances dedicated to single segments or boundaries between them. All **Model** instances on this level contain associative lists or arrays containing information about which **Model** instances to use for the computation of the various quantity concentrations and coefficient values on that segment. For each segment which is subject to the diffusion simulation modeling an own set of segment level **Models** has to be defined.

On the last hierarchy level of **Models** contained in PROMIS-NT input decks the quantity specific initialization, coefficient and post processing **Models** can be found. Opposed to the task of all other models discussed so far, these **Model** instances are not used for the further distribution of information about the model structure but for the actual computation of coefficients or quantity values found in (??) – (??).

### 5.5.2 PROMIS-NT Process Flow

Besides the knowledge of the input deck structure, the hierarchical information flow, the model developer has also to be aware of the program flow and sequence of **Model** evaluations to allow for the successful development of diffusion models. The PROMIS-NT process flow can be divided into three major phases:

1. The initialization phase (Fig. ??) starts by parsing the complete input deck. In case the optional **StartUp Model** instance is defined the concerning **Model** class is immediately instantiated and evaluated. No type requirements or input parameters are defined or demanded for these **Model** instances. The typical application of a **StartUp Model** is to prepare input PIF files or to eventually compile external **Model** libraries.



**Figure 29:** Program flow of the PROMIS-NT initialization phase from the users point of view

The next step is the instantiation and evaluation of the `PromisNTSetup Model` instance, followed by the definition of all quantities resulting from the evaluation of the `QuantityListModel` and `ChargeStateTableModel` instances. After evaluating the `ProcessTempera-`

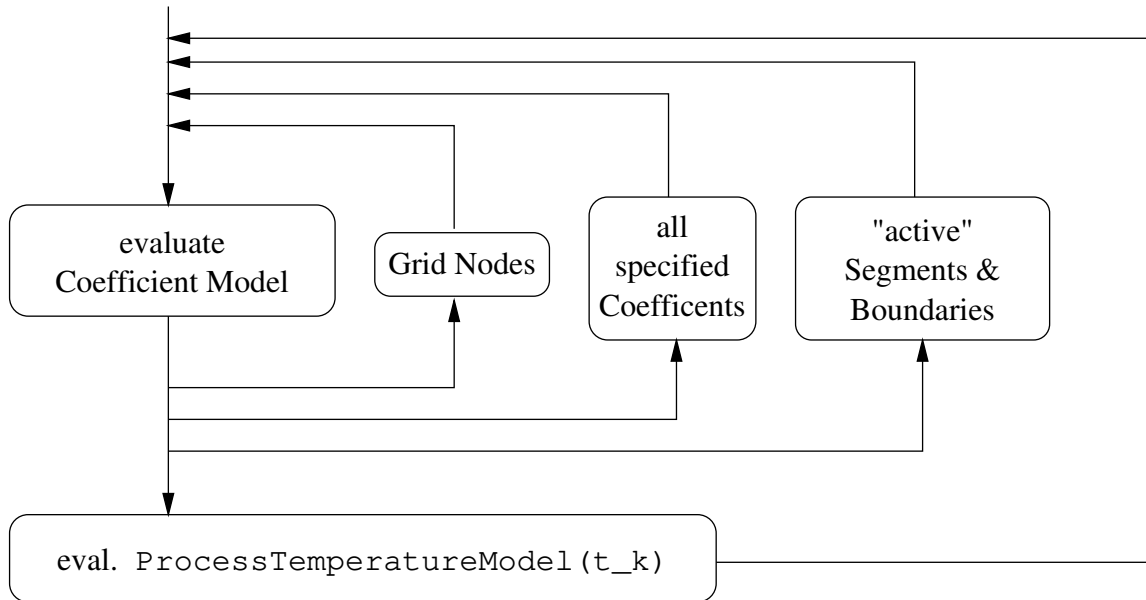
tureModel to obtain the process temperature value for the start time of the simulation and the evaluation of the DeviceSetupModel, the PROMIS-NT process flow continues with the setup of the segment specific **Model** instances.

All SegmentPreProcessingSetupModel, SegmentCoefficientSetupModel, BoundaryCoefficientSetupModel, and SegmentPostProcessingSetupModel instances will be evaluated by iteration over all “active” segments. In this context “active” means that within the PromisNTSetup model instance the associative list diffusionModel assigns the Promis model structure to the according segmen.

The subsequent program loops determine the initial conditions of the differential equation system (??) – (??). All QuantityInitializationModel instances are evaluated by iterating in the innermost loop over the points of the simulation grid, then over all quantities contained on a single segment. In the outermost loop is performed by iterating over all “active” segments.

The initialization phase is concluded by the evaluation of the optional DevicePreProcessingModel instance defined by using the PromisNTSetupModel instance. The purpose of this model is to offer the possibility for interaction with the user in between the quantity initialization and the actual transient simulation such as providing some information about the initial quantity distributions resulting from the setup procedure so far described.

2. The transient simulation (Fig. ??) is performed by iterating over all coefficient models defined during the previous initialization phase to compute the coefficient values in (??) – (??). Again looping is done by first evaluating any model for all points of the simulation grid, repeating this innermost loop for all coefficient models and with the lowest frequency for all “active” segments and boundaries. These loops in turn are repeated for all time steps  $t_k$ . Due to the time step control mechanisms of the simulator no ordering of subsequent time steps is possible.

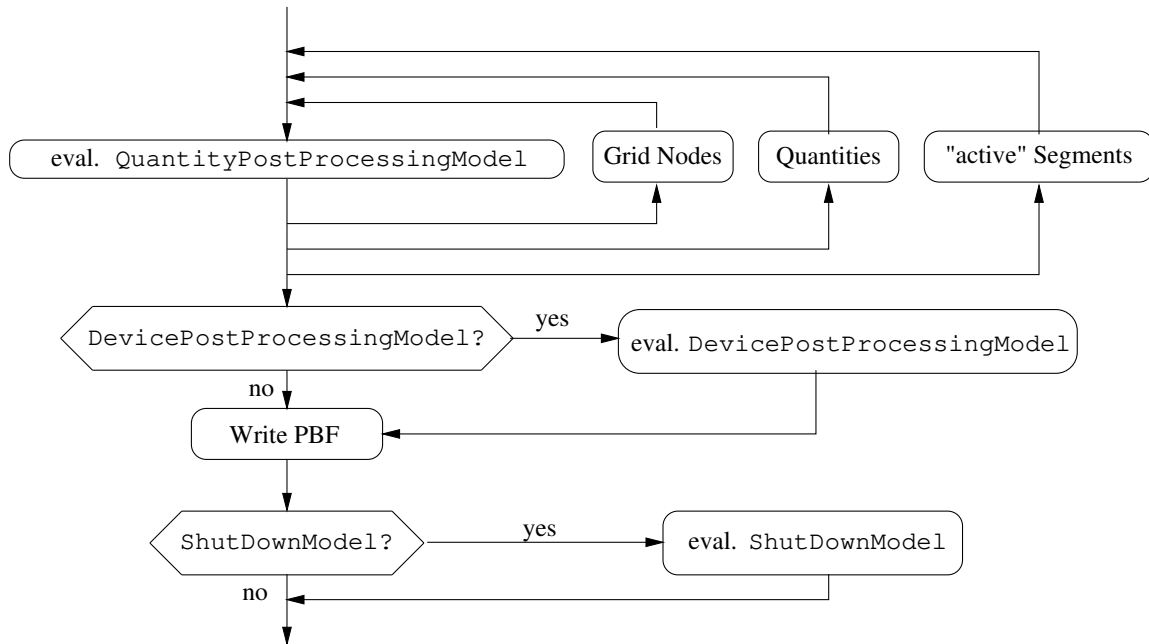


**Figure 30:** Program flow of PROMIS-NT during the transient simulation phase from the users point of view

3. The post processing phase (Fig. ??) shows some similarities to the initialization phase and is used to determine the final values of all quantities which will be written to the resulting PIF file. Therefore a triple loop over all QuantityPostProcessingModel instances is performed analogously to the QuantityPreProcessingModel evaluation loop during the initialization phase. Again

an optional `DevicePostProcessingModel` instance can be used for interacting with the user before the resulting PIF file is generated.

The last possibility to intervene the program flow of PROMIS-NT is to define the optional `ShutDownModel` instance. Similar to the `StartUpModel` instance it can for example be used for user defined conversions of the resulting PIF file.



**Figure 31:** Program flow of the PROMIS-NT post processing phase from the users point of view

## 5.6 PROMIS-NT Application Examples

The following sections present examples for the implementation of diffusion modeling approaches by using the MDL input deck of PROMIS-NT. Their purpose is first to provide an introduction and templates for the usage of the PROMIS-NT diffusion modeling environment. Second these examples shall demonstrate the usability of the Algorithm Library concepts and the MDL extension language for the creation of highly customizable and extensible simulators on the basis of an existing traditional simulator application.

### 5.6.1 Five Species Phosphorus Pair-Diffusion Model

Richardson, Carey, and Mulvaney formulated the five species phosphorus diffusion model[?, ?] described by the differential equation system in (??) – (??). In these equations  $P$ ,  $I$ ,  $V$ ,  $E$ , and  $F$  represent the concentrations of substitutional phosphorus, the silicon interstitials, vacancies, phosphorus-vacancy, and phosphorus-interstitial pairs, respectively.

$$\frac{\partial V}{\partial t} = D_V \nabla^2 V - k_{for}^E P V + k_{rev}^E E - k_{bi} (V I - V^{eq} I^{eq}) \quad (21)$$

$$\frac{\partial I}{\partial t} = D_I \nabla^2 I - k_{for}^F P I + k_{rev}^F F - k_{bi} (V I - V^{eq} I^{eq}) \quad (22)$$

$$\frac{\partial E}{\partial t} = D_E \nabla^2 E + k_{for}^E P V - k_{rev}^E E \quad (23)$$

$$\frac{\partial F}{\partial t} = D_F \nabla^2 F + k_{for}^F P I - k_{rev}^F F \quad (24)$$

$$\frac{\partial P}{\partial t} = -k_{for}^E P V + k_{rev}^E E - k_{for}^F P I + k_{rev}^F F \quad (25)$$

The according MDL input deck can be structured into

- the definition of global **Parameters** containing the values of the coefficients  $k_{for}^E, k_{for}^F, \dots$  (Example ??),
- the global simulation setup for the two affected material segments (Example ??). To allow for series of subsequent simulation experiments the names of input and output device descriptions and the simulation times are forwarded to the simulator by using environment variables.
- the definition of the required quantities  $P, V, I, E$ , and  $F$  (Example ??). Since the electrostatic potential  $\psi$  is not taken into consideration within (??) – (??) the definition of charge states can be omitted.
- The diffusion modeling setup depicted in Example ??
- the setup of initialization and post processing models for all affected quantities (Example ?? and Example ??), responsible for the initialization of the quantities  $I$  and  $V$  to their initial values of  $1.0e14 cm^{-3}$ .
- and finally the definition of the required coefficients of the pair diffusion model (Example ?? and Example ??).

```
// Parameters
Parameter<double> k_for_e = 1.0e-14; // [cm^3/s]
Parameter<double> k_for_f = 1.0e-14;
Parameter<double> k_bi    = 1.0e-10;

Parameter<double> k_rev_e = 10.;      // [1/s]
Parameter<double> k_rev_f = 12.;

Parameter<double> Dv = - 1.0e-10;    // [cm^2/s]
Parameter<double> Di = - 1.0e-09;
Parameter<double> De = - 1.0e-13;
Parameter<double> Df = - 2.0e-13;

Parameter<double> Veq = 1.0e14;      // [1/cm^3]
Parameter<double> Ieq = 1.0e14;
Parameter<double> VIeq = 1.0e14 * 1.0e14;
```

Example 5.1: Global parameter declarations

```

Instance PromisNTSetup = PairDiffInitModel;
NewModel PairDiffInitModel : PromisNTSetupModel {
  evaluate {
    :inputPBF = "$COMPILE";
    :outputPBF = "$OUTPUT";

    :endTime = "$ENDTIME";
    :stepTime = "$STEPTIME";

    :quantityList = "P_Pair_Quantities";
    :chargeStateTable = "P_Pair_Charge_States";
    :processTemperature = "PairDiffTemperatureModel";
    :deviceSetup = "P_Pair_Diff_DeviceSetup";

    :diffusionModel["Si"] = "Promis";
    :diffusionModel["Ambient"] = "Promis";
    :boundaryModel["Ambient"]["Si"] = "Promis";
  }
}

NewModel PairDiffTemperatureModel : ProcessTemperatureModel {
  evaluate {
    :temp = 1173.0;
  }
}

```

Example 5.2: Simulator setup for the five species pair diffusion model

```

NewModel P_Pair_Quantities : QuantityListModel {
  evaluate {
    :quantity["P"] = {{ P }};
    :quantity["V"] = {{ V }};
    :quantity["I"] = {{ I }};
    :quantity["E"] = {{ E }};
    :quantity["F"] = {{ F }};
  }
}

NewModel P_Pair_Charge_States : ChargeStateTableModel {
  evaluate {}
}

```

Example 5.3: Quantity setup for the five species pair diffusion model

```

NewModel P_Pair_Diff_DeviceSetup : DeviceSetupModel {
  evaluate {
    :segmentPreProcessingSetup ["Si"] = "SiSegmentPreProcessingSetup";
    :segmentCoefficientSetup ["Si"] = "SiCoefficient";
    :segmentPostProcessingSetup ["Si"] = "SiSegmentPostProcessingSetup";
    :segmentPreProcessingSetup ["Ambient"] = "AmbientPreprocessingSetup";
    :segmentCoefficientSetup ["Ambient"] = "AmbientCoefficient";
    :segmentPostProcessingSetup ["Ambient"] = "AmbientPostprocessingSetup";

    :boundaryCoefficientSetup ["Ambient"] ["Si"] = "P_Pair_BoundarySetup";
  }
}

```

Example 5.4: Device setup for the five species pair diffusion model

```

NewModel SiSegmentPreProcessingSetup : SegmentPreProcessingSetupModel {
    basicSetup {
        :quantityInitialization["P"] = "Zero_Init";
        :quantityInitialization["V"] = "Zero_Init";
        :quantityInitialization["I"] = "Zero_Init";
        :quantityInitialization["E"] = "Zero_Init";
        :quantityInitialization["F"] = "Zero_Init";
    }
    evaluate {
        call basicSetup;
        :quantityInitialization["V"] = "VI_Init";
        :quantityInitialization["I"] = "VI_Init";
    }
}

NewModel AmbientPreprocessingSetup : SegmentPreProcessingSetupModel {
    evaluate { call basicSetup; }
}

NewModel SiSegmentPostProcessingSetup : SegmentPostProcessingSetupModel {
    setup {
        :quantityPostprocessing["P"] = "No_Post";
        :quantityPostprocessing["V"] = "No_Post";
        :quantityPostprocessing["I"] = "No_Post";
        :quantityPostprocessing["E"] = "No_Post";
        :quantityPostprocessing["F"] = "No_Post";
    }
    evaluate { call setup; }
}

NewModel AmbientPostprocessingSetup : SiSegmentPostProcessingSetup {
    evaluate { call setup; }
}

```

Example 5.5: Pre- and postprocessing setup

The simulation results depicted in Fig. ??, Fig. ??, and Fig. ?? show the expected formation of the flat point defect profiles in the inner device regions determined by their reaction term ( $V I - V^{eq} I^{eq}$ ) in the according equations (??) and (??). The formation of the sharp gradients of the defect concentrations near the surface induced by the Dirichlet boundary conditions and the coupling with the equations for the phosphorus, phosphorus-interstitial, and phosphorus-vacancy pair profiles in term is responsible for the creation of the phosphorus plateau, kink, and tail. The resulting profiles perfectly coincide with the results obtained by the authors of this pair diffusion model.

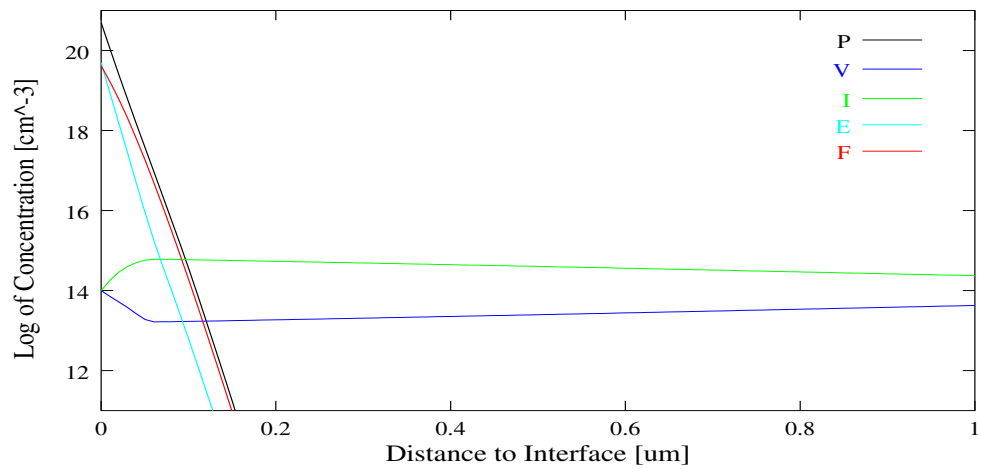
### 5.6.2 Performance Considerations

Benchmarking a number of simulators to obtain reliable performance numbers is due to different implementations of gridding and discretization schemes a difficult task. In most situations it is impossible to force two different simulators to compute their results with exactly the same models on exactly the same simulation grids. The most interesting performance numbers of the Algorithm Library can be obtained somewhat more easily.

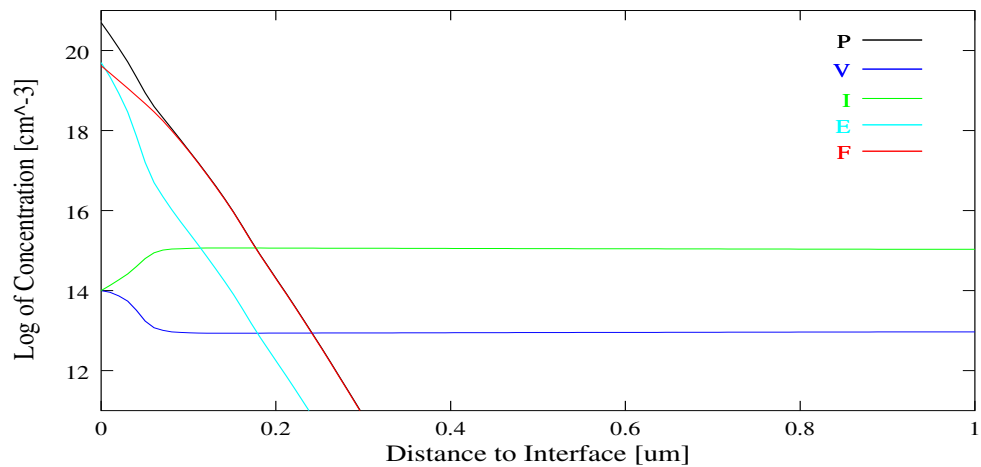
Since the Algorithm Library was put on the PROMIS-NT simulator kernel after the initial release of the first version of the PROMIS-NT simulator, benchmarks with diffusion models which were

- implemented by “hard coded” C++ functions in the original PROMIS-NT application,

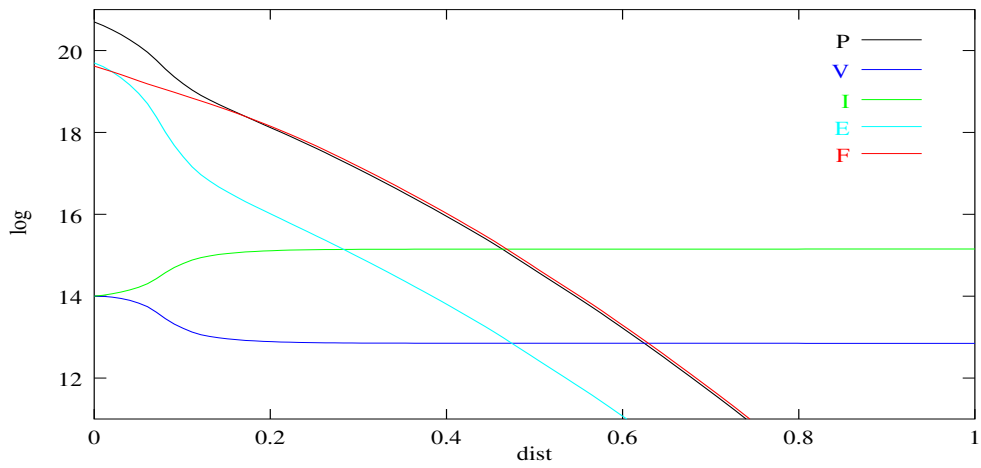




**Figure 32:** Dopant distribution after 6 seconds



**Figure 33:** Dopant distribution after 60 seconds



**Figure 34:** Dopant distribution after 600 seconds

- implemented by using the Algorithm Library interfaces and manually written C++ classes,
- implemented by using MDL definitions on the input deck of the simulator,
- and implemented by utilizing the MDL just in time compiler mechanism

could easily be implemented and delivered comparable results with the following characteristics:

No measurable differences were found between model implementations with “hard coded” C++ functions and “manually written” C++ classes managed by the Algorithm Library.

```

NewModel SiCoefficient : CoefficientSetupModel {
  evaluate {
    :alpha["P"]["P"] = "Uno_alpha"; :alpha["V"]["V"] = "Uno_alpha";
    :alpha["I"]["I"] = "Uno_alpha"; :alpha["E"]["E"] = "Uno_alpha";
    :alpha["F"]["F"] = "Uno_alpha";

    :a["V"]["V"] = "V_a"; :a["I"]["I"] = "I_a";
    :a["E"]["E"] = "E_a"; :a["F"]["F"] = "F_a";

    :gamma["P"] = "P_gamma"; :gamma["V"] = "V_gamma";
    :gamma["I"] = "I_gamma"; :gamma["E"] = "E_gamma";
    :gamma["F"] = "F_gamma";
  }
}

NewModel AmbientCoefficient : CoefficientSetupModel {
  evaluate {
    :alpha["P"]["P"] = "Uno_alpha"; :alpha["V"]["V"] = "Uno_alpha";
    :alpha["I"]["I"] = "Uno_alpha"; :alpha["E"]["E"] = "Uno_alpha";
    :alpha["F"]["F"] = "Uno_alpha";
  }
}

NewModel P_Pair_BoundarySetup : BoundaryCoefficientSetupModel {
  evaluate {
    :dirichlet2["V"] = 1.0e14; :dirichlet2["I"] = 1.0e14;
    :dirichlet2["P"] = 5.0e20; :dirichlet2["E"] = 5.0e19;
    :dirichlet2["F"] = 4.16666e19;
  }
}

```

#### Example 5.6: Coefficient setup

```

NewModel Zero_Init : QuantityInitializationModel {
  evaluate {
    :Cown = 0.0;
  }
}

NewModel VI_Init : QuantityInitializationModel {
  evaluate {
    :Cown = 1.0e14;
  }
}

NewModel No_Post : QuantityPostprocessingModel {}

```

#### Example 5.7: Quantity initialization and postprocessing

```

NewModel Uno_alpha : CoeffModel_alpha {
    evaluate { :Coeff = 1.; }
}

// Coefficients for P related equation
NewModel P_gamma : CoeffModel_gamma {
    evaluate {
        :Coeff      = - ::k_for_e * :P * :V + ::k_rev_e * :E -
                      ::k_for_f * :P * :I + ::k_rev_f * :F;
        :dCoeff\d_P = - ::k_for_e * :V - ::k_for_f * :I;
        :dCoeff\d_V = - ::k_for_e * :P;
        :dCoeff\d_I = - ::k_for_f * :P;
        :dCoeff\d_E =  ::k_rev_e;
        :dCoeff\d_F =  ::k_rev_f;
    }
}

// Coefficients for V related equation
NewModel V_a : CoeffModel_a {
    evaluate { :Coeff = ::Dv; }
}

NewModel V_gamma : CoeffModel_gamma {
    evaluate {
        :Coeff      = - ::k_for_e * :P * :V + ::k_rev_e * :E -
                      ::k_bi * ( :V * :I - ::VIEq );
        :dCoeff\d_P = - ::k_for_e * :V;
        :dCoeff\d_V = - ::k_for_e * :P - ::k_bi * :I;
        :dCoeff\d_I = - ::k_bi * :V;
        :dCoeff\d_E =  ::k_rev_e;
    }
}

// Coefficients for I related equation
NewModel I_a : CoeffModel_a {
    evaluate { :Coeff = ::Di; }
}

NewModel I_gamma : CoeffModel_gamma {
    evaluate {
        :Coeff      = - ::k_for_f * :P * :I + ::k_rev_f * :F -
                      ::k_bi * ( :V * :I - ::VIEq );
        :dCoeff\d_P = - ::k_for_f * :I;
        :dCoeff\d_V = - ::k_bi * :I;
        :dCoeff\d_I = - ::k_for_f * :P - ::k_bi * :V;
        :dCoeff\d_F = + ::k_rev_f;
    }
}

```

Example 5.8: Coefficient **Models** (part 1)

The apparently required indirection for the evaluation of Algorithm Library based models and the initialization of the Algorithm Library structures caused only minimal overhead which was covered by the inaccuracy of the system timers.

The usage of MDL **Model** definitions caused runtime overheads in the range of 50 to 200 percent. Using just in time compiled MDL **Models** normally decreased this overhead to 5 to 10 percent depending on the amount of optimization effort spent on the hand written **Models**.

```

// Coefficients for E related equation
NewModel E_a : CoeffModel_a {
    evaluate { :Coeff = ::De; }
}

NewModel E_gamma : CoeffModel_gamma {
    evaluate {
        :Coeff      = ::k_for_e * :P * :V - ::k_rev_e * :E;
        :dCoeff\d_P = ::k_for_e * :V;
        :dCoeff\d_V = ::k_for_e * :P;
        :dCoeff\d_E = - ::k_rev_e;
    }
}

// Coefficients for F related equation
NewModel F_a : CoeffModel_a {
    evaluate { :Coeff      = ::Df; }
}

NewModel F_gamma : CoeffModel_gamma {
    evaluate {
        :Coeff      = ::k_for_f * :P * :I - ::k_rev_f * :F;
        :dCoeff\d_P = ::k_for_f * :I;
        :dCoeff\d_I = ::k_for_f * :P;
        :dCoeff\d_F = - ::k_for_f;
    }
}

```

#### Example 5.9: Coefficient **Models** (part 2)

Similar results were obtained by benchmarking the MINIMOS-NT device simulator. Also no significant differences between different CPUs and compilers could be measured.

## 6 Comparison of Finite Element and Finite Box Discretization for Three-Dimensional Diffusion Modeling Using AMIGOS

### 6.1 Introduction

The maximum principle is the most important property of solutions to convection-diffusion equations. In its simplest form it states that both the maximum and the minimum concentrations occur on the boundary or at the initial time. This implies that if the boundary and initial values are positive, then the solution is positive everywhere and the concentration never reaches negative values. It is desirable that the employed discretization also satisfies a maximum principle. As is well known, this is guaranteed, if the system matrix resulting from the discretization is an M-matrix [?].

### 6.2 Discretization using AMIGOS

We compare the results of two different spatial discretizations for diffusion in three dimensions using AMIGOS [?] which is especially designed for simple but efficient model development. Through its powerful analytical model interface (AMI) it was possible to implement the Finite Volume (FV) as well as the Finite Element (FE) method. This allows the comparison of the solutions on identical meshes with the same linear solver in a very simple and straightforward manner.

Both for FV and FE we use the well known standard approaches with backwards Euler time discretization.

For FV (see e.g. [?]) we calculate the Voronoi boxes and the corresponding interface areas for each element. In the case of FE we use the Galerkin weighted residual approach with linear form functions.

Then the system matrix  $K$  is of the form

$$K = \frac{M}{\Delta t} + \alpha S$$

where  $M$  denotes the mass matrix,  $S$  is the stiffness matrix, and  $\alpha$  denotes the diffusion constant (homogeneous case). To make  $K$  an M-matrix, the mass matrix has to be lumped, and  $S$  also has to be an M-matrix. Since  $S$  depends on the mesh, this condition translates to a constraint on the mesh.

In two dimensions Delaunay meshes guarantee that the maximum principle is satisfied for FV as well as for FE. In three dimensions Delaunay meshes are still sufficient and necessary for FV, as shown in [?].

However, for FE this does not hold anymore [?]: When applying FE on a Delaunay mesh in three dimensions negative concentrations emerge, which implies that the M-matrix property is lost. Until recently this phenomenon was not fully understood. But by using results from [?] and [?] it is possible to grasp what is going on: The constraints on the mesh for FE and FV are two different purely geometric notions which are equivalent only in two dimensions. Each of them generalizes naturally to higher dimensions, and it can be proved that neither of them implies the other any more. This is an essential discovery with heavy impact on the development of meshing strategies.

### 6.3 Numerical Experiments

To illustrate some of the consequences we solve the pure diffusion equation on one and the same three dimensional Delaunay mesh using FE and FV. We used an ortho-product point distribution on the cubic simulation domain. Every sub-cube was tetrahedralized into six tetrahedra.<sup>1</sup>

In both cases a Gaussian profile (offset  $10^{12}$ ) is used as the initial three-dimensional distribution. As expected FV gives qualitatively correct results. Fig. ?? is a one-dimensional cut, showing the initial distribution and the FE and the FV solution after 120 time-steps. Even for this simple test problem the FE solution strongly violates the maximum principle.

Fig. ?? gives a two-dimensional cut and shows the bad quality of the FE solution. On the black areas the solution becomes negative. Note that the mesh has translational symmetries, which spoil the rotational symmetry of the initial distribution. These areas spread out in time, as shown in Fig. ?. The absolute value of the emerging negative concentrations is much larger than the minimal initial concentration. Finally, Fig. ?? depicts the corresponding relative error between the FE and the FV solution. The error oscillates strongly and is large on the regions, where the concentration is negative. But since mass is conserved the negative concentrations are compensated by additional erroneous mass in the positive areas.

The negative concentrations are a particularly serious problem in diffusion in process simulation, because in typical applications the concentration varies in many orders of magnitude within a small area. For a more complicated transient problem like the pair diffusion model the negative concentrations lead to severe instabilities.

### 6.4 Impact on Meshing Strategies

The obvious cure for these FE-troubles would be to use a mesh which gives an M-Matrix. However, as long as the available meshing tools concentrate on the Delaunay criterion, there is little hope of achieving this.

We want to stress that from a Delaunay point of view these meshes can look really bad, but they are especially tuned to the FE-discretization and give qualitatively correct results.

Otherwise mesh refinement has to be employed. In practice this greatly increases the computational costs and only mitigates the observed effects. It will depend on the application, if one can live with negative concentrations and unphysical flows. As alternative one must decide for FV.

### 6.5 Conclusion

Using AMIGOS, we investigated the constraints which must be imposed on the mesh to avoid the occurrence of negative concentrations in diffusion simulation.

1. In two dimensions a Delaunay triangulation will result in an M-Matrix both for the FE and the FV discretization.

---

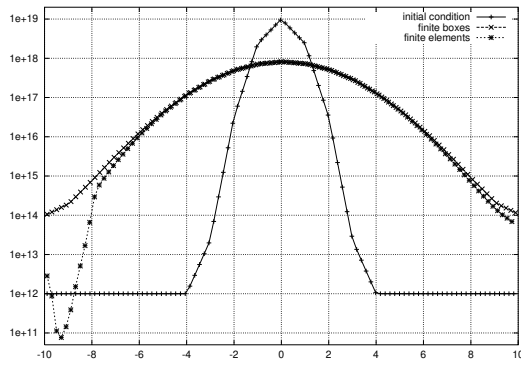
<sup>1</sup>Note that in this simple case the usage of a T5 tessellation for the sub-cubes will result in a Delaunay mesh which fulfills the newly introduced criterion by Xu and Zikatanov.

2. In three dimensions Delaunay is the proper constraint on the mesh for FV. But for FE we get a constraint which may be fulfilled by non-Delaunay grids and not fulfilled by Delaunay triangulations. In short: Delaunay is the wrong criterion (neither necessary nor sufficient) for FE grids in three dimensions.

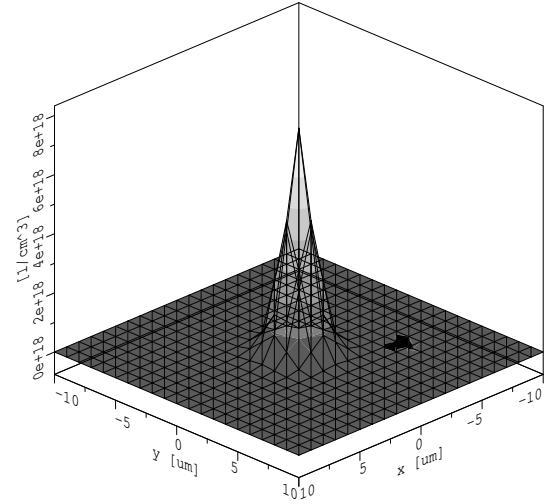
3. Using currently available meshing tools the preferable approach to diffusion modeling is finite volumes.

## 6.6 Acknowledgments

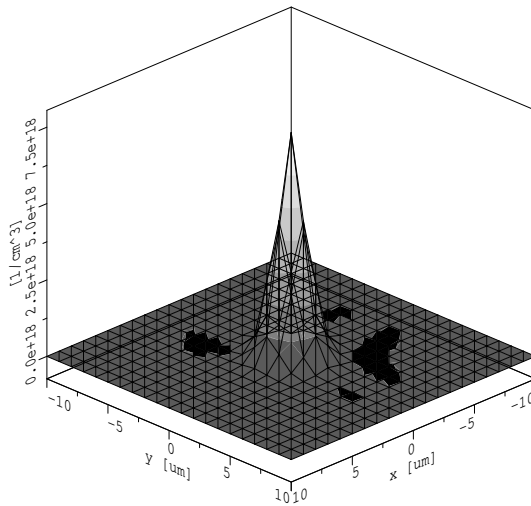
We want to give a special thank to Professor Paula Pietra, Institute of Mathematics of the University of Vienna, for her generous support during the numerical investigations of this work. We also acknowledge support from the “Christian Doppler Forschungsgesellschaft”, Vienna, Austria and Austria Mikrosysteme International AG, Unterpremstätten, Austria.



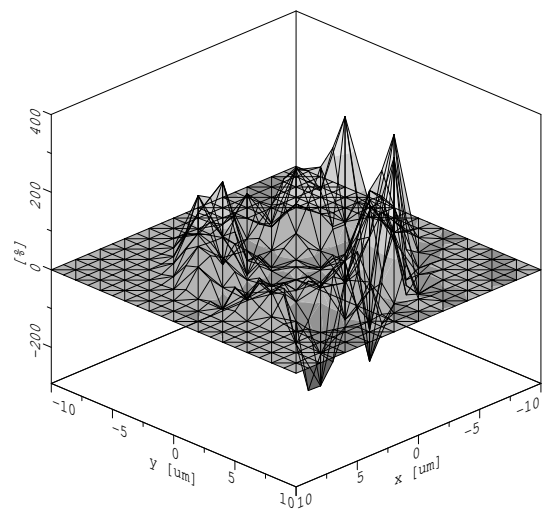
**Figure 35:** Comparison FE versus FV. FE violates the maximum principle.



**Figure 36:** 2D-cut after 5500 seconds. Concentration is negative on black areas.



**Figure 37:** 2D-cut after 8500 seconds. Negative concentrations spread out.



**Figure 38:** Relative error between FE and FV.



## References

- [1] IBM. Advanced Statistical Analysis Program (ASTAP), Program Reference Manual. Technical Report SH20-1118-0, IBM, 1973.
- [2] L.W. Nagel. SPICE2: A Computer Program to Simulate Semiconductor Circuits. Technical Report UCB/ERL M520, University of California, Berkeley, 1975.
- [3] Y. Cheng, M.-C. Jeng, Z. Liu, J. Huang, M. Chan, K. Chen, P.K. Ko, and C. Hu. A Physical and Scalable  $I$ - $V$  Model in BSIM3v3 for Analog/Digital Circuit Simulation. *IEEE Trans. Electron Devices*, 44(2):277–287, 1997.
- [4] Ch.C. Enz. The EKV Model: a MOST Model Dedicated to Low-Current and Low-Voltage Analogue Circuit Design and Simulation. In G.A.S. Machado, editor, *Low-Power HF Microelectronics A Unified Approach*, chapter 7, pp 247–300. IEE London, 1996.
- [5] B. Meinerzhagen and W.L. Engl. The Influence of the Thermal Equilibrium Approximation on the Accuracy of Classical Two-Dimensional Numerical Modeling of Silicon Submicrometer MOS Transistors. *IEEE Trans. Electron Devices*, ED-35(5):689–697, 1988.
- [6] Technology Modeling Associates, Inc., Palo Alto, CA. *TMA MEDICI, Two-Dimensional Device Simulation Program, Version 2.0*, 1994.
- [7] S. Selberherr, A. Schütz, and H.W. Pötzl. MINIMOS—A Two-Dimensional MOS Transistor Analyzer. *IEEE Trans. Electron Devices*, ED-27(8):1540–1550, 1980.
- [8] M.R. Pinto. *PISCES IIB*. Stanford University, 1985.
- [9] J.R.F. McMacken and S.G. Chamberlain. CHORD: A Modular Semiconductor Device Simulation Development Tool Incorporating External Network Models. *IEEE Trans. Computer-Aided Design*, 8(8):826–836, 1989.
- [10] T. Grasser, V. Palankovski, G. Schrom, and S. Selberherr. Hydrodynamic Mixed-Mode Simulation. In K. De Meyer and S. Biesemans, editors, *Simulation of Semiconductor Processes and Devices*, pp 247–250. Springer, Leuven, Belgium, 1998.
- [11] T. Simlinger. *Simulation von Heterostruktur-Feldeffekttransistoren*. Dissertation, Technische Universität Wien, 1996.
- [12] H. Brech, T. Grave, T. Simlinger, and S. Selberherr. Optimization of Pseudomorphic HEMT's Supported by Numerical Simulations. *IEEE Trans. Electron Devices*, 44(11):1822–1828, 1997.
- [13] L.W. Nagel and R.A. Rohrer. Computer Analysis of Nonlinear Circuits, Excluding Radiation (CANCER). *IEEE J. Solid-State Circuits*, SC-6(4):166–182, 1971.
- [14] F.H. Branin, G.R. Hogsett, R.L. Lunde, and L.E. Kugel. ECAP II – A New Electronic Circuit Analysis Program. *IEEE J. Solid-State Circuits*, SC-6(4):146–166, 1971.
- [15] W.J. McCalla and W.G. Howard. BIAS-3 – A Program for the Nonlinear DC Analysis of Bipolar Transistor Circuits. *IEEE J. Solid-State Circuits*, SC-6(1):14–19, 1971.
- [16] W.T. Weeks, A.J. Jimenez, G.W. Mahoney, and D. Mehta. Algorithms for ASTAP – A Network-Analysis Program. *IEEE Trans. Circuit Theory*, CT-20(4):628–634, 1973.
- [17] A. Stach. Simulation von MOSFET-Schaltungen. Diplomarbeit, Technische Universität Wien, 1995.
- [18] U. Tietze and C. Schenk. *Halbleiter-Schaltungstechnik*. Springer, Berlin, 1971.

- [19] C.W. Ho, A.E. Ruehli, and P.A. Brennan. The Modified Nodal Approach to Network Analysis. *IEEE Trans.Circuits and Systems*, CAS-22(6):504–509, 1975.
- [20] W. Van Petegem, B. Geeraerts, W. Sansen, and B. Graindourze. Electrothermal Simulation and Design of Integrated Circuits. *IEEE J.Solid-State Circuits*, 29(2):143–146, 1994.
- [21] S. Selberherr. *Analysis and Simulation of Semiconductor Devices*. Springer, 1984.
- [22] R.E. Bank, D.J. Rose, and W. Fichtner. Numerical Methods for Semiconductor Device Simulation. *IEEE Trans.Electron Devices*, ED-30(9):1031–1041, 1983.
- [23] W. Fichtner, D.J. Rose, and R.E. Bank. Semiconductor Device Simulation. *IEEE Trans.Electron Devices*, ED-30(9):1018–1028, 1983.
- [24] F. Assad, K. Banoo, and M. Lundstrom. The Drift-Diffusion Equation Revisited. *Solid-State Electron.*, 42(3):283–295, 1998.
- [25] K. Blotekjaer. Transport Equations for Electrons in Two-Valley Semiconductors. *IEEE Trans.Electron Devices*, ED-17(1):38–47, 1970.
- [26] P.T. Landsberg and S.A. Hope. Two Formulations of Semiconductor Transport Equations. *Solid-State Electron.*, 20:421–429, 1977.
- [27] G. Baccarani and M.R. Wordeman. An Investigation of Steady-State Velocity Overshoot in Silicon. *Solid-State Electron.*, 28(4):407–416, 1985.
- [28] M. Rudan and F. Odeh. Multi-Dimensional Discretization Scheme for the Hydrodynamic Model of Semiconductor Devices. *COMPEL*, 5(3):149–183, 1986.
- [29] D. Chen, E.C. Kan, U. Ravaioli, C.-W. Shu, and R.W. Dutton. An Improved Energy Transport Model Including Nonparabolicity and Non-Maxwellian Distribution Effects. *IEEE Electron Device Lett.*, 13(1):26–28, 1992.
- [30] C. Lab and Ph. Caussignac. An Energy-Transport Model for Semiconductor Heterostructure Devices: Application to AlGaAs/GaAs MODFETs. *COMPEL*, 18(1):61–76, 1999.
- [31] N.R. Aluru, K.H. Law, P.M. Pinsky, and R.W. Dutton. An Analysis of the Hydrodynamic Semiconductor Device Model – Boundary Conditions and Simulations. *COMPEL*, 14(2/3):157–185, 1995.
- [32] R.E. Bank and D.J. Rose. Global Approximate Newton Methods. *Numer.Math.*, 37:279–295, 1981.
- [33] R.E. Bank and D.J. Rose. Parameter Selection for Newton-like Methods Applicable to Nonlinear Partial Differential Equations. *SIAM J.Numer.Anal.*, 17(6):806–822, 1980.
- [34] C. Fischer. *Bauelementsimulation in einer computergestützten Entwurfsumgebung*. Dissertation, Technische Universität Wien, 1994.
- [35] C.W. Ho, D.A. Zein, A.E. Ruehli, and P.A. Brennan. An Algorithm for DC Solutions in an Experimental General Purpose Interactive Circuit Design Program. *IEEE Trans.Circuits and Systems*, CAS-24(8):416–421, 1971.
- [36] S.H.K. Embabi. *Digital BiCMOS Integrated Circuit Design*. Kluwer, 1993.
- [37] P. Antognetti and G. Massobrio. *Semiconductor Device Modeling with SPICE*. McGraw-Hill, 1988.
- [38] R. Plasun, M. Stockinger, and S. Selberherr. Integrated Optimization Capabilities in the VISTA Technology CAD Framework. *IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1244–1251, 1998.

- [39] M. Stockinger, A. Wild, and S. Selberherr. Closed-Loop MOSFET Doping Profile Optimization for Portable Systems. In *Proc. 2nd Int. Conf. on Modeling and Simulation of Microsystems* [?], pp 411–414.
- [40] W. Pyka and S. Selberherr. Three-Dimensional Simulation of TiN Magnetron Sputter Deposition. In A. Touboul, Y. Danto, J.-P. Klein, and H. Grünbacher, editors, *28th European Solid-State Device Research Conference*, pp 324–327, Bordeaux, France, 1998. Editions Frontieres.
- [41] E. Strasser and S. Selberherr. Algorithms and Models for Cellular Based Topography Simulation. *IEEE Trans.Computer-Aided Design*, 14(9):1104–1114, 1995.
- [42] P. Fleischmann, W. Pyka, and S. Selberherr. Mesh Generation for Application in Technology CAD. *IEICE Trans.Electron.*, E82-C(6):937–947, 1999.
- [43] A. Hössinger and S. Selberherr. Accurate Three-Dimensional Simulation of Damage Caused by Ion Implantation. In *Proc. 2nd Int. Conf. on Modeling and Simulation of Microsystems* [?], pp 363–366.
- [44] A. Hössinger, S. Selberherr, M. Kimura, I. Nomachi, and S. Kusanagi. Three-Dimensional Monte-Carlo Ion Implantation Simulation for Molecular Ions. In *Electrochemical Society Proceedings*, volume 99-2, pp 18–25, 1999.
- [45] W. Bohmayr, A. Burenkov, J. Lorenz, H. Ryssel, and S. Selberherr. Trajectory Split Method for Monte Carlo Simulation of Ion Implantation. *IEEE Trans.Semiconductor Manufacturing*, 8(4):402–407, 1995.
- [46] H. Puchner. *Advanced Process Modeling for VLSI Technology*. Dissertation, Technische Universität Wien, 1996.
- [47] S. Halama. *The Viennese Integrated System for Technology CAD Applications—Architecture and Critical Software Components*. Dissertation, Technische Universität Wien, 1994.
- [48] F. Fasching. *The Viennese Integrated System for Technology CAD Applications—Data Level Design and Implementation*. Dissertation, Technische Universität Wien, 1994.
- [49] A.S. Wong and A.R. Neureuther. The Intertool Profile Interchange Format: A Technology CAD Environment Approach. *IEEE Trans.Computer-Aided Design*, 10(9):1157–1162, 1991.
- [50] G. Mayer. Anwendung des PIF (Profile Interchange Format) in der Prozeß-und Device-Simulation. Diplomarbeit, Technische Universität Wien, 1990.
- [51] Technology Modeling Associates, Inc., Sunnyvale, California. *TMA TSUPREM-4, Two-Dimensional Process Simulation Program, Version 6.5 User's Manual*, 1997.
- [52] Technology Modeling Associates, Inc., Palo Alto, CA. *TMA TSUPREM-4, Two-Dimensional Process Simulation Program, Version 6.2*, 1995.
- [53] W.B. Richardson, G.F. Carey, and B.J. Mulvaney. Modeling Phosphorus Diffusion in Three Dimensions. *IEEE Trans.Computer-Aided Design*, 11(4):487–496, 1992.
- [54] B.J. Mulvaney, W.B. Richardson, and T.L. Crandle. PEPPER - A Process Simulator for VLSI. *IEEE Trans.Computer-Aided Design*, 8(4):336–349, 1989.
- [55] M. Radi, E. Leitner, E. Hollensteiner, and S. Selberherr. AMIGOS: Analytical Model Interface & General Object-Oriented Solver. In *Simulation of Semiconductor Processes and Devices*, pp 331–334, Cambridge, Massachusetts, 1997.

- [56] M. Putti and Ch. Cordes. Finite Element Approximation of the Diffusion Operator on Tetrahedra. *SIAM J.Sci.Comput.*, 19(4):1154–1168, 1998.
- [57] F.W. Letniowski. Three-Dimensional Delaunay Triangulations for Finite Element Approximations to a Second-Order Diffusion Operator. *SIAM J.Sci.Stat.Comput.*, 13(3):765–770, 1992.
- [58] *2nd Int. Conf. on Modeling and Simulation of Microsystems*, San Juan, Puerto Rico, USA, 1999.