VISTA Status Report
December 1996

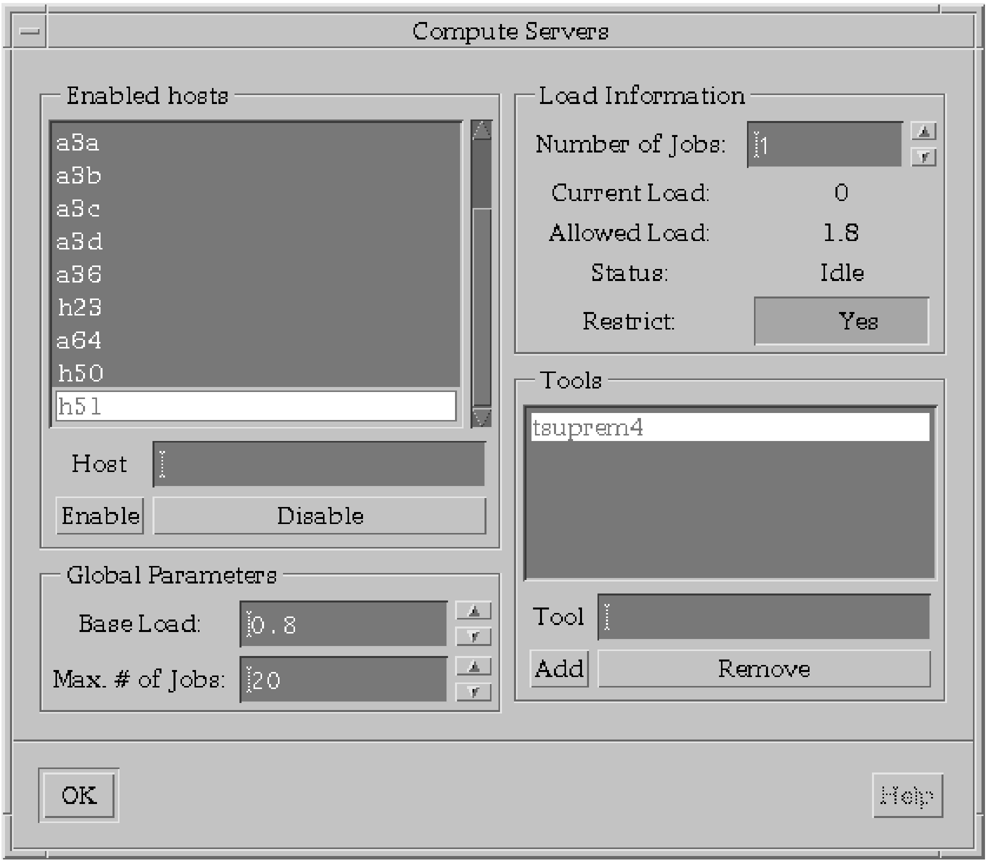
H. Kirchauer, R. Plasun, M. Radi, S. Selberherr, R. Strasser

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

Contents

1 Job Farming

VISTA is able to make use of any host in your LAN for your computations, provided that it is reachable via remote shell (rsh) and that the path to your working directory is the same on all machines. In order to register a computer with the VISTA shell you have to open the *Compute Servers* panel in the *Job-Control* menu as depicted in Fig. 1.



Compute Servers

Enabled hosts

a3a
a3b
a3c
a3d
a36
h23
a64
h50
h51

Load Information

Number of Jobs: 1
Current Load: 0
Allowed Load: 1.8
Status: Idle
Restrict: Yes

Tools

tsuprem4

Global Parameters

Base Load: 0.8
Max. # of Jobs: 20

Figure 1: The Compute Server Panel is used to enable/disable hosts, to configure specific tools for a host, and to specify parameters for load balancing.

1.1 Host Registration

The *Enabled Hosts* area of Fig. 1 displays a list of all hosts known to the framework so far. To enable additional hosts, enter their name into the *Host* field and press the *Enable* button.

1.2 Host Configuration

1.2.1 Workload Parameters and Status Information

To configure a host you have to select its entry in the list of enabled hosts. The right hand part of the panel displays information about the configuration of the current selected host. The *Load Information* area displays the current status of a host and enables you to specify how many jobs should be run on that host. Additionally it displays the current load, the allowed (or desired) load on that host and whether that host is considered *idle* or *busy* by the framework. For information about load balancing refer to Section 1.3.

1.2.2 Host Usage Restrictions

If you do not want a host to be included for job farming, but you need to run a specific tool on that host, you can make it a restricted host. This will cause the framework to use this host, if a tool is only available on that host. Typically tools that demand a license require this feature.

1.2.3 Tool Information

In order to let the framework decide on which host it should run a job, you have to specify which tools are available on each host. This is done in the *Tools* area. It displays a list of tools that are available on the current selected host. You can interactively add and remove tools. Once defined, VISTA knows on which host a specific tool can be invoked. If a tool is not registered on any host, it is assumed to be available everywhere.

1.3 Load Balancing

In a large cluster of workstations it is important to balance the load between all available nodes. Typically large simulations experiments require a lot of system jobs, which should make use of all CPUs available in the environment. At the same time computations on computers with a high workload should be avoided, as it might cause your simulation to get stuck.

Therefore VISTA enables you to make optimal use of your computer resources. VISTA allows to specify a global load limit (*Baseload* which applies to any known host) and host specific load limits (*Number of Jobs*). The allowed workload (which is displayed in *Allowed Load*) is

$$w_{limit} = baseload + njobs.$$

VISTA queues jobs to a host as long as the allowed load will not be exceeded *after* the job is submitted (Typically one computing process causes a load increase of one). The idle/busy limit of a host is

$$w_{idle/busy} = w_{allowed} - 1.$$

If the current load is above this limit no jobs will be queued to this host. The status is displayed in the panel (Busy/Idle).

Basically hosts are sorted by their remaining load for queuing, which is the difference between their allowed load and their current load. That means that you can favor hosts over others by specifying a higher number of jobs for them.

The global load limit makes sense in situations where workload is high and you want to raise all limits at the same time. Usually the baseload parameter should be between one and two, which avoids computations on machines with running jobs.

1.4 Workload Feedback

VISTA collects load information periodically using the `rup` command. Polling intervals are configurable, good results have been achieved with a maximum interval of 60 seconds and a minimum interval of 10 seconds. `rup` utilizes the `rstatd` which should be configured on every host you use. If VISTA is unable to determine the load this way, it assumes your host to be idle.

2 High-Level Optimization Capabilities of the VISTA Framework

Submicron semiconductor development relies on TCAD essentially in the process optimization stages. For these highly complex tasks, there are special facilities available in the VISTA TCAD framework.

Optimization problems can be devided into two groups:

- The first one contains the optimization of extracted physical parameters in a given set of input parameter space. Direct optimization methods are not very well suited for this problem because in order to calculate one set of input and output values a whole proces flow and usually several device tests must be simulated. To save CPU-time, methods like Design of Experiments (DoE) and Response Surface Methodology (RSM) have to be used [1][2].
- The second class of problems are concerned with the calibration of simulator modules. However, in this case only a single simulator – eventually with pre- and postprocessing – needs to be executed. Direct optimization methods work very well for this task.

This article will concentrate on the first type of problem.

2.1 Design of Experiments

For automatic generation of experiments a DoE module can be used. The type of the experimental design can be chosen out of a large number of available types (Table 1). The result of the DoE is the *design matrix*. To probe the parameter space in the most efficient way, transformations for the parameters can be defined; see Section 2.3 for details. After simulating these runs the control and the response values are written to the *experiment table* and can be processed by other tools.

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

Table 1: Available experimental designs

2.2 Response Surface Methodology

The RSM module fits polynomial functions to the data in the *experiment table.* As in the DoE module, additional transformations for the controls and the responses can be added; see Section 2.3 for details. With the RSM-viewer the fitted surface can be visualized as 2D or 3D graphics (Figure 4). Sliders enable the user to understand the influence of the controls.

2.3 Transformations

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

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

It is important to note that all transformation functions have to be defined by specifying code for both the forward and reverse directions and assigning a reference name to the transformation before they can be used. All information on transformations is stored centraly and accessed exclusively by the reference name. E.g., for a given technology, a transformation called `vth-lg` can be defined, which analytically reflects the short-channel effect and is used to linearize the dependence of the threshold voltage on the gate length for DoE and RSM.

2.4 Optimizer

For optimizing device performance parameters over a given input variable space, a constrained optimizer with sequential quadratic approximations has been integrated. It minimizes the target function, wich can be assembled out of the input and output values. The gradient is calculated by evaluating finite differences, and the Hessian is built by an BFGS update. For the calibration problems an optimizer based on the Levenberg Marquardt algorithm [3] is available.

2.5 Architecture

The DoE module, the RSM module and the optimizer are external tools. The optimizer and the RSM module are interactive programs, so spezial interface agents handle communication between the framework and the tool. These agents are implemented as object classes and can easily subclassed to integrate additional tools.

2.6 Application

The following example shows how all these functions work together. The goal of the example is to make the n-well as deep as possible, but hold the threshold voltage at a fixed value. The control variables are the implant dose and energy and the drive time (Table 2).

Implant dose	4e12	12e12	cm^{-2}
Implant energy	110	270	keV
Drive time	10	270	min

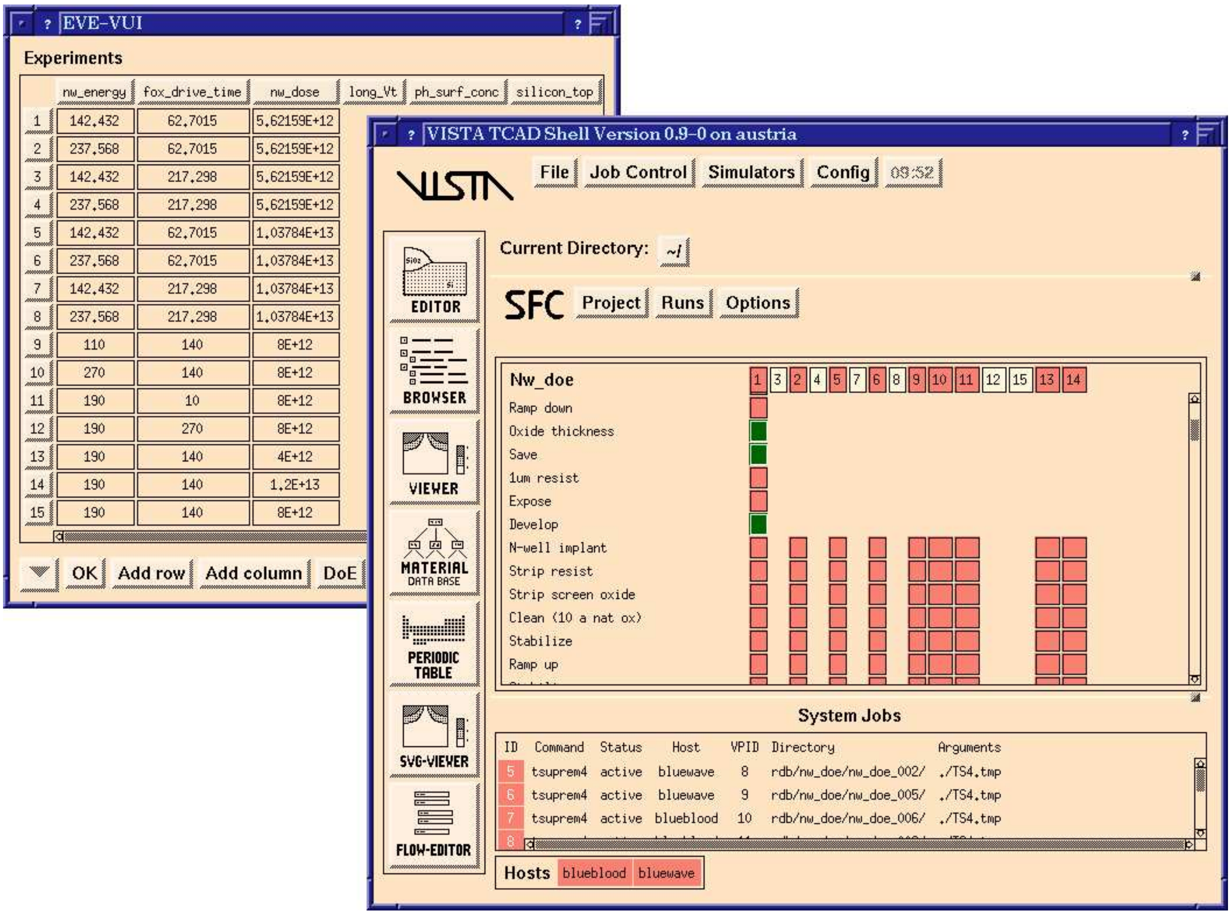
Table 2: Range of the control variables.

For simulating the process TSUPREM4 and for the threshold voltage extraction MEDICI were used. The listing (Figure 2) shows the definition of the problem. Both – controls and responses – have nicknames so they can be referenced in the script, e.g., for building the target function.

After the generation of a Central Composite Inscribed design and calculation of 15 runs (Fig. 3), a response surface model is generated and the optimization is started. The optimizer solves the constrained problems by querying the RSM-tool for results of the fitted surface. So the optimal process parameters (`nw_energy` $= 220.5$, `fox_drive_time` $= 203.8$, `nw_dose` $= 8.4E + 12$) for a deep well can be found. The surface plot (Figure 4) shows the characteristic near the optimal point.

```
;; load additional modules
(vos::load #"sf-vui")
(vos::load #"sfev-vui")
(vos::load #"optagent-2")

(vos::load #"sfvdoe")
(vos::load #"sfvrsm")
(vos::load #"optcalc")

;; set the flow and the design file
(Flow-Eve #"~/nwell/nw_doe.sfe" :file #"~/flows/nwell.des")

;; list of control variables
;;    nickname         stepname            name in step
(setq control-defs
  '(("nw_energy"      "n-well implant"    "energy")
    ("fox_drive_time" "stabilize-special" "time")
    ("nw_dose"        "n-well implant"    "dose")))

;; define these control variables
(dolist (cd control-defs)
  (Eve-Define-Control (car cd) (cadr cd) (caddr cd))
  (Eve-Set-Control (car cd) (fourth cd)))

;; set the range of the controls
(Eve-Set-Control "nw_energy"        190  :min 110   :max 270)
(Eve-Set-Control "fox_drive_time"   190  :min 10    :max 270)
(Eve-Set-Control "nw_dose"         8e12  :min 4e12  :max 12e12)

;; define responses
;;                    nickname           stepname    name in step
(Eve-Define-Response "long_Vt"           "Vt"        "Vt")
(Eve-Define-Response "ph_surf_conc"      "Xj's etc." "ph_surf_conc")
(Eve-Define-Response "silicon_top"       "Xj's etc." "silicon_top")
(Eve-Define-Response "abs_nwell_xj+0.5" "Xj's etc." "abs_nwell_xj+0.5")
(Eve-Define-Response "nwell_xj" "" "" :eval-expr
 '(and abs_nwell_xj+0.5 silicon_top
   (+ (- abs_nwell_xj+0.5 silicon_top) 0.5)))
(Eve-Define-Response "target" "" "" :eval-expr
 '(and abs_nwell_xj+0.5 silicon_top
   (+ (* 100 (abs (- long_Vt -0.8)))
    (- 2 (+ (- abs_nwell_xj+0.5 silicon_top) 0.5)))))

;; schow the run table
(sfc::vui-eve)

;; generate a CCI design
(Opt-Doe :design 'CCI)

;; start the optimizer when all runs are calculated
(flet ((runs-return (x cl ca)
(sfc::optimize-rsm :target "target")))
 (sfc::set-idle-callback #'runs-return nil))
```

Figure 2: Listing for solving this optimization problem.



Experiments

	nw_energy	fox_drive_time	nw_dose	long_Vt	ph_surf_conc	silicon_top
1	142.432	62.7015	5.62159E+12			
2	237.568	62.7015	5.62159E+12			
3	142.432	217.298	5.62159E+12			
4	237.568	217.298	5.62159E+12			
5	142.432	62.7015	1.03784E+13			
6	237.568	62.7015	1.03784E+13			
7	142.432	217.298	1.03784E+13			
8	237.568	217.298	1.03784E+13			
9	110	140	8E+12			
10	270	140	8E+12			
11	190	10	8E+12			
12	190	270	8E+12			
13	190	140	4E+12			
14	190	140	1.2E+13			
15	190	140	8E+12			

Current Directory: ~/

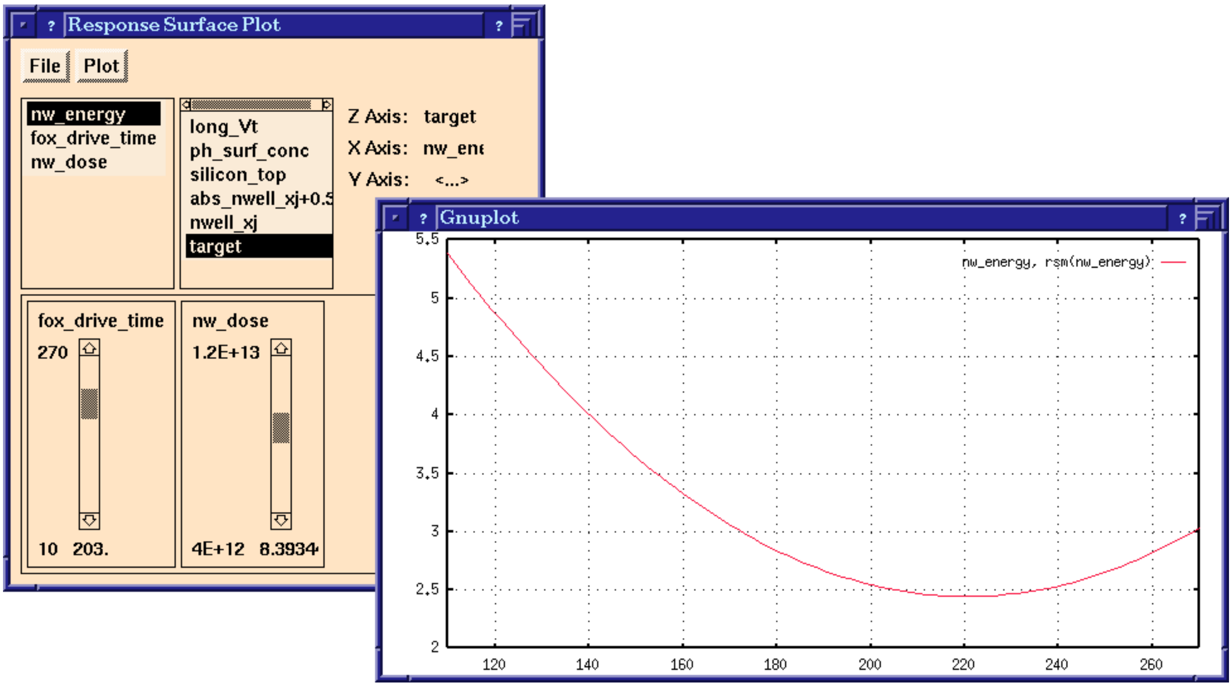
Nw_doe

System Jobs

```
ID  Command   Status  Host       VPID  Directory                 Arguments
5   tsuprem4  active  bluewave   8     rdb/nw_doe/nw_doe_002/    ./TS4.tmp
6   tsuprem4  active  bluewave   9     rdb/nw_doe/nw_doe_005/    ./TS4.tmp
7   tsuprem4  active  blueblood  10    rdb/nw_doe/nw_doe_006/    ./TS4.tmp
8
```

Hosts blueblood bluewave

Figure 3: Running the computation of the 15 CCI experiments



Z Axis: target

X Axis: nw_ene

Y Axis: <...>

fox_drive_time 270 10 203.

nw_dose 1.2E+13 4E+12 8.3934

Figure 4: Response Surface Model plot of the target function versus the energy in the optimal point.

3 Three-Dimensional Lithography Simulation

3.1 Introduction

Among all technologies photolithography holds the leading position in pattern transfer in today's semiconductor industry. The reduction of the lithographic feature sizes towards or even beyond the used wavelength and the increasing nonplanarity of the devices requires high standards of the lithography process. The large cost and time necessary for experiments make simulation an important and especially cost-effective tool for further improvements. However, a rigorous description of the fundamental physical effects governing sub-micrometer-photolithography poses considerable demands on the modeling, as three-dimensional simulation becomes necessary. As a consequence the needed computational resources are extremely high. In three dimensions some approaches (e.g. [4]) are suited only for supercomputers, others are already proposed for workstation based simulation (e.g. [5]).

We developed an overall three-dimensional photolithography simulator running on modern engineering workstations. The simulator consist of three different modules as shown in Fig. 5.

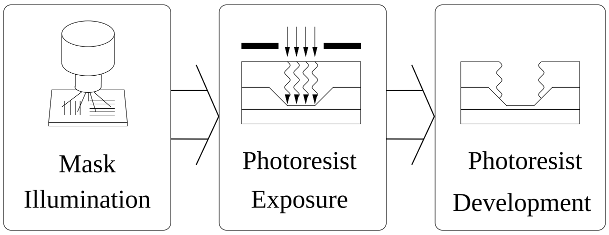


Figure 5: The three basic modules of the photolithography simulator

Each module accounts for one of the fundamental processes of photolithography, namely imaging, exposure/bleaching, and development. Each of the three processes requires its specific simulation approach and can be treated independently. The individual modules are briefly characterized as follows:

- The **imaging module** describes the illumination of the photo-mask. The light propagation through the optical system and the light transmission through the photo-mask has to be simulated. The output of the imaging module is the aerial image, which is the light intensity incidenting on top of the wafer.
- The **exposure/bleaching module** simulates the chemical reaction of the photosensitive resist. Thereby the light propagation within the optically nonlinear resist as well as electromagnetic scattering effects due to a nonplanar topography have to be modeled. The result of the exposure/bleaching module is the latent bulk image.
- The **development module** describes an isotropic etching process, whereby the etch rate is determined by the previously calculated bulk image. The final developed resist profile is the result of the overall photolithography simulator.

In the following sections we describe the different modules in greater detail and present simulation results for contact hole printing over planar and nonplanar substrate to demonstrate the capability of the simulator.

3.2 Imaging Simulation

Our aerial image simulator is based on a vector-valued extension [6] of the scalar theory of Fourier optics [7]. To apply this theory we first reduce the projection printing system to its essential parts as shown in Fig. 6.

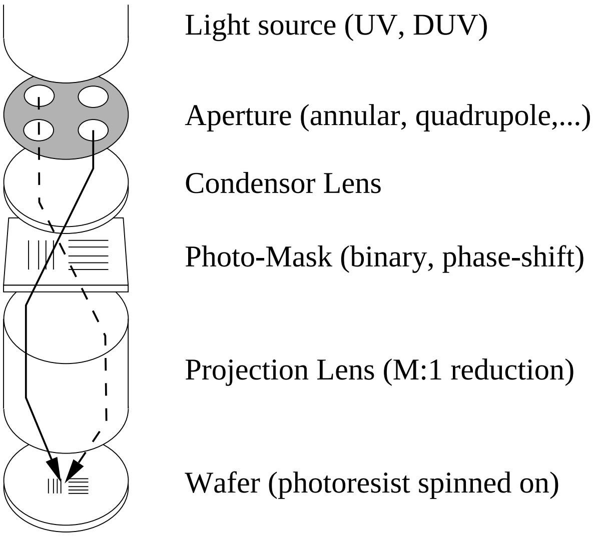

Light source (UV, DUV)
Aperture (annular, quadrupole,...)
Condensor Lens
Photo-Mask (binary, phase-shift)
Projection Lens (M:1 reduction)
Wafer (photoresist spinned on)

Figure 6: The projection printing system is reduced to its essential parts.

The mask-pattern is thereby assumed to be laterally periodic with periods a and b. The photomask is infinitesimally thin with ideal transitions of the transmission characteristic. The piecewise constant transmission function is real-valued (0 or 1) for binary masks, in case of phase-shift masks it is complex-valued with module less than one.

For the simulation of arbitrary illumination forms the distributed light source is discretized into mutually independent coherent point sources Q_{pq}. In Fig. 7 we show the discretization for an annular and a quadrupole aperture.

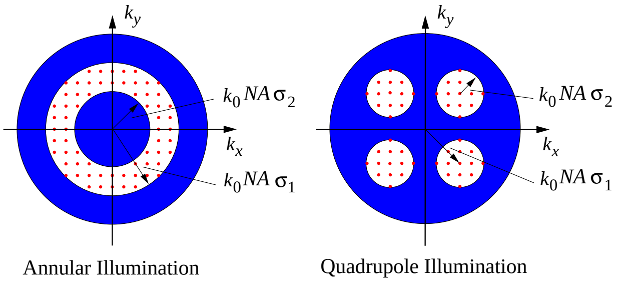


Figure 7: To account for arbitrary illumination forms the light source is discretized into mutually independent coherent point sources located inside the illumination aperture.

A numerically efficient exposure/bleaching simulation requires, that all the individual point source contributions are periodic. Therefore the spacing between the point sources Q_{pq} has to be chosen in a way, that the lateral wavevector components k_x^{pq} and k_y^{pq} of the waves incident onto the photo-mask equal an integer multiple of the sampling frequencies $2\pi/a$ and $2\pi/b$ in the Fourier domain, i.e.,

$$k_x^{pq} = p\frac{2\pi}{a}, \qquad k_y^{pq} = q\frac{2\pi}{b}.$$

This requirement is illustrated in the wavevector diagram of Fig. 8.

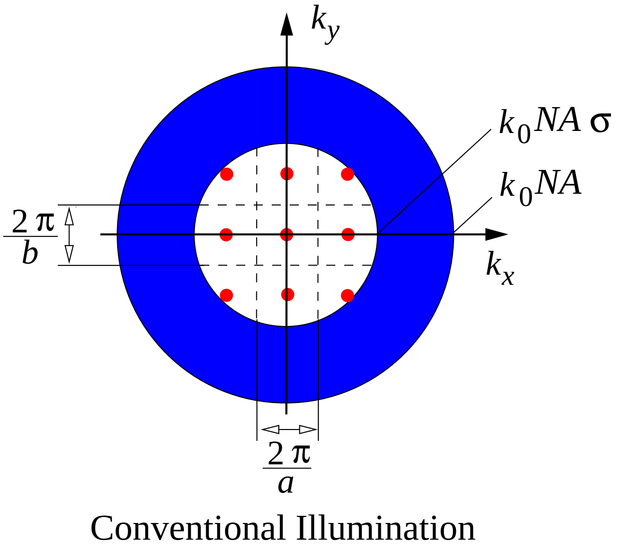


Figure 8: The spacing between the individual point sources is chosen to yield a periodic EM field incident onto the photo-mask.

The resulting image on top of the wafer due to one coherent point source can now be expressed by a superposition of discrete diffraction orders. Because of the periodicity of the EM field this superposition corresponds to a Fourier expansion and writes to

$$\mathbf{E}^{pq}(x,y,0) = \sum_n \sum_m \mathbf{E}^{pq}_{nm}\, e^{j2\pi(nx/a+my/b)}, \quad \mathbf{H}^{pq}(x,y,0) = \sum_n \sum_m \mathbf{H}^{pq}_{nm}\, e^{j2\pi(nx/a+my/b)}. \tag{1}$$

The diffraction orders are homogeneous plane waves with wavevectors

$$\mathbf{k}_{nm} = 2\pi \left[n/a,\ m/b,\ \sqrt{(1/\lambda_0)^2 - (n/a)^2 - (m/b)^2} \right]^{\mathrm{T}},$$

and their amplitudes follow from the vector-valued diffraction theory [6],

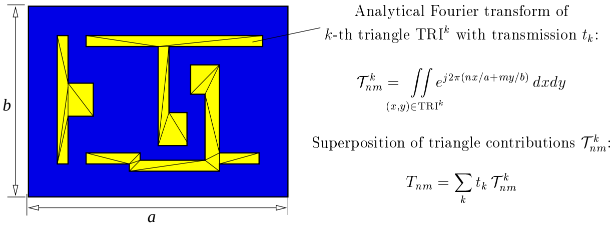
$$\mathbf{E}^{pq}_{nm} = T_{n-p,m-q}\, \mathbf{P}(n,m;p,q), \qquad \mathbf{H}^{pq}_{nm} = \frac{\lambda_0}{2\pi}\sqrt{\frac{\varepsilon_0}{\mu_0}}\, \mathbf{k}_{nm} \times \mathbf{E}^{pq}_{nm}. \tag{2}$$

λ_0 is the actinic wavelength and the time dependence of the EM field is a time-harmonic one, i.e., $\mathcal{E}(\mathbf{x},t) = \mathrm{Re}\left\{\mathbf{E}(\mathbf{x})e^{-j\omega_0 t}\right\}$ and $\mathcal{H}(\mathbf{x},t) = \mathrm{Re}\left\{\mathbf{H}(\mathbf{x})e^{-j\omega_0 t}\right\}$ with an angular frequency of $\omega_0 = 2\pi/\sqrt{\varepsilon_0\mu_0}\lambda_0$.

In (2) T_{nm} stands for the Fourier coefficients of the mask transmission function. As illustrated in Fig. 9 they are computed by first triangulating the piecewise constant transmission function and then superposing weighted analytical Fourier transforms of the triangular patterns.

The second term $\mathbf{P}(n,m;p,q)$ of (2) is the vector-valued counterpart to the pupil-function of the scalar diffraction theory [7] and follows from ray-tracing [6] trough the optical system (cf. Fig. 6). $\mathbf{P}(n,m;p,q)$ is essentially a low-pass filter (no evanescent waves can travel towards the wafer) and accounts for the polarization state, defocus, and higher order aberrations terms.

The aerial image itself is the light intensity $I(x,y)$ incident on top of the wafer and therefore equals the real part of the vertical component of the Poynting vector of the EM field. It is



Analytical Fourier transform of k-th triangle TRI^k with transmission t_k:

$$\mathcal{T}^k_{nm} = \iint_{(x,y)\in\mathrm{TRI}^k} e^{j2\pi(nx/a+my/b)}\,dxdy$$

Superposition of triangle contributions $\mathcal{T}^k_{nm}$:

$$T_{nm} = \sum_k t_k\,\mathcal{T}^k_{nm}$$

Figure 9: The Fourier coefficients T_{nm} of the mask transmission function are computed analytically.

calculated by a weighted incoherent superposition of the mutually independent terms and writes to

$$I(x,y) = \sum_p \sum_q w_{pq}\,\mathrm{Re}\Big[\mathbf{E}^{pq}(x,y,0) \times \mathbf{H}^{pq*}(x,y,0)\Big]_z,$$

whereby the asterisk denotes complex conjugation. For a uniform bright source the weights w_{pq} are determined by the portion of the discretization area $2\pi/a \times 2\pi/b$ within the illumination cone (cf. Fig. 8). A simulation result of the aerial image module is given in Fig. 10.

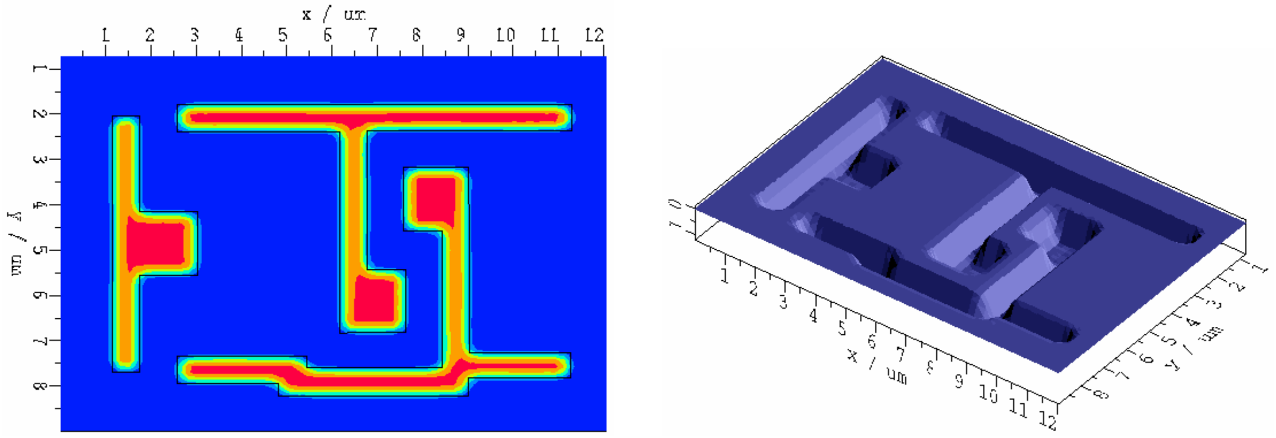


Figure 10: Aerial image of the mask pattern shown in Fig. 9

3.3 Exposure/Bleaching Simulation

From a simulation point of view the exposure/bleaching reaction is by far the most demanding problem within photolithography simulation. Hence, we discuss this module in greater detail. First we present the underlying physical simulation model and then come to the key point of any rigorous photolithography simulator, namely the numerical solution of the Maxwell equations.

3.3.1 Simulation Model

The exposure state of the photoresist is described by the concentration of the photoactive compound (PAC) $M(\mathbf{x};t)$, which constitutes the latent bulk image. The bulk image is transferred into the photoresist by light absorption. Thereby the PAC is dissolved and the optical properties, e.g., the refractive index $n(\mathbf{x};t)$, are changed. As usually this exposure/bleaching reaction is modeled by Dill's 'ABC'-model [8]

$$\frac{\partial M(\mathbf{x},t)}{\partial t} = -C\,I(\mathbf{x},t)\,M(\mathbf{x},t), \qquad n(\mathbf{x},t) = n_0 + j\frac{\lambda}{4\pi}\Big(A\,M(\mathbf{x},t) + B\Big),$$

where $I(\mathbf{x}, t)$ is the exposing light intensity. Consequently, the EM field inside the nonlinear photoresist has to be determined. Because the bleaching rate is small as compared to the frequency of the EM field, we apply a quasi-static approximation, i.e.,

$$M(\mathbf{x}, t_{k+1}) = M(\mathbf{x}, t_k) e^{-C\, I(\mathbf{x}, t_k)\, (t_{k+1} - t_k)}, \tag{3}$$

where the initial PAC distribution is homogeneous $M(\mathbf{x}, t_0) \equiv 1$. Furthermore, we assume a steady-state field distribution within a time step $t_k \le t < t_{k+1}$. Therefore, the EM field is time-harmonic and obeys the Maxwell equations in the form of

$$\operatorname{curl} \mathbf{H}_k(\mathbf{x}) = -j\omega_0\varepsilon_0\, \varepsilon_k(\mathbf{x})\, \mathbf{E}_k(\mathbf{x}), \qquad \operatorname{curl} \mathbf{E}_k(\mathbf{x}) = j\omega_0\mu_0\, \mathbf{H}_k(\mathbf{x}). \tag{4}$$

The complex permittivity $\varepsilon_k(\mathbf{x})$ is related to the refractive index n by Maxwell's law, $\varepsilon_k(\mathbf{x}) = n^2(\mathbf{x}, t_k)$, and the exposing light intensity is given by

$$I(\mathbf{x}, t_k) = \frac{1}{2}\sqrt{\frac{\varepsilon_0}{\mu_0}}\, n_0 \|\mathbf{E}_k(\mathbf{x})\|^2. \tag{5}$$

The equation set (3) to (5) represents the simulation model for the exposure/bleaching reaction, whereby an efficient solution of the inhomogeneous but linear Maxwell equations (4) is the crucial point for the applicability of the model. The principal simulation flow of the exposure/bleaching module is illustrated in Fig. 11.

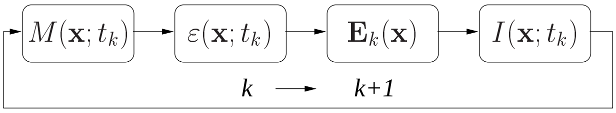


Figure 11: Simulation flow for the exposure/bleaching module

3.3.2 Numerical Solution of the Maxwell Equations

Our solution of the Maxwell equations [9] [10] corresponds to the three-dimensional formulation of the differential method such as an algorithm which was originally developed for the simulation of diffraction gratings [11] and was later adapted for two-dimensional photolithography simulation in [6]. The differential method itself requires a rectangular shaped simulation domain ($a \times b \times h$) with periodic boundary conditions in lateral direction. Inside the simulation domain arbitrary inhomogeneous and nonplanar regions can be simulated. Above ($z < 0$) and below ($z > h$) multiple planar homogeneous layers form a stratified medium, that can be treated analytically and are considered by the vertical boundary conditions. A typical formation is shown in Fig. 12.

The strategy behind the differential method is briefly described as follows: First, the dependence of the EM field on the lateral x- and y-coordinates is expressed by Fourier series. Inserting these expansions into the Maxwell equations transforms the partial differential equations (PDEs) into a system of ordinary differential equations (ODEs). Once the boundary conditions (BCs) are determined and the ODEs system is solved, the obtained field coefficients are transformed back to the spatial domain. A schematic overview of the various steps involved in the numerical algorithm is illustrated in Fig. 13.

A more detailed discussion of the lateral discretization, the boundary conditions and the vertical discretization is presented in the following three items:

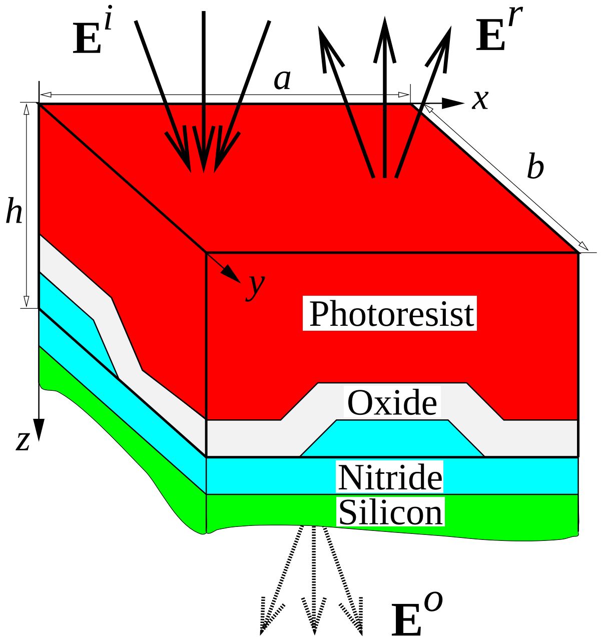


Figure 12: In the given example the simulation domain consists of the inhomogeneous photoresist, the nonplanar oxide and the nonplanar part of the nitride. Above ($z < 0$) and below ($z > h$) multiple planar homogeneous layers form a stratified medium.

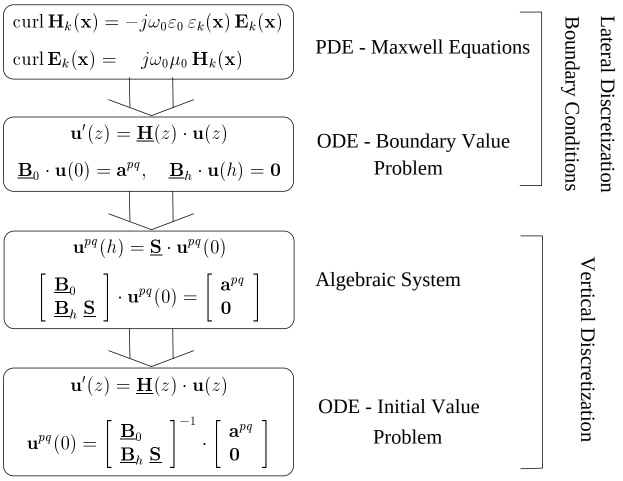

$\operatorname{curl} \mathbf{H}_k(\mathbf{x}) = -j\omega_0\varepsilon_0\,\varepsilon_k(\mathbf{x})\,\mathbf{E}_k(\mathbf{x})$
$\operatorname{curl} \mathbf{E}_k(\mathbf{x}) = j\omega_0\mu_0\,\mathbf{H}_k(\mathbf{x})$
PDE - Maxwell Equations
$\mathbf{u}'(z) = \underline{\mathbf{H}}(z) \cdot \mathbf{u}(z)$
$\underline{\mathbf{B}}_0 \cdot \mathbf{u}(0) = \mathbf{a}^{pq}, \quad \underline{\mathbf{B}}_h \cdot \mathbf{u}(h) = \mathbf{0}$
ODE - Boundary Value Problem
Boundary Conditions
Lateral Discretization
$\mathbf{u}^{pq}(h) = \underline{\mathbf{S}} \cdot \mathbf{u}^{pq}(0)$
$\begin{bmatrix} \underline{\mathbf{B}}_0 \\ \underline{\mathbf{B}}_h\,\underline{\mathbf{S}} \end{bmatrix} \cdot \mathbf{u}^{pq}(0) = \begin{bmatrix} \mathbf{a}^{pq} \\ \mathbf{0} \end{bmatrix}$
Algebraic System
$\mathbf{u}'(z) = \underline{\mathbf{H}}(z) \cdot \mathbf{u}(z)$
$\mathbf{u}^{pq}(0) = \begin{bmatrix} \underline{\mathbf{B}}_0 \\ \underline{\mathbf{B}}_h\,\underline{\mathbf{S}} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{a}^{pq} \\ \mathbf{0} \end{bmatrix}$
ODE - Initial Value Problem
Vertical Discretization

Figure 13: Overview of the numerical algorithm

- **Lateral Discretization:** Due to the periodic nature of the incident light (1) and the assumed lateral periodicity of the simulation domain the EM field as well as the permittivity inside the simulation domain can be expressed by Fourier series,

$$\varepsilon(\mathbf{x}) = \sum_{n,m} \varepsilon_{nm}(z) e^{j2\pi(nx/a+my/b)}, \qquad \mathbf{U}(\mathbf{x}) = \sum_{n,m} \mathbf{U}_{nm}(z) e^{j2\pi(nx/a+my/b)}, \tag{6}$$

whereby $\mathbf{U}$ stands in for the two EM field vectors $\mathbf{E}$ and $\mathbf{H}$ respectively. Here, it is most important to emphasize that the above expansions are valid for all point sources Q_{pq} and time steps t_k.

Insertion of (6) into (4) transforms the PDEs into an infinite dimensional set of coupled ODEs for the Fourier coefficients of the lateral field components, as the vertical field components can be expressed analytically by the lateral ones. Next, the Fourier expansions of

(6) are truncated. Thus, only coefficients $\{E_{x,nm}, E_{y,nm}, H_{x,nm}, H_{y,nm}\}_{|n|\leq N_x,|m|\leq N_y}$ symmetrically centered around the vertical incident ray $n = m = 0$ are related by the final ODE-system,

$$\frac{d}{dz}\mathbf{u}(z) = \underline{\mathbf{H}}(z) \cdot \mathbf{u}(z) \qquad \text{with} \qquad \underline{\mathbf{H}}(z) = \begin{bmatrix} \underline{\mathbf{0}} & \underline{\mathbf{R}}(z) \\ \underline{\mathbf{G}}(z) & \underline{\mathbf{0}} \end{bmatrix}. \tag{7}$$

In the above matrix-vector notation the complex-valued vector $\mathbf{u}(z) = \Big[\mathbf{e}_x(z), \mathbf{e}_y(z), \mathbf{h}_x(z), \mathbf{h}_y(z)\Big]^{\mathrm{T}}$ comprises the Fourier coefficients of the lateral field components, e.g., $(\mathbf{e}_x(z))_k = E_{x,n(k)m(k)}(z)$, and the elements $\underline{\mathbf{R}}(z)$ and $\underline{\mathbf{G}}(z)$ of the system matrix $\underline{\mathbf{H}}(z)$ contain the Fourier coefficients of the permittivity $\varepsilon(\mathbf{x})$. Each of the $\mathbf{e}$ and $\mathbf{h}$ vectors has dimension $(2N_x + 1) \times (2N_y + 1)$ due to the symmetric truncation. Therefore the entire ODE system is of dimension

$$N_{\mathrm{ODE}} = 4 \times (2N_x + 1) \times (2N_y + 1) \approx 16 \times N_x \times N_y. \tag{8}$$

- **Boundary Conditions:** Above and below the simulation domain we have homogeneous planar layers (cf. Fig. 12). Inside one layer the EM field can be expressed by a plane wave or Rayleigh expansion [12]. The mathematical formulation is that of a Fourier expansion with vertically known dependence of the coefficients. Above the simulation domain ($z < 0$) we have to consider incident and reflected waves,

$$\mathbf{U}(\mathbf{x}) = \sum_{|n|\leq N_x} \sum_{|m|\leq N_y} \left[\mathbf{U}^i_{nm} e^{jk^0_{z,nm}z} + \mathbf{U}^r_{nm} e^{-jk^0_{z,nm}z}\right] e^{j2\pi(nx/a+my/b)}, \tag{9}$$

below ($z > h$) only outgoing waves occur disregarding multiple planar layers,

$$\mathbf{U}(\mathbf{x}) = \sum_{|n|\leq N_x} \sum_{|m|\leq N_y} \mathbf{U}^o_{nm} e^{jk^s_{z,nm}(z-h)} e^{j2\pi(nx/a+my/b)}. \tag{10}$$

Matching the two Rayleigh expansions (9) and (10) with the field representation (6), valid inside the simulation domain, and eliminating the unknown reflected and outgoing wave amplitudes, $\mathbf{U}^r_{nm}$ and $\mathbf{U}^o_{nm}$, respectively, yields exactly half of the BCs at the top ($z = 0$) and at the bottom ($z = h$) of the simulation domain. The incident amplitudes $\mathbf{U}^i_{nm}$ are of course involved in the BCs and excite the EM field inside the simulation domain. They are the output of the illumination simulation and given by (2). This means that we have different BCs for each coherent point source Q_{pq}. Using the above introduced vector notation we find

$$\underline{\mathbf{B}}_0 \cdot \mathbf{u}(0) = \mathbf{a}^{pq}, \qquad \underline{\mathbf{B}}_h \cdot \mathbf{u}(h) = \mathbf{0}. \tag{11}$$

The two rectangular matrices $\underline{\mathbf{B}}_0$ and $\underline{\mathbf{B}}_h$ are independent of the specific Q_{pq}, whereas the matrix-vector $\mathbf{a}^{pq}$ comprises the incoming wave amplitudes $\mathbf{E}^{pq}_{nm}$ and $\mathbf{H}^{pq}_{nm}$ of one coherent point source contribution. This means, that we have transformed the Maxwell equations (4) into a linear complex-valued two-point boundary value problem (7) and (11) with multiple BCs.

- **Vertical Discretization:** We adapted the memory saving “shooting method” [13] as it allows a very efficient treatment of the multiple right hand sides of the first BC in (11). The algorithm is based on the observation, that the system matrix $\underline{\mathbf{H}}(z)$ in (7) as well as the two boundary matrices $\underline{\mathbf{B}}_0$ and $\underline{\mathbf{B}}_h$ of (11) are independent of the chosen point source. Exploiting this situation, we first establish a relation between the two boundary points $z = 0$ and $z = h$. This is accomplished by applying an explicit discretization scheme to (7).

The obtained recursion formula $\mathbf{u}(z_{j+1}) = \underline{\mathbf{S}}_j \cdot \mathbf{u}(z_j)$ between two adjacent mesh points z_j and $z_{j+1} = z_j + h_j$ is then successively evaluated,

$$\mathbf{u}(h) = \left[\prod_{j=0}^{N_z-1} \underline{\mathbf{S}}_j \right] \cdot \mathbf{u}(0) = \underline{\mathbf{S}} \cdot \mathbf{u}(0), \tag{12}$$

whereby N_z is the number of vertical discretization points. Combining this equation with the second BC of (11) yields $\underline{\mathbf{B}}_h \cdot \mathbf{u}(h) = \underline{\mathbf{B}}_h \, \underline{\mathbf{S}} \cdot \mathbf{u}(0) = \mathbf{0}$, which forms together with the first BC of (11) a linear algebraic system for the initial values $\mathbf{u}^{pq}(0)$ due to one excitation vector $\mathbf{a}^{pq}$,

$$\left[\begin{array}{c} \underline{\mathbf{B}}_0 \\ \underline{\mathbf{B}}_h \, \underline{\mathbf{S}} \end{array} \right] \cdot \mathbf{u}^{pq}(0) = \left[\begin{array}{c} \mathbf{a}^{pq} \\ \mathbf{0} \end{array} \right]. \tag{13}$$

This linear system is solved by performing a LU decomposition, which is an extremely efficient solution method for linear systems with multiple right hand sides [14]. Once the initial values $\mathbf{u}^{pq}(0)$ are found, the solution vector $\mathbf{u}^{pq}(z_j)$ inside the simulation domain is calculated by integrating the ODE system (7). As the elements of $\mathbf{u}^{pq}(z_j)$ correspond to the Fourier coefficients of the EM field, they have to be transformed back to the spatial domain. Finally, the point source contributions are incoherently superposed to build up the absorbed light intensity within the photoresist needed in (5) for the exposure/bleaching model.

The proposed algorithm has the big advantage, that the vertical mesh size N_z does not influence the storage consumption as the recursion matrices $\underline{\mathbf{S}}_j$ in (12) do not have to be stored individually. The memory usage is therefore only determined by the rank N_{ODE} of the ODE system (8) and is of order $\mathcal{O}(N_{\mathrm{ODE}}^2) \approx 256 \times \mathcal{O}(N_x^2 \times N_y^2)$. Typically 30 Fourier coefficients are needed for each lateral direction. In this case $N_x = N_y = 15$ and the ODE system is of rank $N_{\mathrm{ODE}} = 3844$. Assuming 16 Bytes for a double precision complex number, approximately 250 MB memory are required to store the system matrix. For three-dimensional rigorous photolithography simulation this storage consumption is in accordance with other frequency-domain methods (e.g., [5]), and lies dramatically below time-domain methods (e.g., [4]).

For the investigation of the numerical costs we have to bear in mind, that the Maxwell equations (4) have to be solved for multiple time steps (cf. Fig. 11). The numerical costs for one time step are mainly determined by the evaluation of the recursion (12) and by the solution of (13). Both operations are of order $\mathcal{O}(N_{\mathrm{ODE}}^3)$. Hence, the total run time grows for N_t time steps and N_z vertical discretization points with $\mathcal{O}(N_t \times N_z \times N_{\mathrm{ODE}}^3)$ and is typically a few hours on DEC-6000/400 workstation.

3.4 Development Simulation

The development of the photoresist is modeled as a surface-controlled etching reaction [8]. We use Kim's 'R'-model to relate the bulk image to a spatially inhomogeneous etch or development rate [15]. This development rate is stored on a tensorproduct grid, because the above discussed differential method requires a laterally uniform spaced grid to apply the numerically highly efficient two-dimensional Fast Fourier Transform (FFT) algorithm. For the simulation of the time-evolution of the development front we have adapted the cellular-based topography simulator [16] of the VISTA framework to read the development rate from the tensorproduct grid. The basic idea behind this surface advancement algorithm is to apply a structuring element along the exposed surface which removes successively photoresist cells of the underlying cellular geometry representation. Within the scope of lithography simulation the shape of the structuring element depends on the precalculated development rate multiplied by the chosen time step.

As the development rate exhibits a strong dependence on the spatial coordinates, e.g., due to standing waves or notching effects during photoresist exposure, a sufficiently high number of cells has to be chosen to resolve these variations. For example, in case of standing waves we know that the distance between the maxima and minima of the absorbed light intensity and therefore also of the development rate is $\lambda/4$ yielding approximately $50\,nm$ for I-line illumination ($\lambda = 365\,nm$) and a refractive index of 1.8 for the photoresist. For an accurate movement of the development front this significant distance should be resolved by 15 cells [16]. Hence a cell density of 300 cells/μm is needed. The applicability of the advancement algorithm for this cellular geometry resolution has been demonstrated in [16].

3.5 Simulation of Contact Hole Printing

To demonstrate the capability of our approach we simulated contact hole printing over a planar and a stepped topography (cf. Fig. 14 and Fig. 15).

In both cases the simulation domain was $1.5\,\mu m \times 1.5\,\mu m \times 1.0\,\mu m$ large. For the imaging and exposure/bleaching simulation 31 Fourier modes or $N_x = N_y = 15$ were used to represent the EM field consuming 250 MB memory. The number N_z of vertical discretization points was 100 and 5 time steps were used for the bleaching reaction. The run time was about 6 hours on a DEC-6000/400 workstation.

The development simulation was performed with a cell density of 300 cells/μm. The memory usage was 60 MB assuming 1 Byte per cell and the run time was 30 minutes on a DEC-6000/400 workstation.

In Fig. 14 we show the aerial image obtained by the vector-valued approach discussed in Section 3.2. Conventional I-line illumination with a numerical aperture of $NA = 0.5$ and a partial coherence factor of $\sigma = 0.7$ was used. 9 mutually incoherent point sources were needed to account for the partial coherence. The point source location is shown in the wavevector diagram of Fig. 8.

In Fig. 15 a contour plot of the PAC is shown in the upper picture and the developed photoresist profile in the lower picture. The contours are given for PAC = $0.2, 0.3, \ldots, 1.0$. The exposure-dose was $120\,mJ/cm^2$ and the development time was $50\,sec$. The simulation parameters were for the Dill-model $n_0 = 1.65$, $A = 0.55\,\mu m^{-1}$, $B = 0.045\,\mu m^{-1}$, $C = 0.013\,cm^2/mJ$ and for the Kim-model $R_1 = 0.25\,\mu m/sec$, $R_2 = 0.0005\,\mu m/sec$, $R_3 = 7.4$ (cf. Table IV in [15]).

A comparison of the simulations exhibits a wider opening in the developed photoresist for the stepped topography than for the planar substrate. Hence, the effective diameter of the contact hole depends on the nonplanarity of the wafer topography.

3.6 Summary

An overall three-dimensional workstation based photolithography simulator has been developed, that accounts for all three fundamental subprocesses of mask imaging, resist exposure/bleaching

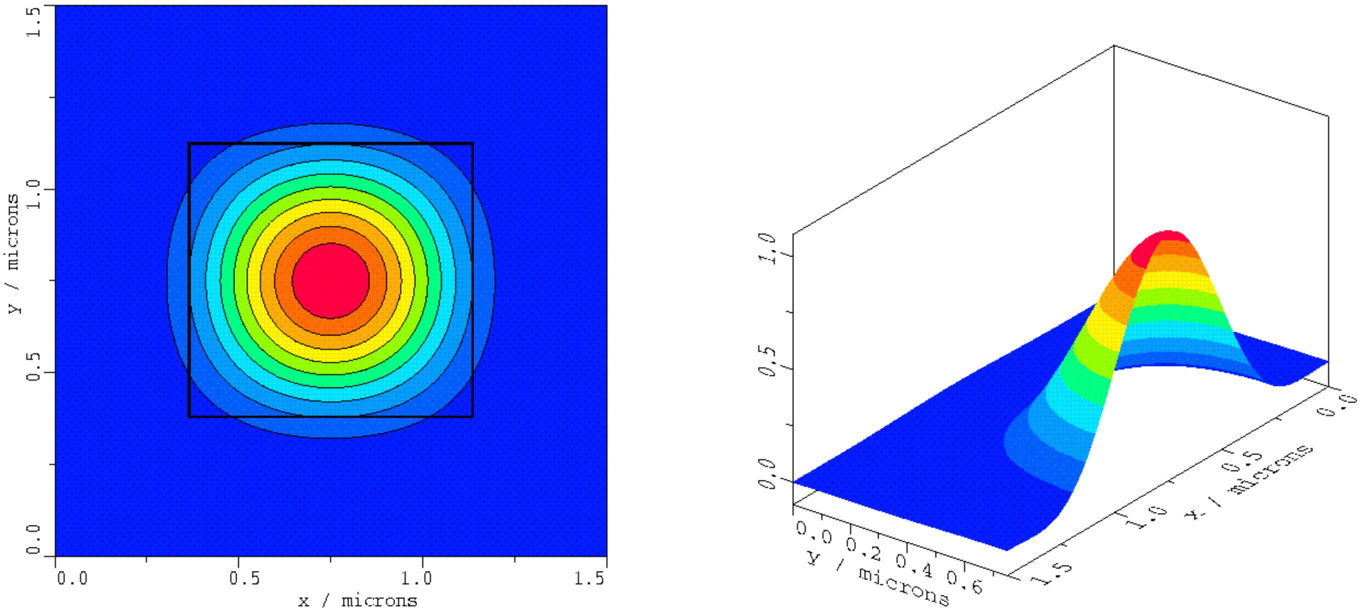


Figure 14: Aerial image of a $0.75\,\mu m \times 0.75\,\mu m$ wide contact hole centered in the middle of the the $1.5\,\mu m \times 1.5\,\mu m$ large simulation domain.

and resist development. The imaging simulation relies on a vector-valued reformulation of the classical scalar theory of Fourier optics. The exposure/bleaching simulation extends the two-dimensional differential method to the third dimension. This approach has been shown to be extremely efficient for the simulation of nonplanar scattering effects in combination with partial coherent illumination. For the development/etching step the cellular based topography simulator of the VISTA framework has been adapted to account for lithography specific requirements such as rapid varying inhomogeneous etch rates. The capability of the overall simulator was demonstrated by showing simulation results of contact hole printing over a planar and nonplanar substrate.

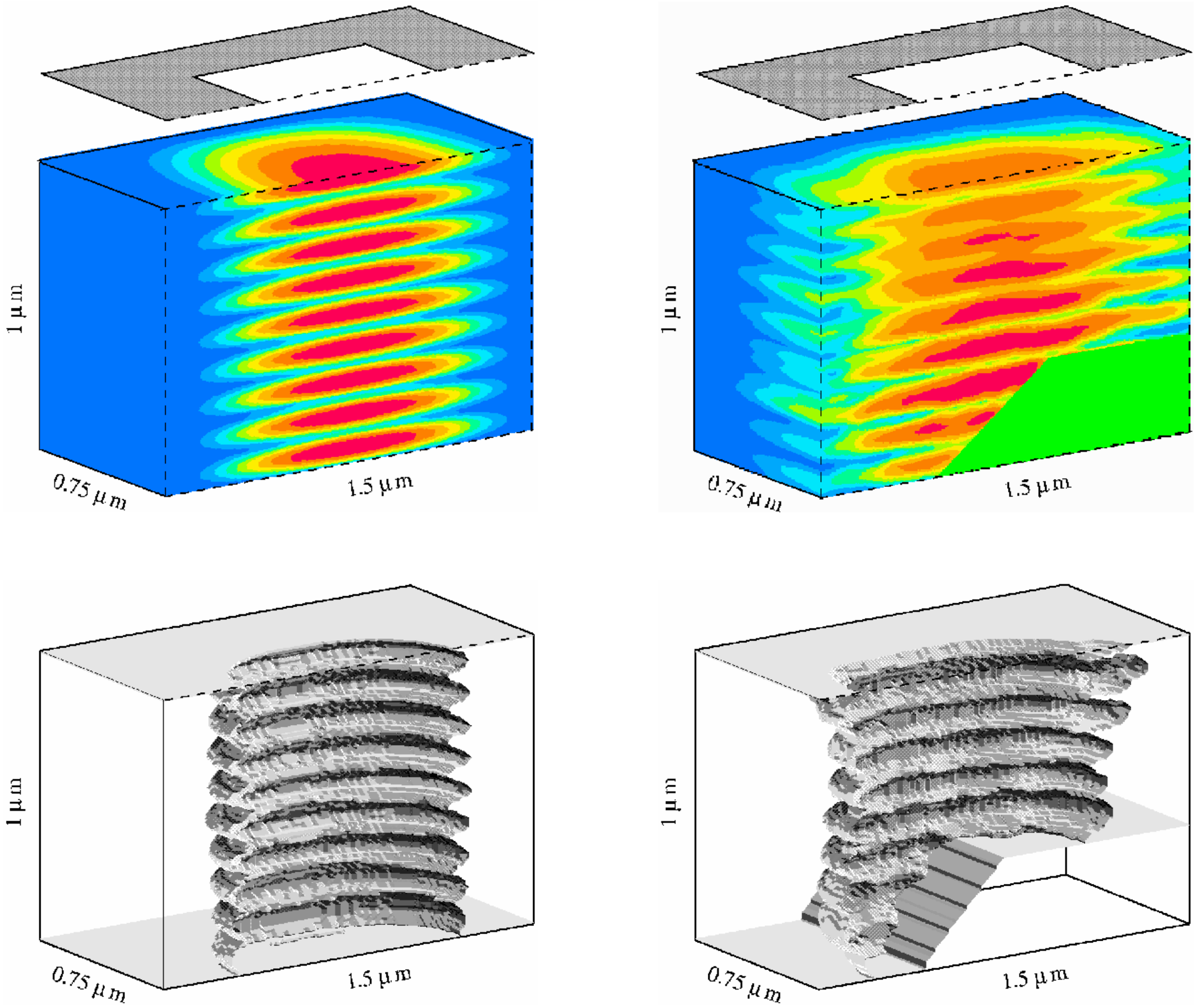


Figure 15: Bulk image and developed photoresist profile over a planar and a non-planar substrate.

4 Analytical PDE Modeling in VISTA

4.1 Motivation

During the last years it has become increasingly difficult to meet all requirements in semiconductor simulation since a lot of new and more complex physical models were developed. However, the transfer of this evolution to computer science turned out to be very time consuming and resulted in a profusion of different simulators which can handle more or less a special but very restricted field of physical behavior, especially, in three dimensions.

Instead of developing a new specialized module for semiconductor simulation, our aim was to fulfill all requirements on a general simulation process which can handle any partial differential equation system in time and space. Based on the knowledge of the recent years a complete set of common features of all simulators have been worked out with the special considerations to not restrict the numerical solution methods. As a result of these efforts have developed the Analytical Modeling Interface and Object-oriented Solver (AMIGOS), which transforms any analytical spatial and time dependent differential equation system into a discretized one.

4.2 What is AMIGOS

AMIGOS is a tool for developing any kind of physical model which can be locally discretized. It contains a mathematical interpreter in the manner of programs like Mathematica, Mathcad, Matlab, etc., but it is tailormade for optimized matrix operations as well as for other mathematical expressions (e.g., +, -, *, /, sin, sqrt, etc.) and for operations which are necessary to solve numerical problems. Furthermore, it is provided with a special analytical optimizer to reduce the number of mathematical operations within the numerical model to minimize evaluation time during simulation.

AMIGOS can be used in two ways:

1. In a single-pass mode especially to develop new physical models (developer mode):
The analytical user input will be interpreted, optimized, transformed and solved on any complex simulation domain at once without necessity of time consuming recompilations.

2. In a two-pass mode for optimized use to solve large problems (user mode):
After having found a well fitting model for a physical problem using the developer mode C-Code can be generated automatically from the analytical user input, which is then compiled and linked to a model library and is now available for future use in a highly optimized manner.

The advantages of AMIGOS as against conventional simulation modules are: -10pt

- high flexibility
- problem independent
- support all standard discretization methods (e.g., finite elements [17][18], finite boxes and finite differences [19])

- no low-level programming
- short break-in period
- short model developing time
- simple interfacing to model library

AMIGOS is composed of two major parts (cf. Fig. 16):

1. The Analytical Modeling Interface (AMI) transforms the necessary mathematical expressions from a general analytical form into an internal optimized numerical model.
2. The General Object-oriented Solver (GOS) includes all necessary functions to solve a numerical problem on a given simulation domain (e.g., Newton iterator, time-step control, etc.) as well as complete grid management in one, two and three dimensions.

4.3 The Analytical Modeling Interface (AMI)

4.3.1 The Analytical Model Input Language

To simplify the way of model development AMIGOS is equipped with a mathematical interpreter. Using a simple syntax virtually any model can be developed. The Following example shows how a model can be defined using the Analytical Model Input Language:

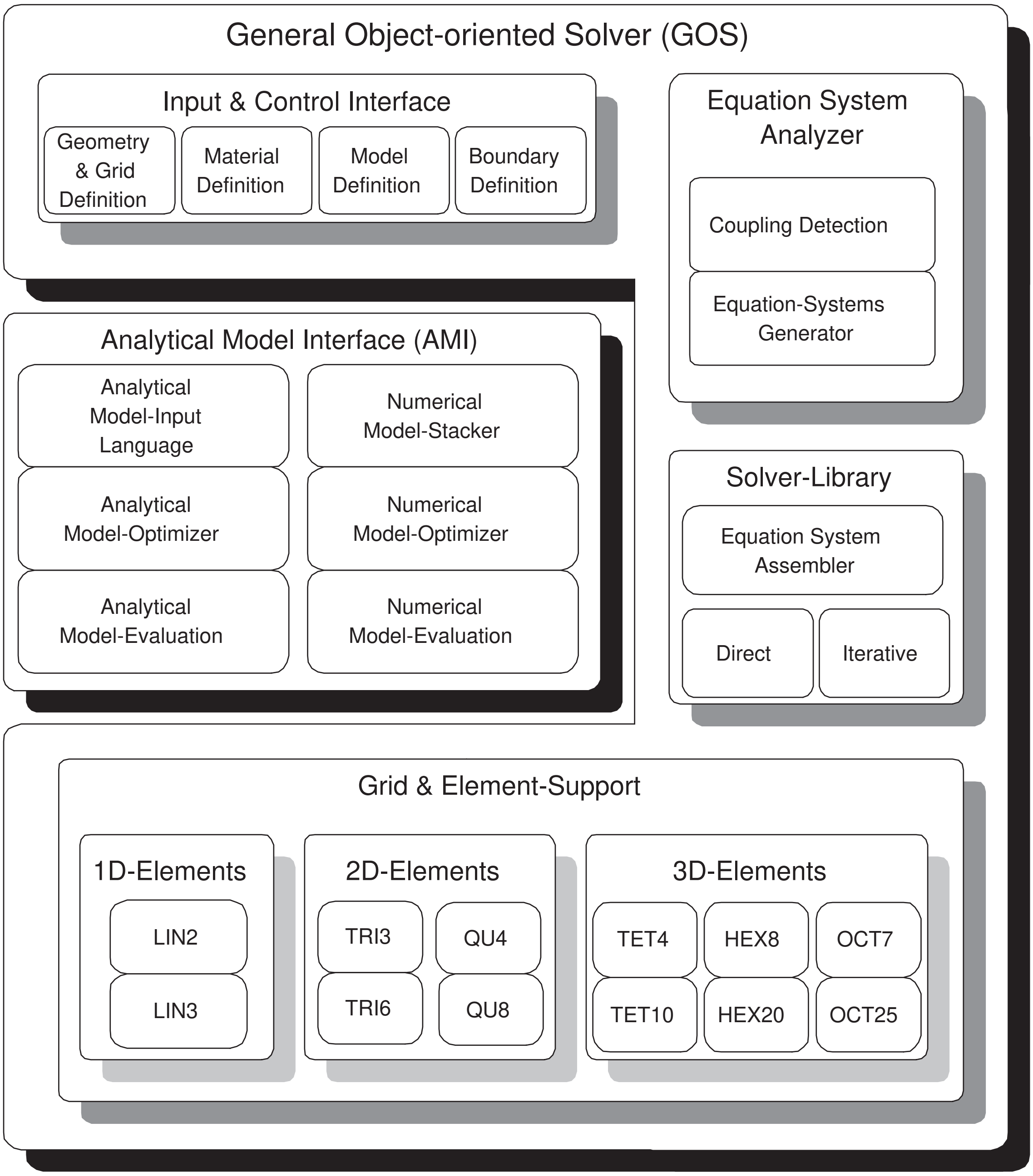

General Object-oriented Solver (GOS)
Input & Control Interface
Geometry & Grid Definition
Material Definition
Model Definition
Boundary Definition
Equation System Analyzer
Coupling Detection
Equation-Systems Generator
Analytical Model Interface (AMI)
Analytical Model-Input Language
Numerical Model-Stacker
Analytical Model-Optimizer
Numerical Model-Optimizer
Analytical Model-Evaluation
Numerical Model-Evaluation
Solver-Library
Equation System Assembler
Direct
Iterative
Grid & Element-Support
1D-Elements
LIN2
LIN3
2D-Elements
TRI3
QU4
TRI6
QU8
3D-Elements
TET4
HEX8
OCT7
TET10
HEX20
OCT25

Figure 16: Block structure of AMIGOS

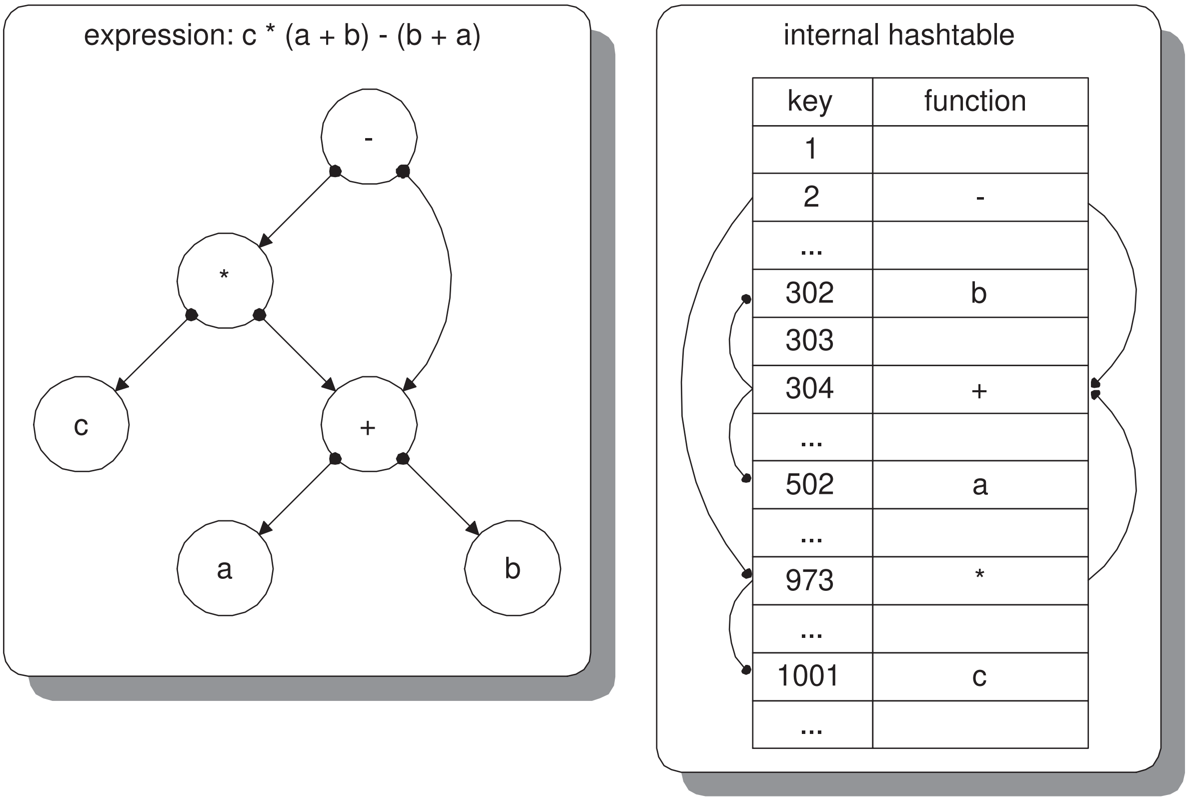
```
MODEL ModelName = [x1,x2,...,xn];
   # MODEL is a predefined keyword
   # ModelName:  the name your model should get
   # X = [x1,x2,x3,...]  the solution-vector
{
   # Begin Model Description
var1 = x1+5*x2;   # any user defined mathematical
var2 = var1+2*x3; # expression(functions,matrices,...)
var3(var1,var2) = var1 - var2;
...
i = 1..n;         # running index-variable from 1 to n
func[i] = 2*X[i]; # just defined for example

   # Definition of residual and its derivative to
   # get the numerical form:
   #       [jacobian] * [X] = [residual]

residual = [[f1(x1,x2,...,xn)]
            [f2(x1,x2,...,xn)]
            [        ...      ]
            [fn(x1,x2,...,xn)]];

jacobian = [[df1/dx1] [df1/dx2] ...  [df1/dxn]
            [df2/dx1] [df2/dx2] ...  [df2/dxn]
                                ...
            [dfn/dx1] [dfn/dx2] ...  [dfn/dxn]];
}
```

4.3.2 The Analytical Model Optimizer

To minimize the evaluation time of a model a special optimizer has been developed, that tries to reduce the number of mathematical operations to a minimum. Especially, in use with matrix operations it reduces the number of operations by an average of about two third, since each partial solution of an expression will be stored for later reuse so every mathematical term is guaranteed to be calculated just once. The simple example in Fig. 17 shows the general functionality of the optimizer, where the number of operations decreases from four to three.

4.3.3 The Analytical Model Evaluator

The Analytical Model Evaluator is used for evaluating simple mathematical expressions by the optimizer on the one hand, and, on the other hand, it offers a possibility for verifying a model, by presetting all unknown variables with any number so that a numerical solution for the residual vector and the jacobian matrix respectively can be printed and be checked for correctness.



key	function
1	
2	-
...	
302	b
303	
304	+
...	
502	a
...	
973	*
...	
1001	c
...	

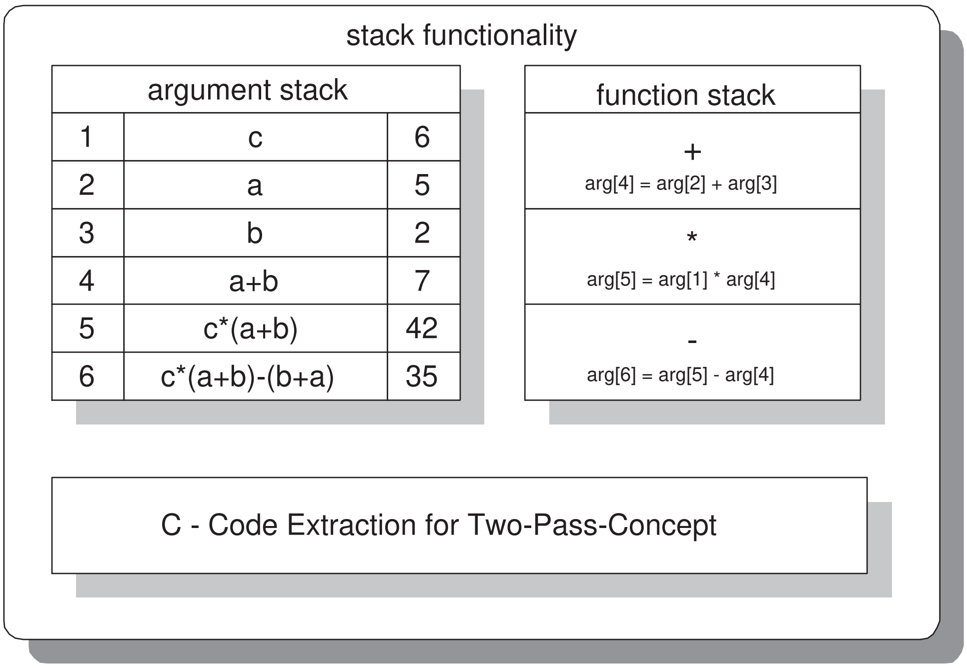
Figure 17: Internal structure of expression handling and optimization using a binary tree converted into an extendable hash table

4.3.4 The Numerical Model Stacker

Once the analytical part of AMI has finished it holds a complete description of the model. Now it is necessary to connect the abstract model quantities (defined as solution vector) to real quantities which are defined on a segment of the simulation domain. This is done by the Numerical Model Stacker that builds an internal array for functions (mathematical expressions) and their arguments. Furthermore, it detects the input and output variables (solution variables, auxiliaries and parameters defined in the analytical model) and prepares the stack so that the data transfer between solver and model evaluator can take place. The example shown in Fig. 18 gives an impression about the internal data management of the Numerical Model Stacker, using a=5, b=2 and c=6 from the previous example (Fig. 17) as input.

4.3.5 The Numerical Model Evaluator

After all previous steps we have got a complete physical model defined for a single finite element supported by the General Object-oriented Solver (GOS). As a result of a call to the Numerical Model Evaluator we get the calculated local residual and jacobian which can now be inserted into the global stiffness matrix of the system. The evaluator itself is only used for the one-pass mode since in the two-pass mode the evaluation of the model is already included in the produced C-code.



stack functionality

argument stack		
1	c	6
2	a	5
3	b	2
4	a+b	7
5	c*(a+b)	42
6	c*(a+b)-(b+a)	35

function stack
+ arg[4] = arg[2] + arg[3]
* arg[5] = arg[1] * arg[4]
- arg[6] = arg[5] - arg[4]

C - Code Extraction for Two-Pass-Concept

Figure 18: Internal data representation of the Numerical Model Stacker

4.4 General Object-oriented Solver (GOS)

As just described the AMI generates a numerical model, even though it has no information about the complete simulation procedure. Therefore it is embedded into the General Object-oriented Solver (Fig. 19) which uses AMI's evaluated output (residual and jacobian) for building the global stiffness matrix on the one hand, and, on the other hand for splitting the global matrix into smaller pieces detecting the couplings between quantities inside and between several segments of the simulation domain (Equation System Analyzer). Furthermore, it has a complete built-in grid adaptation [20] for all supported grid element types as well as a timestep control and newton iterator with several different modes to select.

To simplify the mapping of all abstract variables, outputs and inputs to the real simulation domain a simple grid and boundary description language has been developed, which can be used to define all necessary boundary and volume instructions as well as for choosing among different simulation modes.

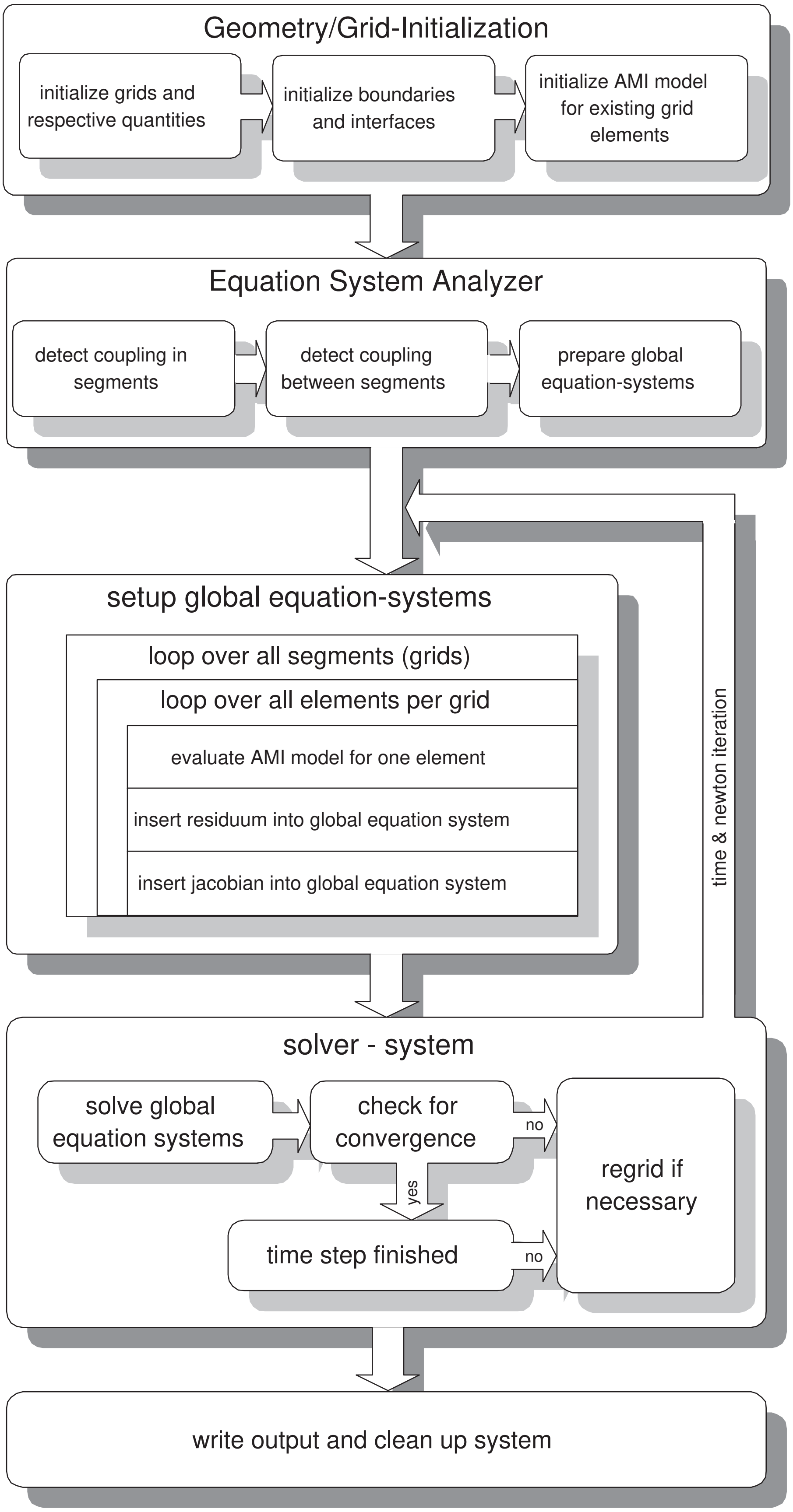

Geometry/Grid-Initialization
initialize grids and respective quantities
initialize boundaries and interfaces
initialize AMI model for existing grid elements
Equation System Analyzer
detect coupling in segments
detect coupling between segments
prepare global equation-systems
setup global equation-systems
loop over all segments (grids)
loop over all elements per grid
evaluate AMI model for one element
insert residuum into global equation system
insert jacobian into global equation system
time & newton iteration
solver - system
solve global equation systems
check for convergence
no
yes
regrid if necessary
time step finished
no
write output and clean up system

Figure 19: Block structure of the General Object-oriented Solver

4.5 Application

4.5.1 Coupled Diffusion of two Charged Dopants

Fig. 20 and 21 show the coupled diffusion of arsenic and boron. The simulation was started from an unsymmetrical gaussian distribution of arsenic (Fig. 20) with a peak concentration of $10^{20}\mathrm{cm}^{-3}$ and a constant boron background doping with $10^{15}\mathrm{cm}^{-3}$. As a well established test case the following coupled diffusion equations were implemented with AMIGOS [21]:

$$\frac{\partial C}{\partial t} = div\left[D\left(C\right)\ \left(grad\left(C\right) + Z\,\frac{C}{U_T}\,grad\left(\psi\right)\right)\right]$$

with

$$grad\left(\psi\right) = \frac{U_T}{\sqrt{4n_i^2 + C_{net}^2}}\ \sum Z_j\ grad\left(C_j\right)$$

$$C_{net} = \sum\ Z_j\ C_j$$

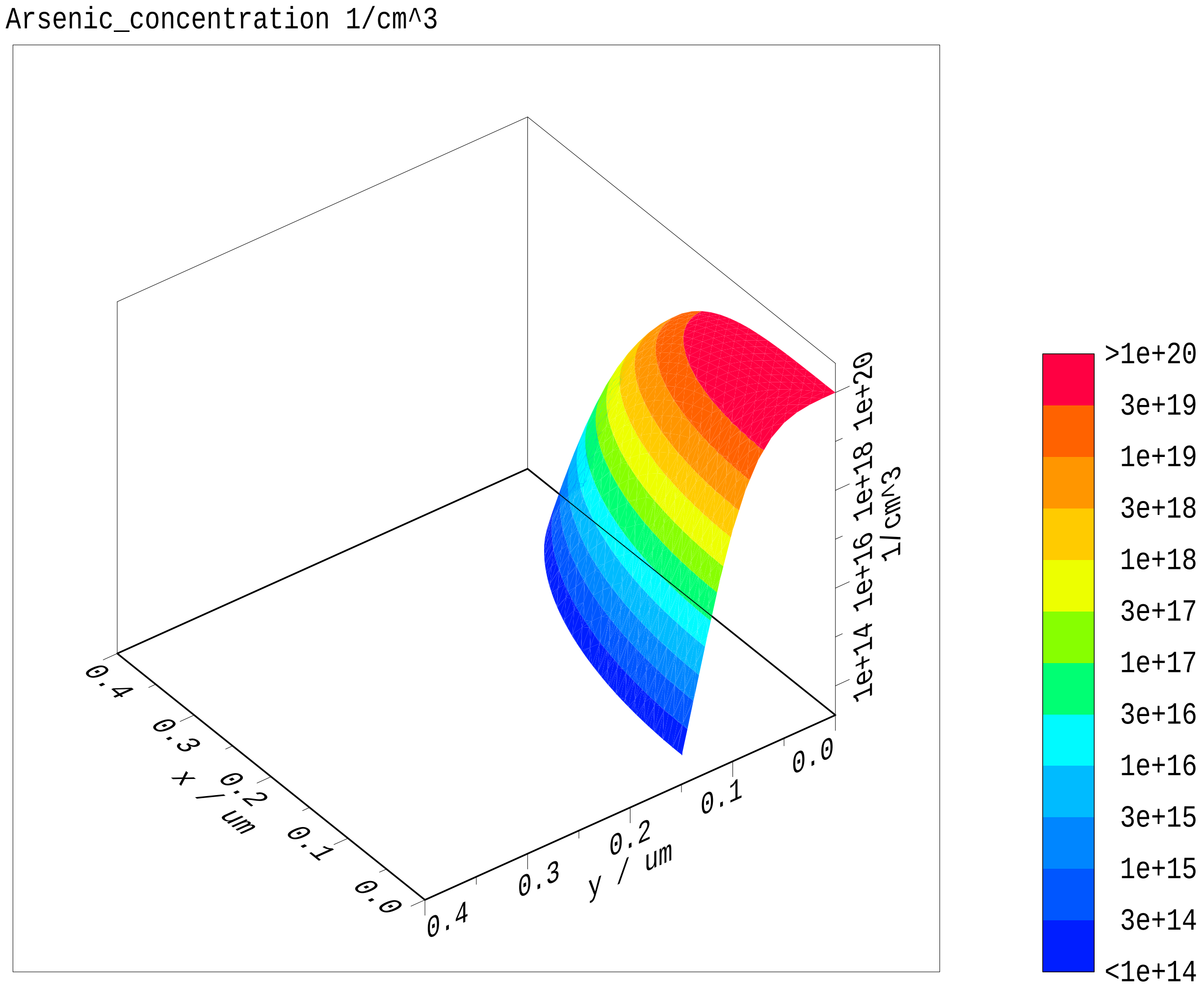


Figure 20: Arsenic concentration before the diffusion process

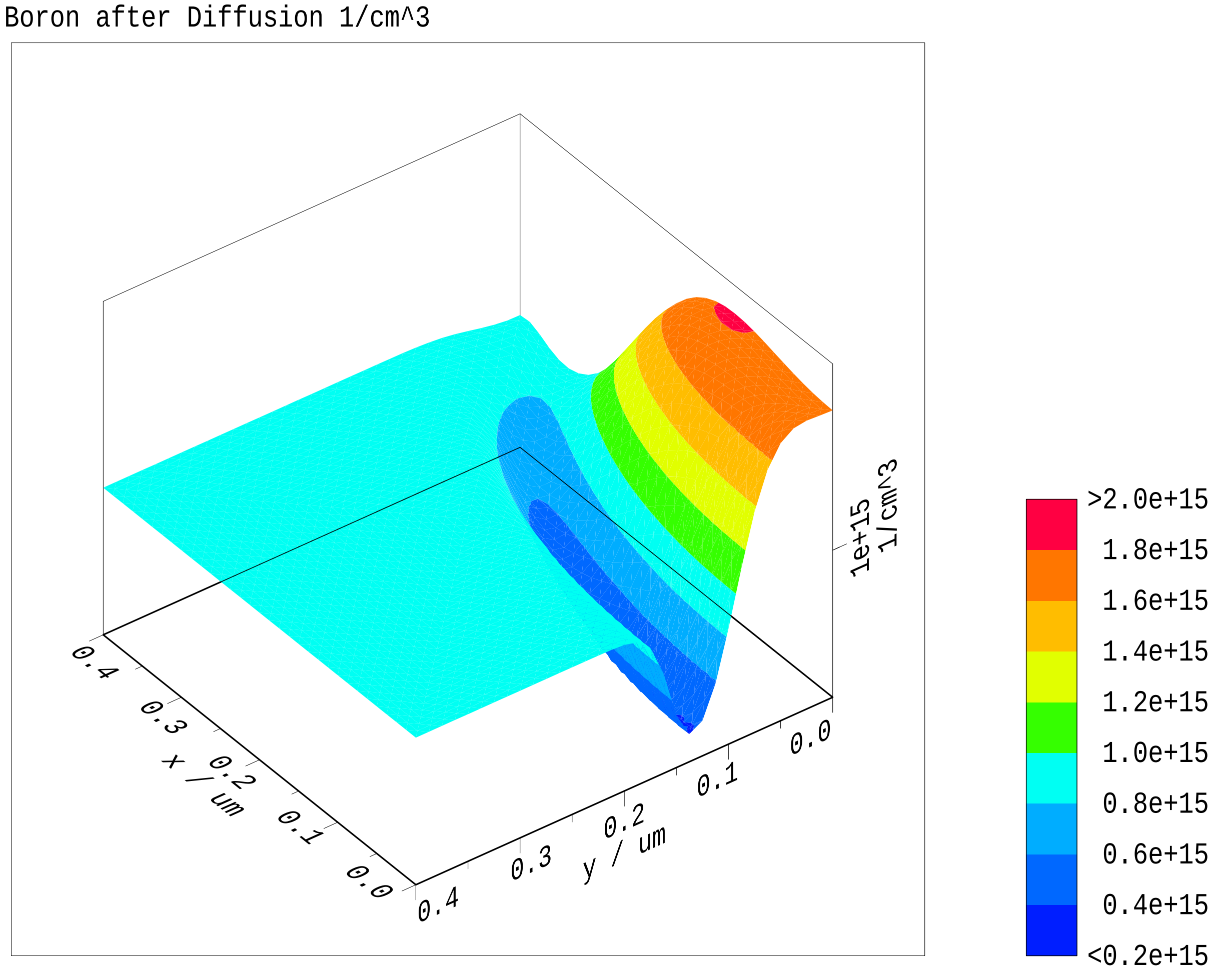


Figure 21: Boron concentration after the diffusion process

References

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

[2] G.E.P. Box and N.R. Draper. *Empirical Model-Building and Response Surfaces.* Wiley, 1987.

[3] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1995.

[4] A.K. Wong and A.R. Neureuther. Rigorous Three-Dimensional Time-Domain Finite-Difference Electromagnetic Simulation for Photolithographic Applications. *IEEE Trans.Semiconductor Manufacturing*, 8(4):419–431, 1995.

[5] K.D. Lucas, H. Tanabe, C.M. Yuan, and A.J. Strojwas. Efficient and Rigorous 3D Model for Optical Lithography Simulation. In H. Ryssel and P. Pichler, editors, *Simulation of Semiconductor Devices and Processes*, volume 6, pp 14–17, Wien, Austria, 1995. Springer.

[6] M.S.C. Yeung. Modeling High Numerical Aperture Optical Lithography. In *Proc.SPIE: Optical/Laser Microlithography*, volume 922, pp 149–167, 1988.

[7] J.W. Goodman. *Introduction to Fourier Optics.* McGraw-Hill, N.Y., 1968.

[8] F.H. Dill. Optical Lithography. *IEEE Trans.Electron Devices*, ED-22(7):440–444, 1975.

[9] H. Kirchauer and S.Selberherr. Rigorous Three-Dimensional Photolitography Simulation Over Nonplanar Structures. In G. Baccarani and M. Rudan, editors, *ESSDERC'96 - 26th European Solid State Device Research Conference*, pp 347–350, Gif-sur-Yvette Cedex, France, 1996. Editions Frontieres.

[10] H. Kirchauer and S. Selberherr. Three-Dimensional Photoresist Exposure and Development Simulation. In *SISPAD'96 - 1996 International Conference on Simulation of Semiconductor Processes and Devices*, pp 99–100, Tokyo, Japan, 1996.

[11] R. Petit. *Electromagnetic Theory of Gratings.* Springer, 1980.

[12] M. Born and E. Wolf. *Principles of Optics.* Pergamon Press, 6th edition, 1993.

[13] U.M. Ascher, R.M.M. Mattheij, and R.D. Russel. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Classics in Applied Mathematics. SIAM, 1995.

[14] G.H. Golub and C.F. Van Loan. *Matrix Computations.* John Hopkins University Press, second edition, 1989.

[15] D.J. Kim, W.G. Oldham, and A.R. Neureuther. Development of Positive Photoresist. *IEEE Trans.Electron Devices*, ED-31(12):1730–1736, 1984.

[16] E. Strasser and S. Selberherr. Algorithms and Models for Cellular Based Topography Simulation. *IEEE Trans.Computer-Aided Design*, 14(9):1104–1114, 1995.

[17] O.C. Zienkiewicz and R.L. Taylor. *Solid and Fluid Mechanics, Dynamics and Non-linearity*, volume 2 of *The Finite Element Method.* McGraw-Hill, fourth edition, 1991.

[18] H.R. Schwarz. *Methode der finiten Elemente.* Teubner, 1991.

[19] A.R. Mitchell and D.F. Griffiths. *The Finite Difference Method in Partial Differential Equations.* Wiley, 1980.

[20] E. Leitner and S. Selberherr. Three-Dimensional Grid Adaptation Using a Mixed-Element Decomposition Method. In H. Ryssel and P. Pichler, editors, *Simulation of Semiconductor Devices and Processes*, volume 6, pp 464–467, Wien, 1995. Springer.

[21] H. Puchner. *Advanced Process Modeling for VLSI Technology.* Dissertation, Technische Universität Wien, 1996.

[22] R. Bauer. *Numerische Berechnung von Kapazitäten in dreidimensionalen Verdrahtungsstrukturen.* Dissertation, Technische Universität Wien, 1994.