VISTA Status Report
June 1999

K. Dragosits, T. Grasser, R. Strasser, S. Selberherr

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

Contents

1 Parallel and Distributed Simulation

Computing power is a vital resource for the extensive application of TCAD simulation. Moreover, the efficient utilization of the computational resources is almost as important as their existence. Excess computing power can often not be tapped due to lacking functionality of the software components involved in a simulation. The shortcomings in this respect exist in two senses. For the first, the employed algorithms are not capable of doing their computations in *parallel*, which means they can not compute on more than one processor at a given time and thus excess processors, e.g. workstations, although available, can not be used. The second reason which inhibits parallel and distributed computation, is the lack of a *middleware* which cares for the problems that arise from parallel computation, namely synchronization, load balancing, or error handling, to name some of them. SIESTA puts strong emphasis on these issues. It's job-farming facilities offer a very efficient front-end for parallel computation on a cluster of heterogeneous workstations.

1.1 Parallel Computation

Since parallel computation offers a way to scale simulation problems, there has always been a strong desire to do so. Inspired by common sense, the naive might think that two computers could solve one problem within half of the time that is required on a single machine, and ten computers could even reduce the amount of time to a tenth of that, and so on. However, it turns out that for many problems these projections do not hold. Depending on the problem and the algorithm in use, this imagination of scalability only holds for low grades of parallelization due to an inherent *communication* overhead.

Although computation itself takes place within less time, an additional amount of time has to be spent on the communication between separated or even dislocated parts of a parallel simulation, as long as the parts are not independent from each other. Faster computing parts will have to wait for slower ones, because the exchange of the simulation state has to take place at certain *simulation times* and computation on the involved heterogenous workstations will take different amounts of *real time*. This means that the slowest part determines the performance of the whole simulation.

One way to improve scalability is to carefully select the level at which parallelization takes place. There might be several levels which allow to split up a simulation problem into several smaller ones which can compute in parallel. This strongly depends on the nature of the problem under consideration and the algorithms that are utilized. Nevertheless, we discuss two extreme cases which should be applicable to a wide range of problems.

1.1.1 Bottom-Level Strategy

The parallelization of a simulation problem at its bottom-level means that it is split up at the lowest possible level (Fig. 1). In the case of the solution of a partial differential equation this could mean that the solver operates in parallel. This kind of parallelization requires a high grade of specialization and strongly depends on the nature of the problem. There exists software which supports the implementation of this kind of parallel algorithm [1, 2, 3]. Nevertheless, the actual implementation differs from case to case and represents a considerable effort.

Bottom-level parallelization produces maximum communication overhead compared to implementations taking place on a higher level and will therefore deliver the worst scalability. However, some situations require parallelization on bottom-level (huge simulation domain, etc.). Especially for simulations reaching

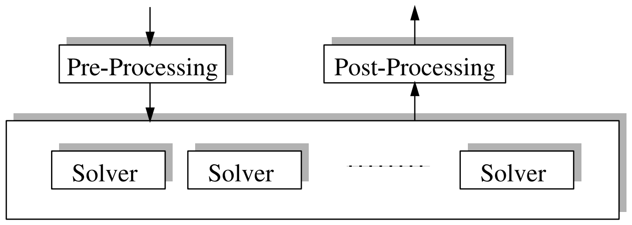


Figure 1: The bottom-level strategy divides a simulation problem at the lowest possible level.

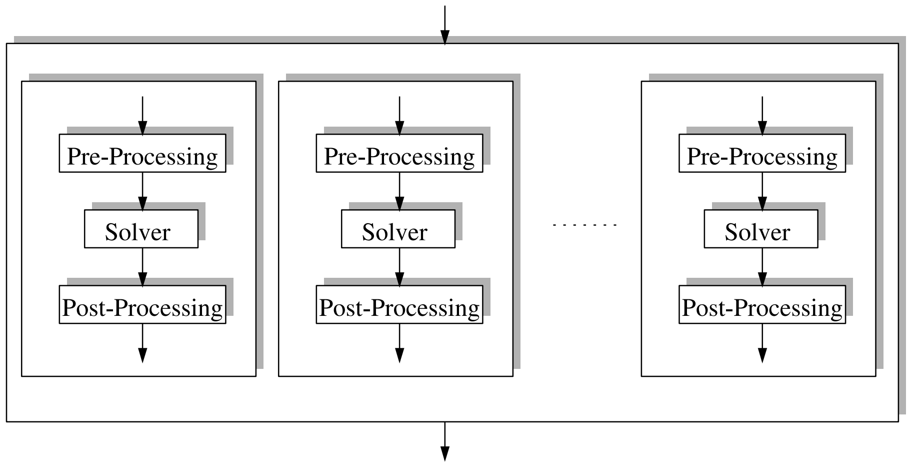


Figure 2: The top-level strategy divides the simulation problem at the highest possible level.

to the limit of existing computers in terms of computation time and memory consumption, a parallelization on bottom-level is the only option available.

1.1.2 Top-Level Strategy

In contrast to its counterpart, a top-level parallelization strategy (Fig. 2) is splitting up a simulation problem at the highest possible level. Let us consider a design of experiments involving several independent executions of a simulator. Instead of executing a bottom-level parallelized simulator sequentially, top-level parallelization executes several (sequential) simulators in parallel.

Since individual executions of a simulator are independent from each other, a communication overhead does not exist at all. Moreover, top-level parallelization does not require any modifications to an existing simulation tool. As long as there exists software which is able to deal with remote execution and the synchronization of the simulations, top-level parallelization can easily be applied to arbitrary simulation tools. It should be stressed that this kind of parallelization is especially efficient when applied to "design of experiments" applications or optimization problems. Since these tasks tend to require large numbers of simulator invocations resulting in huge amounts of computation time, a good scalability is highly desirable.

1.1.3 Summary

The discussion above has shown the implications — summarized in Table 1 — that result from the two extremes of parallelization strategies. It is clear that bottom-level parallelization should only be applied

	Top-level	*Bottom-level*
Communication Overhead	Low	High
Implementation Effort	Low	High
Scalability	Good	Poor
Field of Application	design of experiments; optimization; sensitivity analysis; statistical analysis	extensive single-run simulations; memory extensive problems

Table 1: Properties of parallelizations at *top*-level and *bottom*-level.

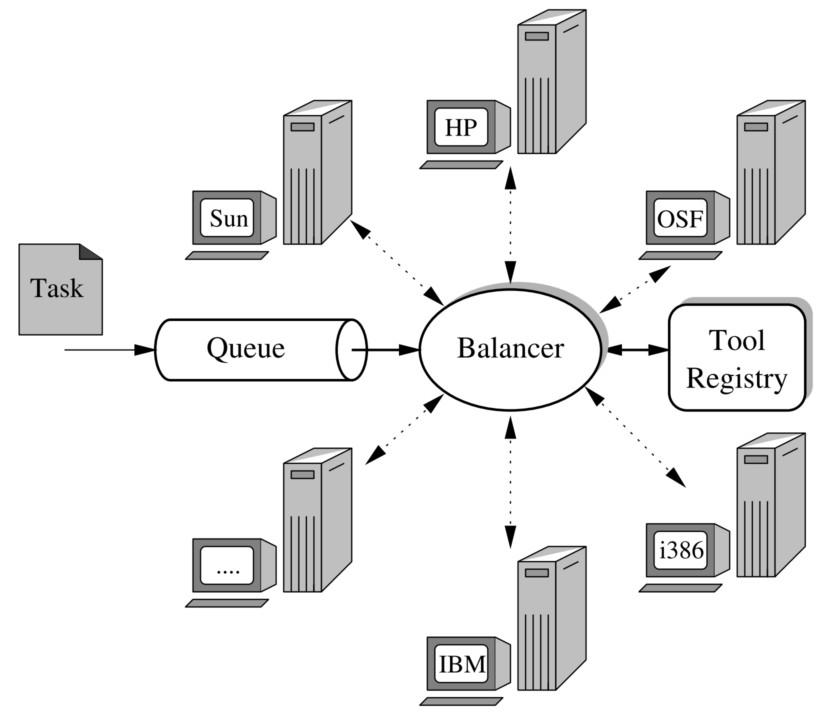


Figure 3: Distributed computing on a cluster of heterogeneous workstations.

where it really makes sense and no other alternative is available. The discussion also points out the potential which parallelization on a high level offers for most of the simulation problems arising from TCAD investigations. Despite of this, few implementations which explore this field exist [4, 5]. All considerations will focus on parallelization at the top-level in the remaining of this chapter.

1.2 Distributed Computing

As long as the parallel computation is not carried out on a single machine with multiple processors, a simulation has to spread out to remote machines using a network connection as depicted in Fig. 3. It will therefore use machines with different properties in terms of speed, memory, and system load. This implies that the computation time which is necessary to finish a given simulation problem will differ from one machine to another. A further implication of distributed computation is an increased susceptibility to single point failures, which will degrade the robustness of the whole simulation system unless measures are taken to account for them.

	$P_{Fail,Total}$
1 Hour Total Simulation Time	16%
1 Day Total Simulation Time	98.8%

Table 2: For a distributed simulation which is susceptible to single point failures the probability to fail is 98.8% if the experiment on a cluster of 20 workstations takes one day.

1.2.1 Workload Distribution

Although there is no need for information exchange between concurrent parts of the parallel simulations, the completion of the whole simulation depends on the very last part to finish. This means that load balancing is necessary in order to reduce the overall simulation time. Load balancing means that faster machines should get a bigger share of the whole workload and, on the other hand, slower machines should obtain less of it. Thus, since the faster machines do more work than the slower ones do, the former will have to wait less until the latter finish. The ultimate case occurs when workload is optimally balanced and all parts finish simultaneously. A strategy which intends to achieve this situation is discussed in Section 1.2.4.

1.2.2 Robustness

Distributed computation takes place in a quite complicated technical system. Aside from the stability of the hardware and the operating system itself, distributed computation adds one more source of system failures which is the network connections. Likewise to the complexity of the system, also the probability of system failures rises. Additionally, the longer the overall simulation takes, the more likely is the occurrence of a failure within that period.

In order to get a feeling for the stability that can be achieved let us briefly sketch a case study of distributed computation on a cluster of workstations. Assuming the probability of a hardware or operating system failure is P_{OS}, the probability of failure due to disk storage shortage is P_{Disk}, and a network failure occurs with a probability of P_{Net}. Thus the probability for the successful completion of a simulation on a single machine is

$$P_{Succ} = (1 - P_{OS}) \cdot (1 - P_{Disk}) \cdot (1 - P_{Net}) \, .$$

Whereas the probability to fail for an ensemble of N simulations which are computing in parallel is

$$P_{Fail,Total} = 1 - \prod^{N} P_{Succ,i} = 1 - (P_{Succ})^N \, ,$$

assuming that all parts of the distributed simulation are exposed to equal risk and an individual failure invalidates the entire simulation. A reasonable assumption for values of P_{OS} and P_{Disk} is one failure per month and one failure per week for network failures P_{Net} which is equivalent to $P_{OS} = P_{Disk} = 1.93 \cdot 10^{-3}$, and $P_{Net} = 5.95 \cdot 10^{-3}$ failures per hour.

Table 2 shows the resulting failure probabilities for two experiments taking one hour's time and one day, respectively, under the assumption that (sub-) processes are computing optimally balanced on a cluster of 20 workstations. Table 2 makes clear that parallel and distributed computation on a local area network results in a fairly unstable system unless special measures are taken in order to improve stability. This is particularly true for large scale simulation experiments such as optimizations, which can take up to a week's time or even longer.

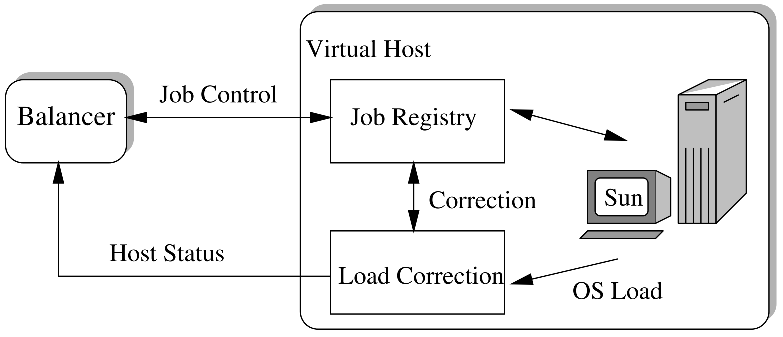


Figure 4: A virtual host corrects the delayed load that is reported by the operating system

1.2.3 Load Polling

The load of a computer is an important measure for an effective balancing mechanism, yet obtaining a correct and up-to-date value of a host's load is not trivial.

First, the load has to be polled periodically within acceptable time intervals. Since usually large numbers of computers are involved (ten and more), this cannot be done sequentially, because not all hosts can be polled within one interval. This is due to the fact that each host can take a couple of seconds until it responds. Moreover, sequential polling is highly susceptible to corrupted computers, since it has to wait for each computer's response until it is able to proceed. Hence, the load of each host needs to be polled in parallel using asynchronous communication.

Secondly, the load which is reported by an operating system, is usually subject to a delay. This delay is not acceptable, since it results in an unstable system which will lead to an oscillation of the computers workload. Therefore, the reported load needs to undergo some processing in order to account for this delay.

Estimation of the Effective System Load. SIESTA uses as much knowledge about the state of a host as possible in order to overcome the dilemma of a delayed value of the operating system load. The diagram depicted in Fig. 4 shows how SIESTA deals with this issue in order to obtain an acceptable value for the *effective load* of a computer which is feasible as a basis for load control leading to a stable system.

SIESTA introduces a virtual host which encapsulates the actual host and instead of communicating with the *real* host it does its business with the hosts virtual substitute. Since this virtual host knows everything about jobs that are running on itself, it is able to correct that portion of the host's load which was produced by jobs that have been submitted by SIESTA. For each of these jobs the virtual host adds a correction term which compensates the delay inherent to the operating system load. Fig. 5 shows this procedure in detail. l_{os} is the load which is reported by the operating system, whereas l_{corr} denotes the correction which is added in order to obtain the effective load l_{eff}.

$$l_{eff} = l_{os} + \underbrace{\underbrace{\sum_{i=1}^{N} e^{-\frac{t-t_i^{start}}{\tau}}}_{\text{running jobs}} - \underbrace{\sum_{i=1}^{M} \left[\left(1 - e^{-\frac{t_i^{stop}-t_i^{start}}{\tau}} \right) * e^{-\frac{t-t_i^{stop}}{\tau}} \right]}_{\text{finished jobs}}}_{l_{corr}}$$

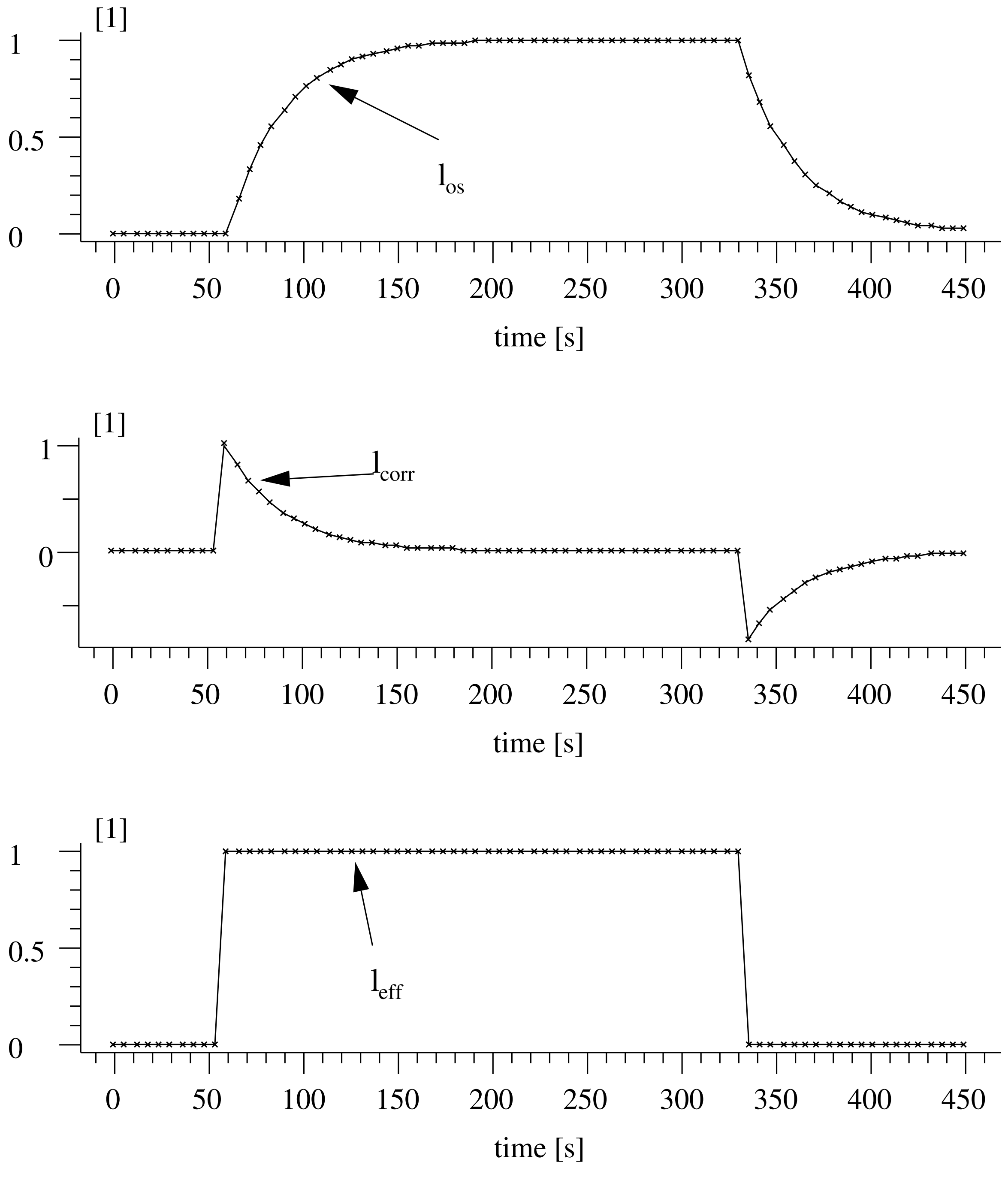


Figure 5: The delayed operating system load l_{os} of a host is corrected by l_{corr} in order to get its effective load l_{eff}.

1.2.4 Host Selection

Assuming that acceptable values for each host's workload are available, a decision has to be made to which one of the registered hosts a job should be submitted. This decision basically depends on three criteria:

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

Tool Management. A flexible management of tools is very important for a smooth operation of the simulation environment in large clusters of heterogeneous workstations. Therefore, a registration of simulation tools, or system programs in general, with the simulation environment is desirable.

Each tool's registration introduces a list of computation hosts where it is potentially available. This restriction might be necessary due to the existence of so called node-locked licenses which require that a tool is executed on a certain machine. Another reason for the restriction to a subset of the available hosts can be different computer architectures or operating systems. It is easily possible that a simulation tool is only available for some operating system. Furthermore, memory considerations might force a user to launch a simulation tool only on hosts where sufficient memory for a proper operation of the tool is available. It should be stressed that a registration is only necessary for tools which either are not available on every host, or for tools which have a limited number of available licenses.

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

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

$$l_i^{max} = \left(l_i^{lim} + l_{base}\right) + 1.0,$$

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

$$p_i = \frac{\max\left(1.0\ ,\ \frac{l_i^{eff}+1.0}{n_i^{cpu}}\right)}{w_i} \tag{1}$$

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

1.3 Performance Estimation

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

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

1.4 User Interaction and Feedback

Remote computation usually involves a couple of workstations and large numbers of tasks that can be in various states. Therefore, it is vital for the user to be able to track the state and actions of the queueing system. Additionally, the user needs the ability to configure the job-farming system and to customize the computation hosts as well as their programs.

1.4.1 The Queue Manager GUI

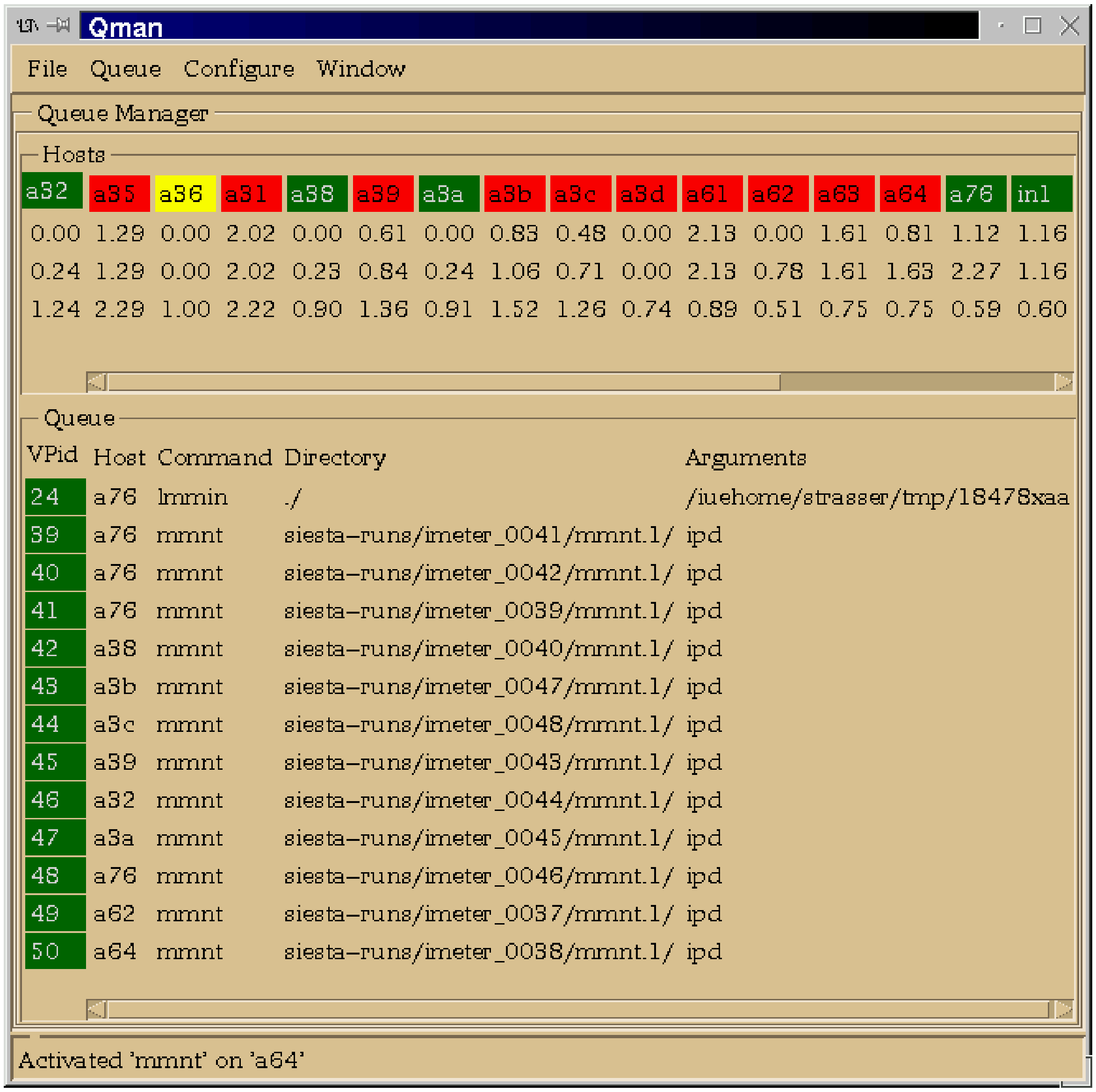
SIESTA offers a GUI (Fig. 6) which gives the user a detailed insight into the Queue Manager's internal state. It should be stressed that each SIESTA application which imports the job-farming module automatically also gets the benefits from this users interface, without any further programming effort.

Computation Hosts. The Hosts part depicts the collection of computation hosts and their state. Colors are utilized to indicate the state of each host. A green colored host is available for computation. Red color means that the hosts load is too high to submit jobs to it. And finally a yellow color denotes hosts which are currently not reachable.

Job Queue. All the jobs that have been submitted to the Queue Manager are depicted in the Queue section. Green color marks executing jobs, whereas red color indicates that a job is waiting until a host becomes available for computation. For each job the command, the command line arguments, and the working directory are displayed. Additionally, the name of the execution host is displayed for executing jobs.

1.4.2 Configuration of the Job Farming System

The Qman Settings window (Fig. 7) allows the user to adjust various settings of the computation hosts, and to configure system tools. Furthermore, several settings used for load balancing can adjusted here. The value of Base Load is added to the load limit of each host (see Section 1.2.4 on page 7) and the total number of executing jobs is limited to the value of Job Limit.

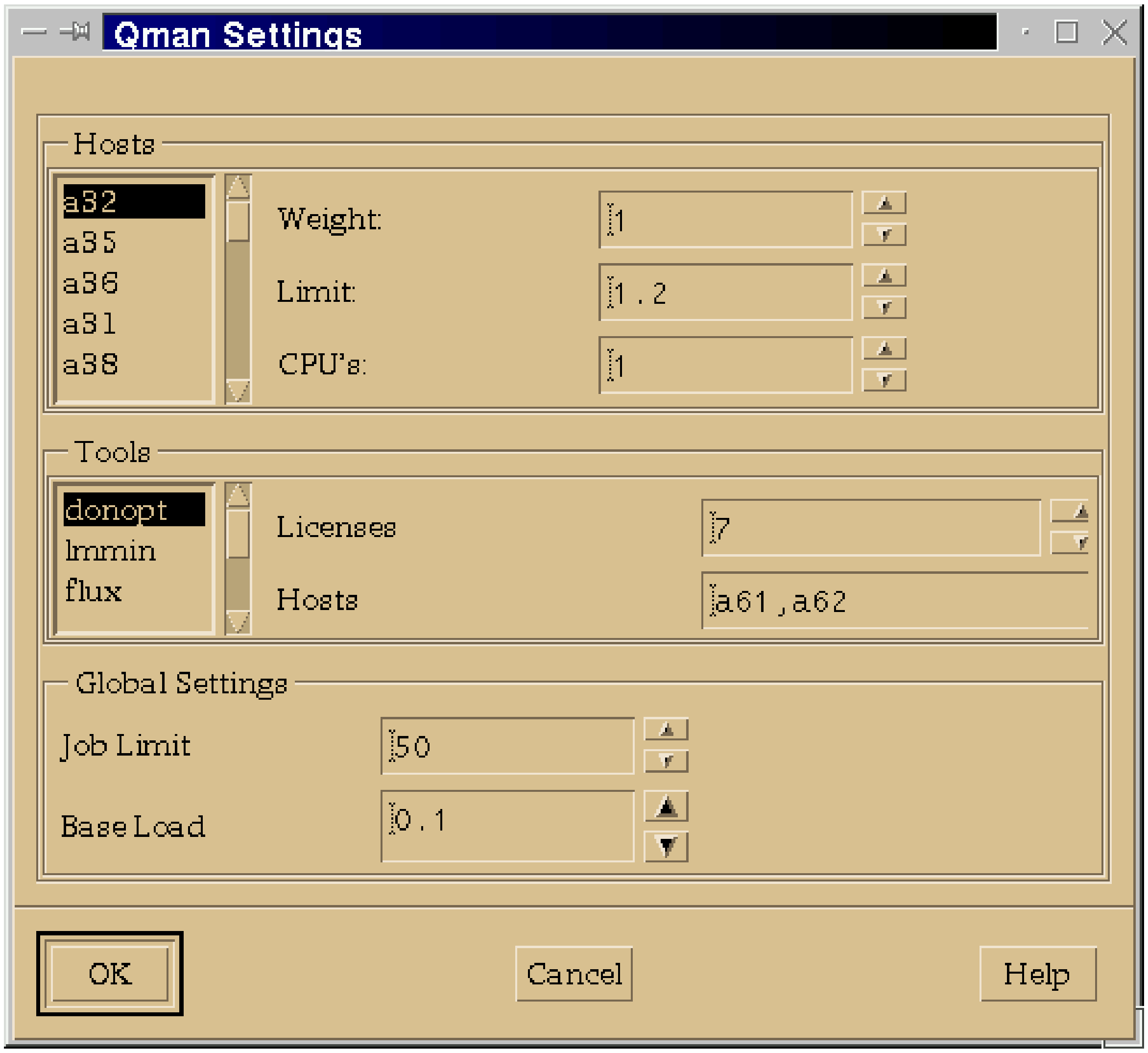


a32	a35	a36	a31	a38	a39	a3a	a3b	a3c	a3d	a61	a62	a63	a64	a76	inl
0.00	1.29	0.00	2.02	0.00	0.61	0.00	0.83	0.48	0.00	2.13	0.00	1.61	0.81	1.12	1.16
0.24	1.29	0.00	2.02	0.23	0.84	0.24	1.06	0.71	0.00	2.13	0.78	1.61	1.63	2.27	1.16
1.24	2.29	1.00	2.22	0.90	1.36	0.91	1.52	1.26	0.74	0.89	0.51	0.75	0.75	0.59	0.60

VPid	Host	Command	Directory	Arguments
24	a76	lmmin	./	/iuehome/strasser/tmp/18478xaa
39	a76	mmnt	siesta–runs/imeter_0041/mmnt.1/	ipd
40	a76	mmnt	siesta–runs/imeter_0042/mmnt.1/	ipd
41	a76	mmnt	siesta–runs/imeter_0039/mmnt.1/	ipd
42	a38	mmnt	siesta–runs/imeter_0040/mmnt.1/	ipd
43	a3b	mmnt	siesta–runs/imeter_0047/mmnt.1/	ipd
44	a3c	mmnt	siesta–runs/imeter_0048/mmnt.1/	ipd
45	a39	mmnt	siesta–runs/imeter_0043/mmnt.1/	ipd
46	a32	mmnt	siesta–runs/imeter_0044/mmnt.1/	ipd
47	a3a	mmnt	siesta–runs/imeter_0045/mmnt.1/	ipd
48	a76	mmnt	siesta–runs/imeter_0046/mmnt.1/	ipd
49	a62	mmnt	siesta–runs/imeter_0037/mmnt.1/	ipd
50	a64	mmnt	siesta–runs/imeter_0038/mmnt.1/	ipd

Activated 'mmnt' on 'a64'

Figure 6: The Queue Manager's GUI comprehensively displays the state of SIESTA's job farming system.


Qman Settings

Hosts

a32
a35
a36
a31
a38

Weight: 1
Limit: 1.2
CPU's: 1

Tools

donopt
lmmin
flux

Licenses: 7
Hosts: a61,a62

Global Settings

Job Limit: 50
Base Load: 0.1

Figure 7: A configuration GUI for SIESTA's job farming system.

2 Mixed-Mode Device Simulation

2.1 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 [6] 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 [7].

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 [8] 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 [9, 10] and the Philips MM9 model [11]. 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 [12], GALENE [13], MEDICI [14], MINIMOS [15], and PISCES [16]. 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 [17, 18]. 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 [19]. 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 the device simulator MINIMOS-NT 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 [20].

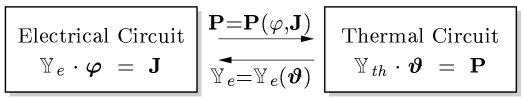

Electrical Circuit
$\mathbb{Y}_e \cdot \boldsymbol{\varphi} = \mathbf{J}$
$\mathbf{P}=\mathbf{P}(\varphi,\mathbf{J})$
$\mathbb{Y}_e=\mathbb{Y}_e(\boldsymbol{\vartheta})$
Thermal Circuit
$\mathbb{Y}_{th} \cdot \boldsymbol{\vartheta} = \mathbf{P}$

Figure 8: 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) [21] or High-Electron-Mobility-Transistors (HEMTs) [22, 23] 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.

2.2 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) [24, 25, 26] or the tableau approach [27]. 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 [28, 29]. To properly account for voltage-defined branches the modified nodal approach (MNA) has been proposed which allows for the introduction of arbitrary branch currents [30].

2.3 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 [31]. Therefore, the temperature must not be considered a constant parameter, but must be introduced as an additional solution variable [32, 33, 34, 35].

Thermal coupling can be modeled by a thermal circuit [31, 36] (cf. Fig. 8). 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 [37] 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.

2.4 Device Simulation

The vast majority of todays routinely performed device simulations are based on a numerical solution of the basic semiconductor equations which include drift-diffusion current relations [37, 38, 39, 40]. 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 [41, 42, 43, 44, 45, 46, 47] 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 [48, 49].

2.5 Mixed Mode Simulation

Several works dealing with circuit simulation using distributed devices have been published so far [17, 18, 19, 50]. 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 [18] 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 [19] 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 ± 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.

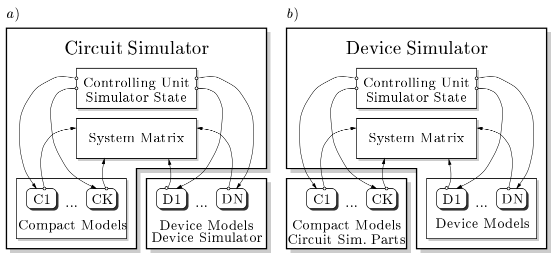


Figure 9: Comparison of the two different strategies: a) Device simulator as client. b) Device simulator as server

A comparison of these different architectures is shown in Fig. 9. In Fig. 9a the device simulator acts as a client to the circuit simulator whereas in Fig. 9b the device simulator is extended with circuit simulator capabilities and can reuse circuit simulator models on demand.

2.6 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 [51, 52]. In MINIMOS-NT a purely heuristic method is used which takes the exponential relation between the potential and the carrier concentration into consideration [53]. 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 [17] 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 ± 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 [17] that a solution can only be obtained for an initial-guess as close to the solution as ± 0.2V 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.* [54] for FET circuits using compact models. They proposed to shunt a resistor of 3 kΩ 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

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

$$G_{min} = 10^{-12}\,S \quad (3)$$

$$G_S^k = \max\left(G_{min},\ G_0 \cdot 10^{-k/\kappa}\right) \quad (4)$$

$$\kappa = 1.0 \ldots 4.0 \quad (5)$$

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.

2.7 Examples

2.7.1 Five-Stage CMOS Ring Oscillator

A five-stage CMOS ring oscillator circuit is shown in Fig. 10. For both the NMOS and the PMOS transistors a device width of $W = 1\ \mu$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 [55]. 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 φ_{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 ($L_G = 2\ \mu$m), the other one with short-channel transistors ($L_G = 0.2\ \mu$m). For the long-channel transistors, the simulation

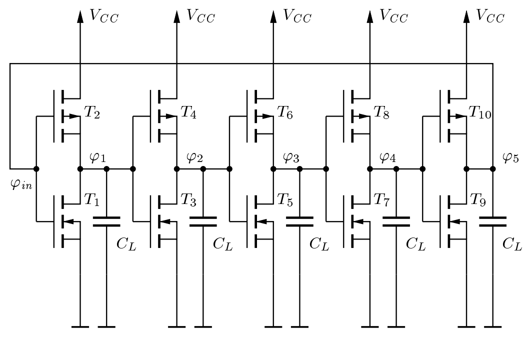


Figure 10: Five-stage CMOS ring oscillator

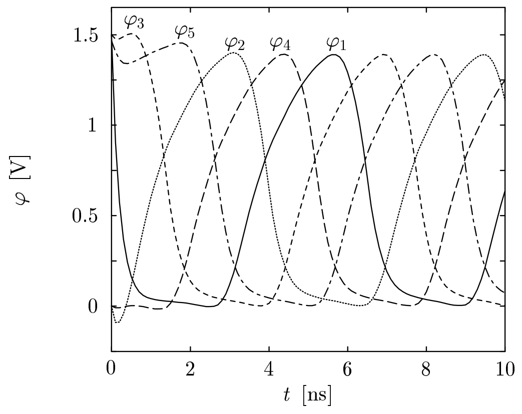


Figure 11: Node voltages of the long-channel five-stage CMOS ring oscillator. Drift-diffusion and hydrodynamic simulation results match perfectly.

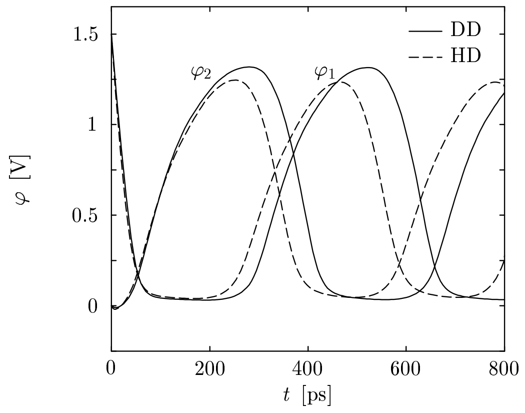


Figure 12: Node voltages φ_1 and φ_2 of the short-channel five-stage CMOS ring oscillator for drift-diffusion and hydrodynamic simulation.

results obtained with the drift-diffusion and hydrodynamic transport models agree so closely, that in the graph no differences are visible (cf. Fig. 11). The simulation results for the short-channel devices are shown in Fig. 12. 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 $\tau_{\mathrm{DD}} \approx 30$ ns and $\tau_{\mathrm{HD}} \approx 26$ 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 %.

2.7.2 Five-Stage CML Ring Oscillator

A current mode logic (CML) gate is an emitter coupled logic (ECL) gate stripped of the emitter-follower [55, 56]. The gain of a single stage without load can be approximated by assuming a simple Ebers-Moll model for the transistors [57] to be approximately -5. When considering an inverter chain consisting of 5 CML inverters as shown in Fig. 13 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

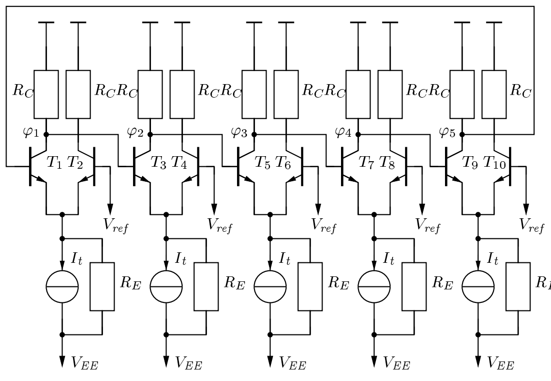


Figure 13: Five-stage CML ring oscillator

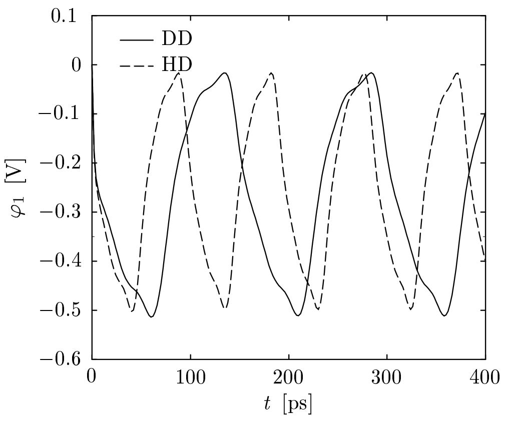


Figure 14: Oscillation of node voltage φ_1 of the five-stage CML ring oscillator. Large discrepancies between drift-diffusion and hydrodynamic simulations are observed.

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 $f_{\mathrm{DD}} = 6.8\ \mathrm{GHz}$ for the drift-diffusion and $f_{\mathrm{HD}} = 10.6\ \mathrm{GHz}$ for the hydrodynamic model which gives a relative difference of 36% for the drift-diffusion model (Fig. 14). 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.

2.7.3 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–$100\ \mathrm{mW}$ 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 [36, 58]. 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 forward-biased pn-junctions is known to be approximately $-2\ \mathrm{mV/K}$, 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 μA709 [31, 59] as shown in Fig. 15 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 [7] (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. 16. The thermal conductances were assumed to be $G_1 = G_2 = 2\ \mathrm{mW/K}$ and $G_9 = G_{15} = 10\ \mathrm{mW/K}$ while the coupling mismatch was modeled by $G_{1,9} = G_{1,15} = G_k = 10\ \mathrm{mW/K}$ 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 [36].

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. 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

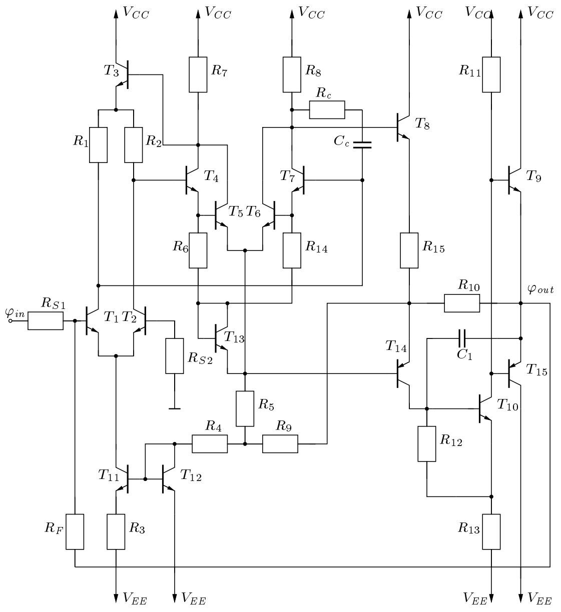


Figure 15: Schematic of the μA709 OpAmp.

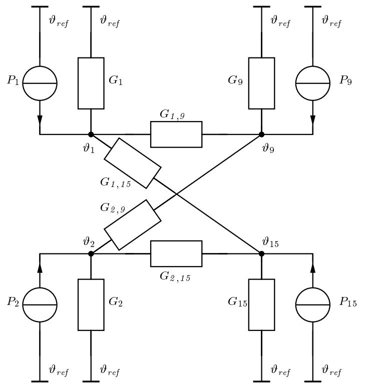


Figure 16: Thermal equivalent circuit used to simulate thermal interaction for the μA709 OpAmp.

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 φ_{in} from -1 mV to 1 mV with $\Delta\varphi_{in} = 20\,\mu$V. From SPICE simulations the open-loop gain of the μA709 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 φ_{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.

The DC transfer characteristic is shown in Fig. 17 with the obvious humps resulting from thermal feedback effects. In Fig. 18 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 [33, 34]. For stronger coupling, even the sign of the open-loop voltage gain may change and cause the OpAmp to become unstable [58].

The maximum temperature and the contact temperature of the output stage are shown in Fig. 19. 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 β 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 $I_C = 20\,\mu$A only self-heating is negligible and the contact temperature resembles the heat transfered 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 mK. Even this small temperature difference has a strong impact on the output characteristic due to the high gain of the circuit.

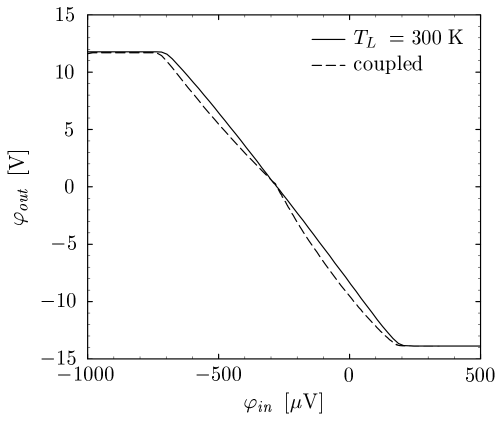


Figure 17: DC transfer characteristic of the μA709 for constant lattice temperature and considering thermal coupling of the input and output stage.

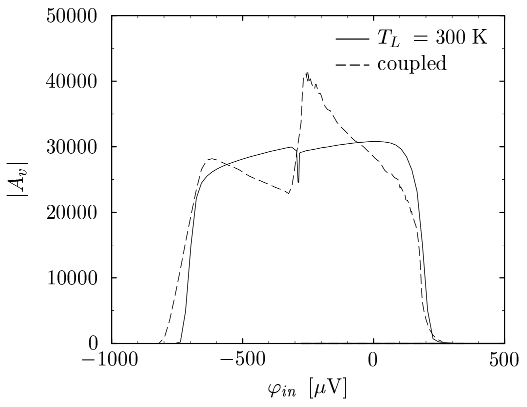


Figure 18: Open-loop gain of the μA709 for constant lattice temperature and considering thermal coupling of the input and output stage.

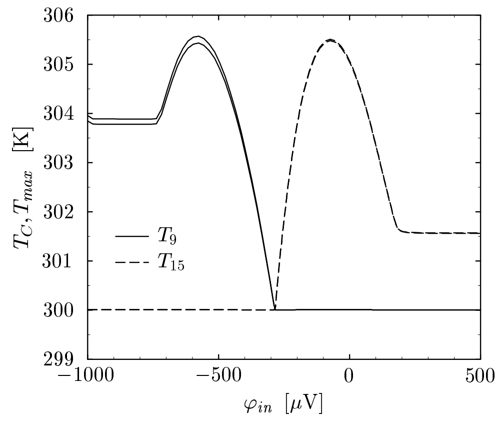


Figure 19: Maximum and contact temperature of the output transistors T_9 and T_{15} during the DC transfer characteristic.

3 Numerical Aspects of the Simulation of Two-Dimensional Ferroelectric Hysteresis

3.1 Introduction

The advancing development of nonvolatile memory cells leads to structures which make use of the hysteretic properties of ferroelectric materials. In order to deal with two-dimensional device structures like ferroelectric memory field effect transistors (FEMFET) [60] schematically outlined in Fig. 20, a two-dimensional hysteresis simulator has been developed.

The simulation of the two-dimensional hysteresis curve leads to the nontrivial problem of field rotation [61][62] and requires the calculation of a set of parameters for the non linear locus curves at each grid point. Aside from calculating the exact field distribution a simulator for ferroelectric devices has to fulfill further properties: To allow the calculation of transfer characteristics it has to be insensitive to the magnitude of the applied voltage steps. To keep pace with future developments of ferroelectric devices, the expansion of the algorithm to three dimensions should be possible.

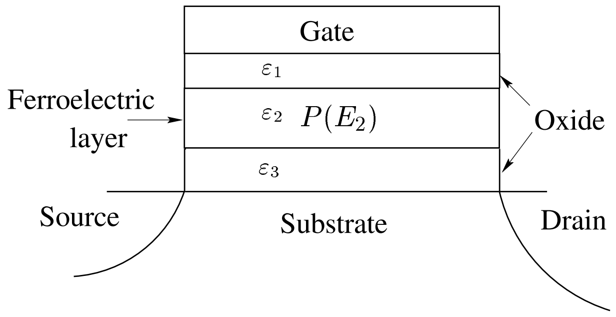


Figure 20: Ferroelectric nonvolatile memory field effect transistor.

3.2 Geometric algorithm

A general algorithm capable of fulfilling the numerical and physical constraints outlined above was introduced in [63]. It lays special focus on the simulation of field rotation. This enables the calculation of the lag angle χ between the electric field and the dielectric displacement that appears when the direction of the electric field changes.

The algorithm is based on splitting the polarization vector into orthogonal components with respect to the direction of the electric field at the next operating point. By means of these two components two corresponding locus curves of the hysteresis are calculated, leading to two polarization components, one parallel and one orthogonal to the electric field. These curves yield the polarization in direction of the electric field and some sort of remanent polarization in the orthogonal direction, thus forming a primary guess $\vec{P}_0$ for the next polarization (Fig. 21).

According to the fact that the saturation polarization is forming an upper limit, the component orthogonal to the electric field is reduced if necessary. This leads to the actual polarization vector $\vec{P}_1$ and the lag angle χ and is schematically outlined in Fig. 22.

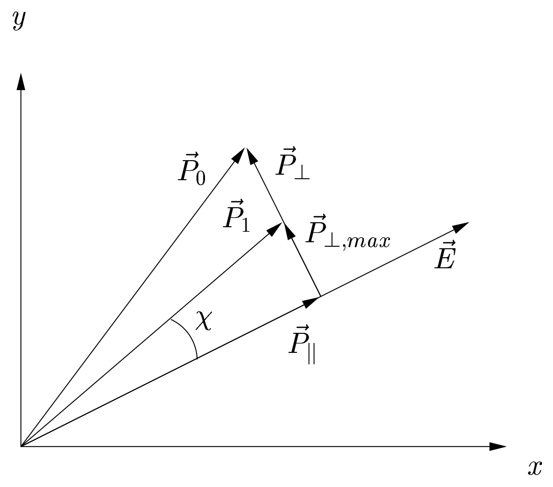


Figure 21: Calculation of the resulting polarization.

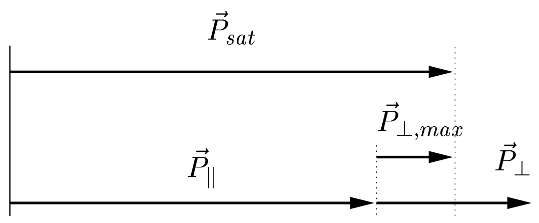


Figure 22: Reduction of the orthogonal component with respect to the saturation.

3.3 Numerical problems

As a consequence of the general approach, the following numerical aspects have to be considered in order to allow a two-dimensional simulation with the box integration method:

- Nonsymmetry of the locus curves
- Influence of previous operating points
- Selection of the shape of the hysteresis curve
- Detection of the correct locus curve

3.3.1 Nonsymmetry of the locus curves

In contrast to most of the functions used in device simulation, the locus curves of the hysteresis are nonsymmetric functions of the absolute value of the applied electric field $||\vec{E}||$. Different to common properties of symmetric functions, a criterion has to be established which decides whether the argument of the function is treated as positive or negative. Accordingly the orientation of the field compared with the box boundary becomes decisive to the sign of the function argument. Furthermore, it cannot be assumed that the sign of the resulting flux of the electric displacement has the same orientation as the applied electric field.

3.3.2 Influence of the previous operating points

The calculation of the current locus curves uses the parallel and orthogonal component of the previously applied fields in respect to the current electric field. This is a major difference to one-dimensional simulation, which only requires the storage of selected turning points [64]. In order to deal with this information in a suitable way it is necessary to make adjustments to the field discretization. It is intuitive that the history information is required on the box boundary and that it cannot be derived from a representation in the grid points alone, as it is suitable for non hysteretic properties [53].

In case of one-dimensional properties, the orthogonal component has zero length and the parallel component equals the previous electric field, which means that also this special case is covered by the algorithm. In order to speed up the computation the locus curves are not calculated from the last turning point but rather from the saturation polarization, so the lancette curves will not exactly fit the one-dimensional hysteresis model. The resulting locus curves of this approach are plotted in Fig. 23. They were achieved as output of the simulation of a capacitor with ferroelectric dielectric.

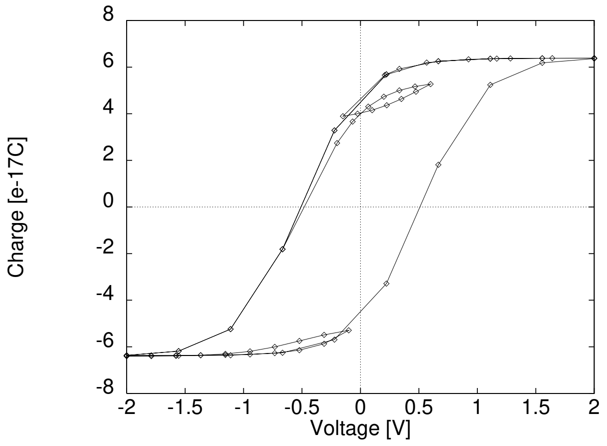


Figure 23: Simulation of a ferroelectric capacitor.

3.3.3 Selection of the shape of the hysteresis curve

Considering the fact that a different locus curve has to be calculated on each box boundary, it is necessary to chose analytic functions. Numerical methods as described in [65] cannot be applied. In order to overcome these difficulties a model was implemented into the device simulator MINIMOS-NT which describes all hysteresis curves by $tanh$ functions derived from analytical calculations.

3.3.4 Detection of the locus curve

A sophisticated task is to calculate the locus curves for a new operating point. As outlined in Fig. 24 one of two possible locus curves has to be chosen at each operating point, depending on the history of the electric field [64].

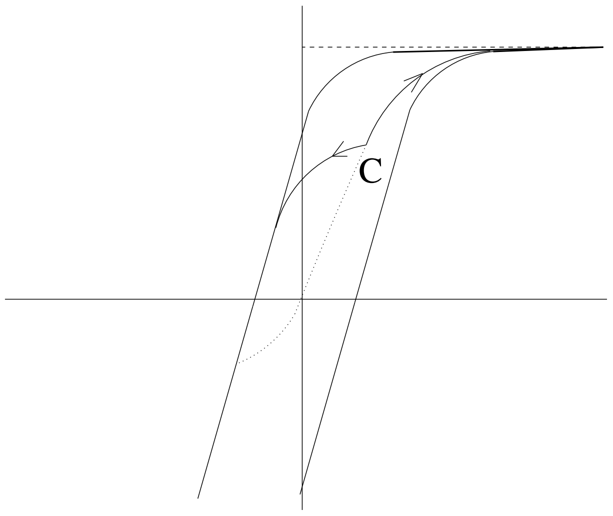


Figure 24: Possible locus curves in an operating point

As a consequence of the two-dimensional algorithm the common starting point C of these two branches will move during the nonlinear iteration. In fact it highly depends on the assumed electric field. Therefore it cannot be guaranteed that the same branch is selected at each iteration step. Regarding the different derivatives of the two functions this will lead to poor convergence and in worst case to oscillations of the nonlinear iteration.

As practice shows a preselection of the correct branch is necessary to achieve convergence, especially for the simulation of complex structures. A suitable approach to detect the direction of the change of the electric field is to solve a linearized equation system. In this system the polarization is kept constant and only the linear part of the dielectric displacement is modified. Using this method an approximation to the electric field in the new operating point is derived.

Based on this information it has to be decided whether the electric field was increased or not. The straightforward approach to compare the absolute values of the old and the new electric field will obviously fail, even if the two field vectors are parallel. For the applied algorithm the parallel component of the old field vector is calculated, and the result is interpreted in dependence of the orientation of the new field vector as outlined in Fig. 25 and Fig. 26.

With this information it is now possible to select the correct branch of the hysteresis curve. The complete scheme is outlined in Fig. 27.

3.4 Simulation results

The algorithm was implemented into the device simulator MINIMOS-NT[63]. A FEMFET was constructed by inserting a ferroelectric segment in the sub-gate area of the NMOS, as outlined in Fig. 20. The operating point of the ferroelectric material was chosen on the initial polarization curve. In this case the

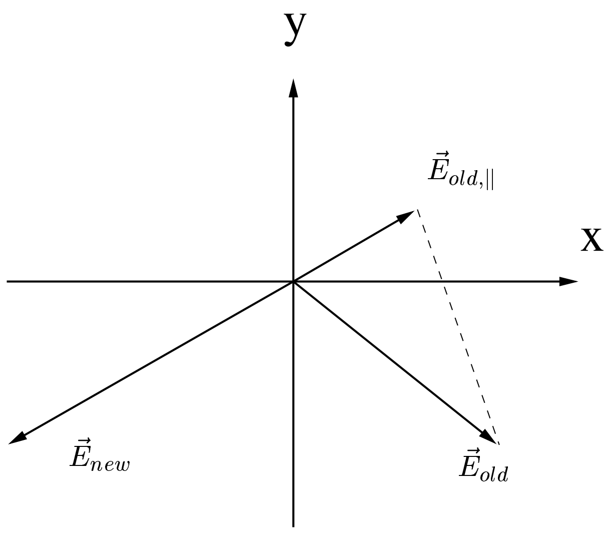


Figure 25: Detection of the change of the electric field, electric field decreases.

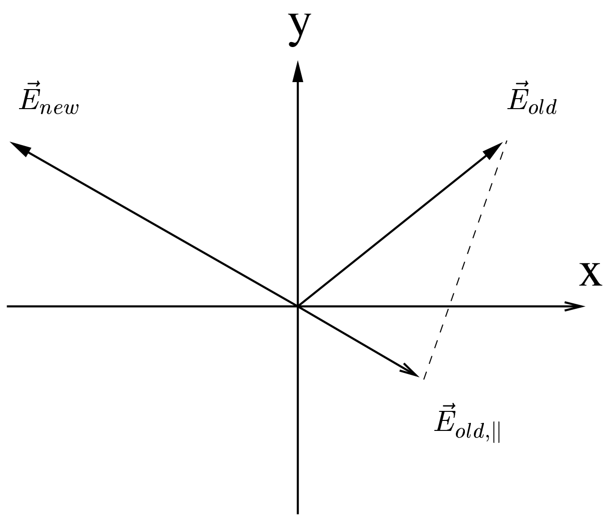


Figure 26: Detection of the change of the electric field, electric field increases.

ferroelectric polarization increases the displacement and leads to a significant higher space charge density in the channel area. This will cause a higher drain current of the FEMFET for the same gate voltage as for the NMOS. As a result of the hysteretic behavior of the polarization the drain current of the device does not only depend on the gate voltage but also on the history of the gate voltage. So the I–V characteristics of the transistor show also a hysteresis which allows the use of the device as a nonvolatile memory. Fig. 28 sketches the simulated I–V characteristics for NMOS and FEMFET obtained by sweeping the gate voltage from zero to saturation and vice versa The threshold voltage of the NMOS was 0.7 V and 0.6 V for the FEMFET. The bulk voltage was set to 0.5 V, the drain voltage to 0.1 V. We received a voltage shift of 0.1 V. Caused by the higher displacement, the drain current of the FEMFET is significantly increased compared to the NMOS.

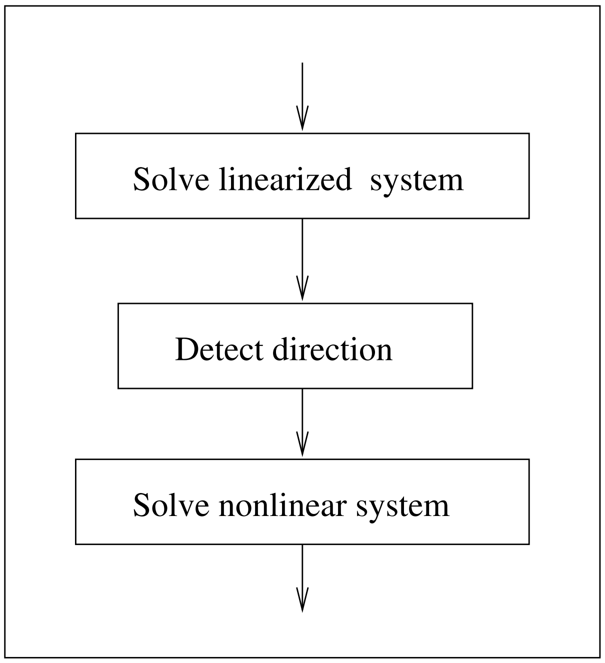


Figure 27: Modified trivial iteration scheme

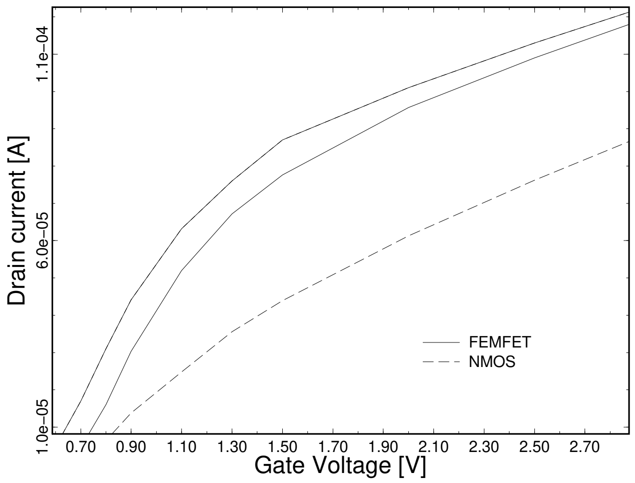


Figure 28: Simulated I-V characteristics of FEMFET and NMOS

References

[1] A. Geist, A Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V.Sunderam. *PVM: Parallel Virtual Machine — A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.

[2] T. J. Mowbray and R. Zahavi. *The Essential CORBA*. OMG/Wiley, 1995.

[3] D. W. Walker. The Design of a Standard Message-Passing Interface for Distributed Memory Concurrent Computers. *Parallel Computing*, 20(4):657–673, 1994.

[4] M. Neeracher. *Scheduling for Heterogeneous Opportunistic Workstation Clusters*. Hartung-Gorre, Konstanz, 1198.

[5] Platform Computing Corporation, Toronto, Canada. *LSF Load Sharing Facility*, 1998. `www.platform.com`.

[6] IBM. Advanced Statistical Analysis Program (ASTAP), Program Reference Manual. Technical Report SH20-1118-0, IBM, 1973.

[7] L.W. Nagel. SPICE2: A Computer Program to Simulate Semiconductor Circuits. Technical Report UCB/ERL M520, University of California, Berkeley, 1975.

[8] 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.

[9] 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.

[10] Swiss Federal Institute of Technology, Lausanne. The EPFL-EKV MOSFET Model Equations for Simulation. http://legwww.epfl.ch/ekv/model.html, 1999.

[11] Philips. Philips Semiconductors Site on Compact Models. http://www-eu2.semiconductors.com/Philips_Models, 1998.

[12] ISE Integrated Systems Engineering. *ISE TCAD Manuals vol. 4, release 4*, 1997.

[13] 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.

[14] Technology Modeling Associates, Inc., Palo Alto, CA. *TMA MEDICI, Two-Dimensional Device Simulation Program, Version 2.0*, 1994.

[15] 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.

[16] M.R. Pinto. *PISCES IIB*. Stanford University, 1985.

[17] J.G. Rollins and J. Choma. Mixed-Mode PISCES-SPICE Coupled Circuit and Device Solver. *IEEE Trans.Computer-Aided Design*, 7:862–867, 1988.

[18] K. Mayaram and D.O. Pederson. Coupling Algorithms for Mixed-Level Ciruit and Device Simulation. *IEEE Trans.Computer-Aided Design*, 11(8):1003–1012, 1992.

[19] 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.

[20] S. Selberherr. *Zweidimensionale Modellierung von MOS-Transistoren*. Dissertation, Technische Universität Wien, 1981.

[21] T. Grasser, V. Palankovski, G. Schrom, and S. Selberherr. Hydrodynamic Mixed-Mode Simulation. In Meyer and Biesemans [66], pp 247–250.

[22] T. Simlinger. *Simulation von Heterostruktur-Feldeffekttransistoren*. Dissertation, Technische Universität Wien, 1996.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] A. Stach. Simulation von MOSFET-Schaltungen. Diplomarbeit, Technische Universität Wien, 1995.

[29] U. Tietze and C. Schenk. *Halbleiter-Schaltungstechnik*. Springer, Berlin, 1971.

[30] 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.

[31] P.R. Gray and R.G. Meyer. *Analysis and Design of Analog Integrated Circuits*. Wiley, 1993.

[32] P.C. Munro and F.Q. Ye. Simulating the Current Mirror with a Self-Heating BJT Model. *IEEE J.Solid-State Circuits*, 26(9):1321–1324, 1991.

[33] K. Fukahori and P.R. Gray. Computer Simulation of Integrated Circuits in the Presence of Electrothermal Interaction. *IEEE J.Solid-State Circuits*, SC-11(6):834–846, 1976.

[34] S.S. Lee and D.J. Allstot. Electrothermal Simulation of Integrated Circuits. *IEEE J.Solid-State Circuits*, 28(12):1283–1292, 1993.

[35] 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.

[36] K. Nemeth. On the Analysis of Nonlinear Resistive Networks Considering the Effect of Temperature. *IEEE J.Solid-State Circuits*, 1(1):550–552, 1976.

[37] S. Selberherr. *Analysis and Simulation of Semiconductor Devices*. Springer, 1984.

[38] 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.

[39] W. Fichtner, D.J. Rose, and R.E. Bank. Semiconductor Device Simulation. *IEEE Trans.Electron Devices*, ED-30(9):1018–1028, 1983.

[40] F. Assad, K. Banoo, and M. Lundstrom. The Drift-Diffusion Equation Revisited. *Solid-State Electron.*, 42(3):283–295, 1998.

[41] K. Blotekjaer. Transport Equations for Electrons in Two-Valley Semiconductors. *IEEE Trans.Electron Devices*, ED-17(1):38–47, 1970.

[42] P.T. Landsberg and S.A. Hope. Two Formulations of Semiconductor Transport Equations. *Solid-State Electron.*, 20:421–429, 1977.

[43] G. Baccarani and M.R. Wordeman. An Investigation of Steady-State Velocity Overshoot in Silicon. *Solid-State Electron.*, 28(4):407–416, 1985.

[44] M. Rudan and F. Odeh. Multi-Dimensional Discretization Scheme for the Hydrodynamic Model of Semiconductor Devices. *COMPEL*, 5(3):149–183, 1986.

[45] 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.

[46] C. Lab and Ph. Caussignac. An Energy-Transport Model for Semiconductor Heterostructure Devices: Application to AlGaAs/GaAs MODFETs. *COMPEL*, 18(1):61–76, 1999.

[47] 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.

[48] T. Grasser, H. Kosina, and S. Selberherr. Consistent Comparison of Drift-Diffusion and Hydro-Dynamic Device Simulation. In *Simulation of Semiconductor Processes and Devices*, Kyoto, 1999.

[49] T. Grasser. *Mixed-Mode Device Simulation*. Dissertation, Technische Universität Wien, 1999.

[50] W. Engl, R. Laur, and H.K. Dirks. MEDUSA - A Simulator for Modular Circuits. *IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems*, 1(2):85–93, 1982.

[51] R.E. Bank and D.J. Rose. Global Approximate Newton Methods. *Numer.Math.*, 37:279–295, 1981.

[52] 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.

[53] C. Fischer. *Bauelementsimulation in einer computergestützten Entwurfsumgebung*. Dissertation, Technische Universität Wien, 1994.

[54] 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.

[55] S.H.K. Embabi. *Digital BiCMOS Integrated Circuit Design*. Kluwer, 1993.

[56] R.L. Treadway. DC Analysis of Current Mode Logic. *IEEE Circuits and Devices Magazine*, pp 21–35, March 1989.

[57] P. Antognetti and G. Massobrio. *Semiconductor Device Modeling with SPICE*. McGraw-Hill, 1988.

[58] J.E. Solomon. The Monolithic Op Amp: A Tutorial Study. *IEEE J.Solid-State Circuits*, SC-9(6):314–332, 1974.

[59] National Semiconductors. Product Folder. http://www.nsc.com/pf/LM/LM709.html, 1999.

[60] S. L. Miller and P. J. McWhorter. Physics of the Ferroelectric Nonvolatile Memory Field Effect Transistor. *J.Appl.Phys.*, 72(12):5999–6010, 1992.

[61] P. Weiss and V. Planer. Hystérèse dans les Champs Tournants. *Journal de Physique (Théor. et Appl.)*, 4/7:5–27, 1908.

[62] H. Pfützner. Rotational Magnetization and Rotational Losses of Grain Oriented Silicon Steel Sheets–Fundamental Aspects and Theory. *IEEE Trans.Magnetics*, 30(5):2802–2807, 1994.

[63] K. Dragosits, M. Knaipp, and S. Selberherr. Two-Dimensional Simulation of Ferroelectric Non-volatile Memory Cells. In Meyer and Biesemans [66], pp 368–371.

[64] B. Jiang, P. Zurcher, R.E. Jones, S.J. Gillespie, and J.C. Lee. Computationally Efficient Ferroelectric Capacitor Model for Circuit Simulation. In *IEEE Symposium on VLSI Technology Digest of Technical Papers*, pp 141–142, 1997.

[65] H.G. Brachtendorf, Ch. Eck, and R. Laur. Macromodeling of Hysteresis Phenomena with SPICE. *IEEE Trans.Circuits and Systems–II: Analog and Digital Signal Processing*, 44(5):378–388, 1997.

[66] K. De Meyer and S. Biesemans, editors. *Simulation of Semiconductor Processes and Devices*, Leuven, Belgium, 1998. Springer.