



VISTA Status Report June 1998

K. Dragosits, T. Grasser, H. Kosina, R. Mlekus, W. Pyka, S. Selberherr



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

Contents

1	Object-Oriented Management of Algorithms and Models	1
1.1	Introduction	1
1.2	Basic Structure	1
1.3	Model Definition Language	3
1.4	Examples	4
2	Integration of Layout, Topography Simulation, and R-C Extraction	7
2.1	Introduction	7
2.2	Isotropic and Anisotropic Etching and Deposition	8
2.2.1	The Surface Movement Algorithm of etch3d	8
2.2.2	Acceleration of Isotropic and Anisotropic Etching and Deposition	8
2.3	Inclusion of Layout Information in Topography Simulation	9
2.4	Resistance/Capacitance Extraction	11
3	Hydrodynamic Mixed-Mode Simulation	12
3.1	Introduction	12
3.2	Segments	12
3.3	Model Assembly	12
3.4	Quantity Classes	13
3.5	Programmable Iteration Scheme	13
3.6	Matrix Assembly	13
3.7	Example	14
4	2D Simulation of Ferroelectric Materials with MINIMOS-NT	16
4.1	Introduction	16
4.2	The Simulation Model	16
4.3	Simulation of the FEMFET	17

5	Simulation of Single-Electron Devices and Circuits	19
5.1	Introduction	19
5.2	SIMON	19
5.3	The T-Memory Cell	21

1 Object-Oriented Management of Algorithms and Models

1.1 Introduction

The increasing complexity of simulation tools combined with the requirement for short development cycles for the implementation of new models and algorithms raises a strong demand for object-oriented development tools and languages supporting the separation of the simulator into modules which can be maintained without interfering other modules.

For that reason a new library based concept was developed, which provides an object-oriented approach to the implementation, parameterization and selection of models¹, without any changes to the source code of the simulator.

The ALGORITHM LIBRARY is designed to support any kind of algorithm using arbitrary user defined data structures as parameters, which are handled in their native C++ representation and are forwarded to the models using references. It offers a set of C++ classes and methods to handle these algorithms and parameters directly in C or C++ code and the object-oriented MODEL DEFINITION LANGUAGE (MDL). The MDL can be used as an interpreted language (using a “just in time” byte code compiler) to ease the development of new algorithms, or – by using a two pass concept – as a compiler language to optimize the speed of simulations. Therefore algorithms and data structures used in the innermost simulation loops can be handled using the mechanisms of this library with almost no performance loss compared in relation to traditional implementations based on function calls.

Sets of models, appropriate parameter types, their operators and functions are packaged into dynamic link libraries which can be loaded during run time to be extended by additional models defined on the input deck by using MDL. Thus binary distributions of simulators can be compiled which are extendible by additional user defined algorithms and models for certain purposes.

These features distinguish the ALGORITHM LIBRARY from general purpose extension languages like TCL [1] or specialized approaches as presented in [2], [3] or [4], where modeling languages are introduced which are specialized to solving PDEs on specific mesh representations and the automatic generation of a Jacobian matrix.

1.2 Basic Structure

Algorithms and models defined with the ALGORITHM LIBRARY are represented by C++ classes derived from the base class `Model` or other previously defined model classes [5]. The thereby defined inheritance tree (Figure 1) is used to classify the various model algorithms and for checking the user supplied definitions on the input deck during the initialization of the ALGORITHM LIBRARY. `Model`-classes encapsulate the algorithm itself, private data values used to evaluate the algorithm, an interface containing the required input and output parameters, and the documentation (Figure 2).

The ALGORITHM LIBRARY provides an interface mechanism which separates arbitrary algorithms and/or model instances from the rest of the simulator. These interfaces contain the information about the type of algorithm to be used (requested model type), a specific instance name for the model and all the input and output parameters which are necessary to evaluate an algorithm of the requested type (“fat” interface concept). The actual algorithm used for a certain model instance can be selected on the input deck of the

¹In this text no conceptual distinction is made between the nouns “model” and “algorithm”.

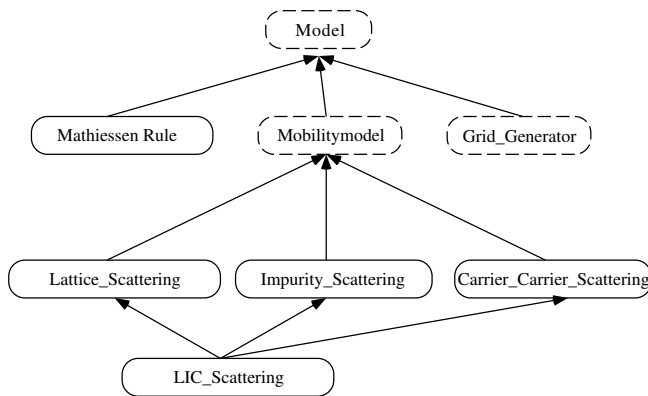


Figure 1: A sample model hierarchy

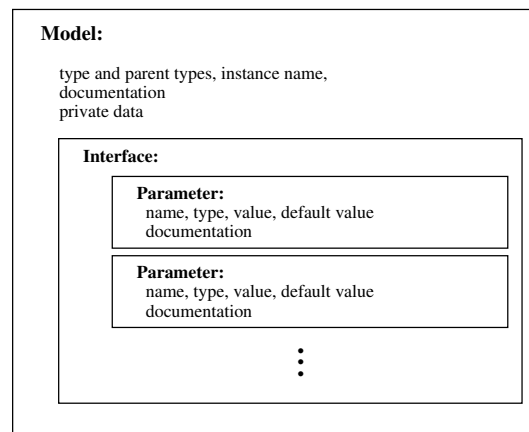


Figure 2: The data structure of a model

program, or by supplying a default type in the interface definition. During the initialization the ALGORITHM LIBRARY checks whether the model instances are either equal to or derived from the requested model type.

Parameter classes contain a reference to the value, a name which has to be unique inside of the given interface, and optionally documentation and default values (Figure 2). Several types of parameters according to the standard C++ variable types are predefined. New parameter types can be instantiated by specializing the template class `Parameter` with arbitrary C++ classes describing the values. For each of these parameters a set of operators and functions can be specified which can be used in calculations defined on the input deck as well as in algorithms defined in C++.

To evaluate the algorithms the parameter values are forwarded to the model instances by reference. Therefore the interfaces of the program and the model have to be linked by the ALGORITHM LIBRARY in the initialization phase of the program, so that the value references of affected parameters are set to equal values as shown in Figure 3. To support optimization of data structures as needed by advanced CPU architectures these references can explicitly be set to specified values.

Parameters with the same name and type are linked automatically; other links can be specified in the C++ code of the program and on the input deck using the MODEL DEFINITION LANGUAGE. A run time type check of the parameters ensures the software integrity of the input deck and the program. Since default values for parameters can be specified in the interface definition of a model, in the definition of the program interface, and on the input deck, the actual default value of linked parameters is determined by the source with the highest priority as depicted in Table 1.

priority	source of default value
3	input deck definition
2	interface definition
1	model definition

Table 1: DEFAULT VALUE PRIORITIES

Different sets of algorithms and appropriate parameter definitions can be collected in separate libraries of object code or MDL source files. Rapid prototyping of new algorithms is supported by an interpreter for the object-oriented MDL which is used to parse model definitions on the input deck of the program. Additional models can be implemented and tested during the run time of the program and added later to

the model libraries by using the MDL-compiler which translates the definitions on the input deck into C++ code.

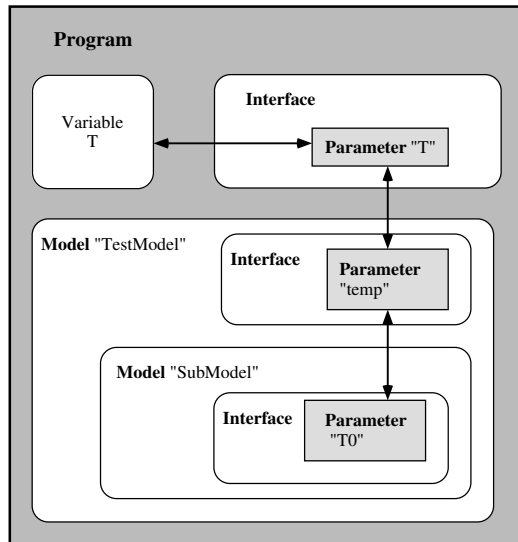


Figure 3: Linking of parameters

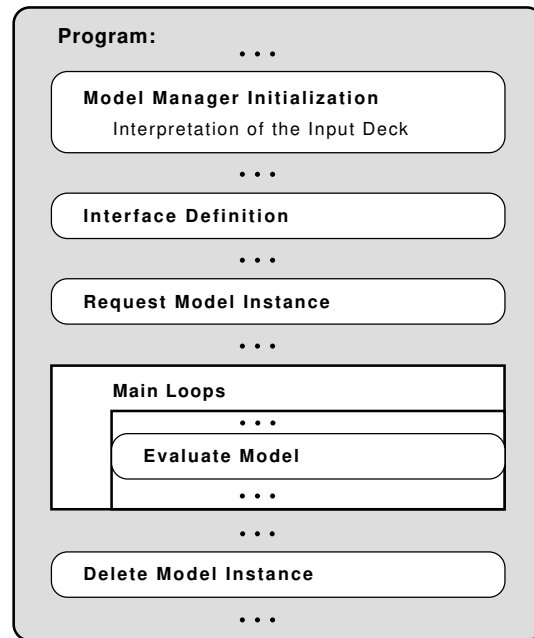


Figure 4: Structure of a program

An instance of a specific algorithm can be generated by forwarding the model type name to the ALGORITHM LIBRARY or by giving an instance name for the algorithm. In this case the actual class type is determined at run time by parsing the input deck. To evaluate the algorithm, its class instance is connected to an interface providing the necessary parameter values.

1.3 Model Definition Language

The ALGORITHM LIBRARY contains an interpreter and a compiler for the MODEL DEFINITION LANGUAGE which allows to:

- Define the actual algorithms (model instances) to be used for a specific task.
- Define the parameter values for model instances and default values for the parameters of certain types of algorithms.
- Define new MDL algorithms by inheriting and combining methods and interfaces from previously defined ones.
- Define global parameters which can be used to exchange parameter values between model instances where the author of the program didn't anticipate the necessity for such communication.
- Request a database record, describing all available algorithms, their interfaces and documentation, and the thereby defined model hierarchy.
- Request a debug report describing the actually used algorithms, the values and default values of parameters for specific model instances, and a table showing how these parameters are linked together.

MDL classes contain private and protected interface parameters, private and protected local parameters, and sub-models defined on previously scanned MDL source files or object libraries of compiled C++ code. The inheritance rules for protected and local parameters are similar to the C++ inheritance rules and support multiple inheritance of parameters and single inheritance of the evaluation rule.

The evaluation and initialization rules of MDL classes can contain calculations with parameters of any type. For the predefined C++ compatible parameter types the standard C++ operators are predefined with C++ compatible precedence rules. Operators for user defined parameter types can be used if they are supported by the classes describing the parameter values. These calculations can be combined with evaluations of sub-models by using conditional and loop expressions and evaluations of sub-models provided from the ALGORITHM LIBRARY.

A minimal program using the ALGORITHM LIBRARY to evaluate a single algorithm may be structured as shown in Figure 4:

1. The ALGORITHM LIBRARY is initialized by parsing and analyzing the input deck.
2. The “fat” interface containing all parameters a certain type of algorithm might need, the required model type, and a default model type is created. Optionally the documentation of the interface can be defined in this place, too.
3. The model instance is requested from the ALGORITHM LIBRARY and linked to the parameter interface.
4. Repeat as necessary: Compute the values of the input parameters; evaluate the model; use the resulting parameter values for further computations.
5. To release the acquired resources cleanly, the required model instance has to be deleted after the last evaluation. A shutdown function for the ALGORITHM LIBRARY resets the library into the initial state, so that new definitions can be parsed independently from any previous ones.

Steps 1–3 should take place during the initialization phase of the program because they require the rather time consuming parsing and interpretation of the input deck. Once the internal data structures of the ALGORITHM LIBRARY are assembled, the additional time consumption caused by the usage of the ALGORITHM LIBRARY are typically between 5–30 % depending on the complexity of the models.

1.4 Examples

In the process simulator PROMIS-NT [6] the ALGORITHM LIBRARY is used for the complete control of the simulation flow. Several interfaces and parameter and model types are implemented and prepackaged to be used in the MDL input deck files:

- All information needed to execute a specific simulation (e.g. input files, simulation time, ...) is given to the simulator by using an initialization algorithm which overwrites a number of predefined default values. (Figure 5)
- The process temperature progression can be specified as a function of the actual time and the start and end time of the simulation.

- For every impurity or stress distribution considered an initialization function depending on the values of all other distributions and the position values can be defined which is applied after reading the distributions from the input file.
- The coefficients α_{ij} , γ_i , a_{ij} , b_{ij} , c_{ij} , d_i , β_{ij} and Φ_i of the transport equations 1, 2 and 3 can be specified as functions of process temperature, time and position and all but β_{ij} can be functions of all distributions.

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

$$\vec{J}_i = \sum_{j=1}^N (a_{ij} \cdot \text{grad} W_j + b_{ij} \cdot W_j \cdot \text{grad} \psi + \vec{c}_{ij} \cdot W_j) + \vec{d}_i \quad (2)$$

$$\sum_{j=1}^N \beta_{ij} \cdot J_j^n + \Phi_i = 0 \quad (3)$$

W_j denotes the dependent variables giving values of the affected distributions and N is the number of equations. ψ is one of the dependent variables W_j . J_j^n is the current of the j -th distribution perpendicular to the surface.

- Filter algorithms can be applied to the distributions before writing them into the result file.

Figure 5 shows an excerpt of a PROMIS-NT input deck. The respective distributions are denominated with their material names which is equivalent to certain values of i and j in the equations 1 to 3. The model developer needs no further information but the names of the required Model instances (e.g. `ProcessTemperature` for the algorithm specifying the process temperature) and the names of the appropriate input and output parameters. These informations are documented in the manual of the specific simulator or can be requested using MDL commands like `listModels` which would provide a complete list of all available `Model` classes;


```

// load prepackaged mdl-definitions
// from objectk-libraries and MDL-files
set $VMODELPATH=".:$HOME/promis-nt/models";
#include "promis-defaults.mdl"

set $VLIBRARYPATH = ".:$VPROJECT/lib";
LoadObjectLibrary "promis";

// Global Variables
Parameter<boolean> false = 0;
Parameter<boolean> true  = 1;
Parameter<stdio>   iostr;

// general simulator setup
NewModel PromisNT_Init : DiffInitModel
{
  evaluate
  {
    :inputPBF           = "example.pbf";
    :endTime            = "15._min";
    :quantityList       = "Quantities";
    :deviceInitialization = "DeviceInit";
    :processTemperature  = "RampUpTemp";
  }
}

// list of quantities to be used
NewModel Quantities : QuantityListModel
{
  evaluate
  {
    :quantity["B_active"] ={{B_active}};
    :quantity["As_active"] ={{As_active}};
    :quantity["Sxx"]      ={{Sxx      }};
    // code snipped ...
  }
}

// the process temperature model
NewModel
  RampUpTemp : ProcessTemperatureModel
{
  evaluate
  {
    if ( :time < 60.0 )
    {
      :T = 300.0 + :time * 973.0 / 60.0;
    }
    else
    {
      :T = 1273.0;
    }
  }
}

// Setup:
// segment initialization of quantities
// coefficient table for each segment

NewModel ExampleInit : SegmentInitModel
{
  evaluate
  {
    :quantityInit["Si"] = "SiInit";
    :quantityInit["SiO2"] = "DefaultInit";

    :coefficients["Si"] = "SiCoeffs";
    :coefficients["SiO2"] = "SiO2Coeffs";
  }
}

// specify the quantity initialization
// models for the Si Segment
NewModel SiInit : QuantityListModel
{
  evaluate
  {
    :qList["B_active"] = "B_Init";
    :qList["Sxx"]      =
      "DefaultQuantityInitModel";
  }
}

// code snipped ...

// Initialization check for neg. values
NewModel B_Init : QuantitiyInitModel
{
  evaluate
  {
    if ( :B_active < 0.0 )
    {
      ::iostr << "Error:...";
      :result = ::false;
    }
  }
}

// Select Coefficient Models for Si
NewModel SiCoeffs : CoefficientInitModel
{
  evaluate {
    :alpha["B_active"]["B_active"] = alpha1;
    :alpha["B_active"]["Sxx"]      = alpha2;
  }
}

// code snipped ...

NewModel alpha1 : CoeffModel {
  Instance D = DiffusionCoefficientBoron;
  evaluate
  {
    :Coeff = 1.0 / D.Coeff;
    :dCoeff_dB_active = -
      D.dCoeff_dB_active / pow(D.Coeff,2.);
  }
}

// code snipped ...

```

Figure 5: Example input deck for PROMIS-NT

2 Integration of Layout, Topography Simulation, and Resistance/Capacitance Extraction

Due to the decreasing feature sizes in today's semiconductor industry the inclusion of all relevant three-dimensional physical and geometric correlations becomes more and more indispensable for understanding and controlling the procedures which participate in integrated circuits fabrication. Therefore it is necessary not only to optimize single tools in process and device simulation but also to work towards integration of the tools and "communication" between the simulators. Since every simulation tool has its optimum data representation, the main issue of this integration is fulfilling the requirements of sophisticated methods for combining and converting the different geometry and grid structures.

2.1 Introduction

Since three-dimensional simulations make high demands on system resources, the most important requirements for three-dimensional simulation are accurate modeling within acceptable limits of memory consumption and efficient algorithms for remaining within reasonable CPU times.

Generally there are two aspects for process modeling. One is the highly accurate simulation of single process steps. Concerning topography simulation, such processes are sputter deposition, reactive ion etching or diffusion determined CVD processes. Investigation and modeling of all contributing physical effects is necessary for getting insight into the dependencies of the resulting film profile, step coverage, uniformity and other film properties on the applied process parameters. In most cases these simulations are rather time consuming but the results are very accurate.

The other aspect is process integration. For extracting the electrical characteristics of a device, it is necessary to create the appropriate input for the device simulator. Generally simple three-dimensional topography editors for all the details needed are not at disposal and integrated process simulation has to be applied for the generation of the desired structures. In most cases this simulation can be performed satisfyingly with simple and fast models for most of the process steps and only the critical process steps have to be calculated more accurately.

This dualism in process simulation can be seen in many examples: Lithography simulation may rigorously solve the Maxwell equations in three dimensions, leading to simulation results accounting for standing waves, defects, interference and reflection effects. On the other hand it may be accurate enough to use the aerial image or the mask itself as patterning information for the structure. Monte Carlo implantation algorithms supply excellent impurity or dopant distribution profiles but they are frequently replaced by fast and satisfying analytical models. Reactive ion etching with the consideration of particle distributions as close to the real conditions as possible, with the inclusion of scattering and reflection effects, absorption and re-emission probabilities or CVD simulation accounting for gas diffusion in the reactor and surface diffusion of the deposited film reveals all the details of the evolving film profile but for less critical processes simple isotropic models are exact enough and incomparably faster. These two aspects very often require completely different approaches to simulation but most commonly highly sophisticated single process step simulations are used for the development, the extraction, and calibration of the simple models.

Most recent developments in topography simulation at the Institute for Microelectronics include both aspects, accurate simulation of single process steps – such as parameter optimization for TiN magnetron sputter deposition – on the one hand and algorithm acceleration for simple models on the other hand. It is obvious, that accurate modeling also profits of the improvements in the algorithmic core of the program.

If you consider the process integration aspect, the algorithms for isotropic and anisotropic etching and deposition of the topography simulator `etch3d` were revised. The program package was also extended with the three-dimensional solid modeling tool `mgc`. It supplies basic topography manipulation and represents a link to layout simulation.

2.2 Isotropic and Anisotropic Etching and Deposition

Extending newly developed simulation tools to larger and more complex structures may reveal that the models and algorithms examined so far with rather small test geometries are too slow. The simulation of larger structures including information from layout files showed that improvements of the algorithms for isotropic and anisotropic etching and deposition are necessary for getting acceptable simulation times. The revisions and extensions will be explained in detail in the following section.

2.2.1 The Surface Movement Algorithm of `etch3d`

For the simulation we use a cell-based structuring element algorithm derived from image processing [7]. The geometry is represented by a cellular structure and a material index is assigned to each cell. The propagation of the surface is calculated by moving a structuring element – in the general three-dimensional case it is an ellipsoid – along the surface and by switching the material index of cells hit by this element. The process step to be simulated determines the shape of the structuring element. For isotropic etching and deposition the structuring element is a sphere for all cells of the surface. The radius of the sphere is determined by the etch/deposition rate and the time step. Anisotropic etching/deposition with different rates in lateral and vertical directions, is modeled by a spheroid with a vertical axis of rotation.

2.2.2 Acceleration of Isotropic and Anisotropic Etching and Deposition

As can be seen on the left of Fig. 6, the rigorous application of spheres for all surface cells, independent of their position, introduces a lot of redundant operations.

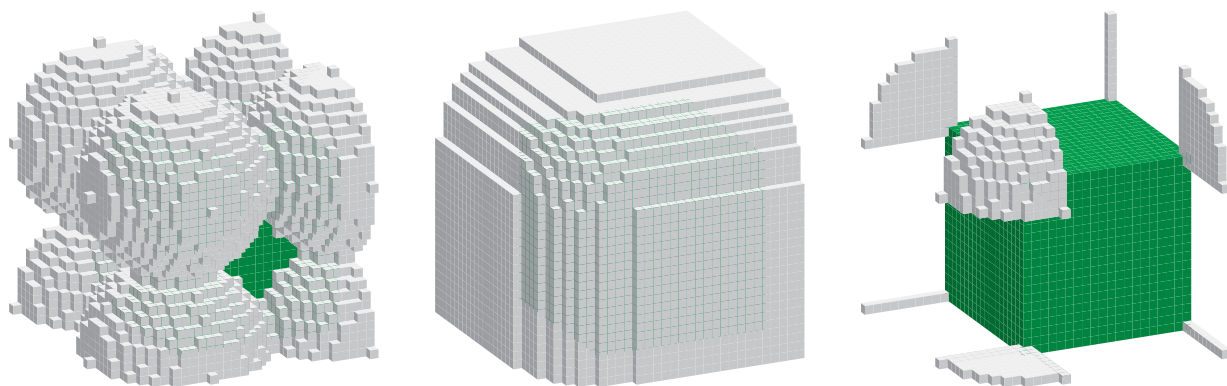


Figure 6: Simulation of isotropic deposition with different structuring elements: On the left the rigorous application of spheres, on the right the fastened algorithm with a combination of spherical segments and lines and in the middle the final structure, which is the same for both algorithms. Note that the cellular structures are plotted without surface triangulation.

Especially the cubes near the surface are hit several times by spheres originating from different neighboring surface cells but their material index needs to be changed only once. High efficiency is needed for the structuring element algorithm, since most of the algorithms have a n^3 dependency due to the three-dimensional cell array used for geometry representation. If n is the number of cells per micron, the number of operations on the three-dimensional material array depends on n^3 . The same applies for the scanning operation of the region surrounding the structuring element. The calculation time of this second operation additionally increases with the deposition rate r , because the rate determines the radius of the sphere and the volume to be scanned for each surface cell.

The reduction of the spheres to appropriate spherical segments as depicted on the right hand side of Fig. 6 provides the needed acceleration of the simulation. The initial structure is shown on the rightmost figure. The figures on the left and on the right demonstrate the different structuring elements by applying it only to the surface cells at the corners of the initial geometry. The spherical structuring elements on the left imply a lot redundant information. As can be seen in the figure, even this widely separated spheres overlap. If you consider spheres applied to all surface cells, the amount of redundant operations gets obvious. This redundancy is eliminated by restricting the spheres, depending on whether the surface cell is located within a plane, an edge or a corner of the geometry. At corners of the structure the sphere is reduced to an eighth, at edges to a one cell thick quarter circle and within planes to a one cell thick line. Even with this restrictions you still get the same final result (depicted in the middle of Fig. 6) as for the spherical algorithm.

With this measures the CPU time is up to 100 times lower than for the complete spherical algorithm. The amount of acceleration gained depends on the complexity of the structure. The more flat planes it contains, the more is the reduction in calculation time.

2.3 Inclusion of Layout Information in Topography Simulation

With the acceleration described above and with the realization of the solid modeling tool *mgc* it was possible to simulate large area structures within short time. One application of *mgc* is the function “mask”. It allows the application of a mask of specified thickness and material to the surface of a given structure. The surface may also be non planar. Simple geometric elements, such as circles, ellipses, squares and rectangles may be used for the patterning information. This simple geometries may also be combined in consequent steps. Another way to pattern the mask is using layout information. This can either be done by directly using the mask layers from the file or by including the results of a previous aerial image simulation.

Fig. 7 shows the simulation of a $5.2 \mu\text{m} \times 2.8 \mu\text{m} \times 3.2 \mu\text{m}$ two metal wire structure. The process sequence is as follows: The initial silicon block is created with the “solid”-function of *mgc*. With this step the simulation area and the vertical extension of the simulation domain are specified. The next step is the deposition of an unstructured metal film followed by the structured resist mask. The metal film is etched back by unidirectional etching. Stripping of the remaining resist is done with *mgc* “strip”. This could be done also with isotropic etching, but since this step performs no patterning the simulation is faster and accurate enough using the solid modeling tool. Having finished the metal line of the first layer, an insulating silicon dioxide layer is isotropically deposited and the metal layer, resist mask, etch back, and stripping sequence is repeated, now on the non planar oxide surface resulting in the final two layer metal structure. Fig. 7 illustrates the process flow. The layout of the two layers is depicted on the top, the resist mask for the first metal line on the left hand side in the middle, the insulating oxide layer on the first metal line right in the middle, the topography after masking the second metal layer left at the bottom, and bottommost on the right the etched back second metal layer. For getting the final geometry only the resist has to be stripped.

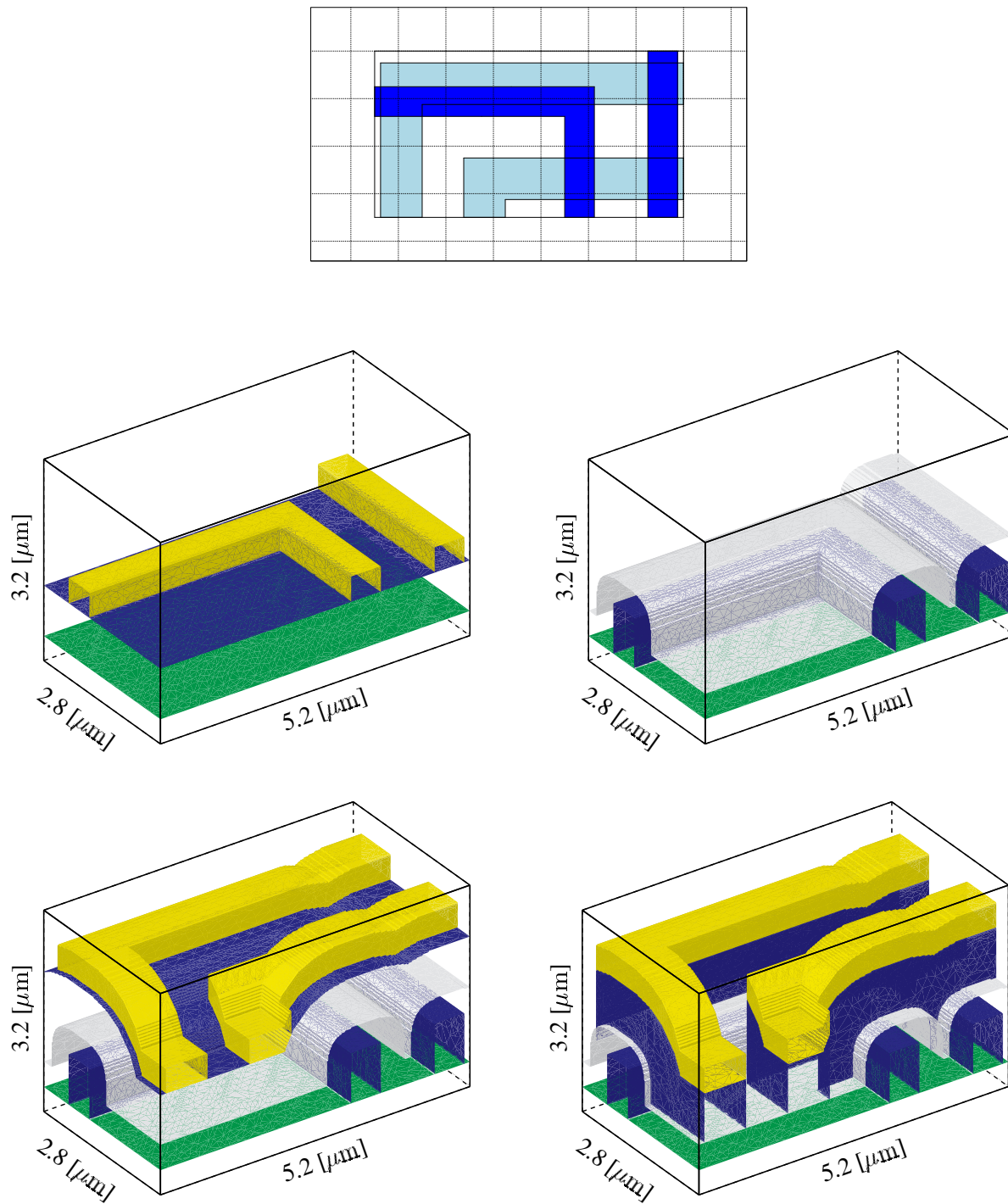


Figure 7: Different process steps of a $5.2 \mu\text{m} \times 2.8 \mu\text{m} \times 3.2 \mu\text{m}$ two metal wire structure. On the top the layout, left in the middle the resist mask for the first metal line, right in the middle the insulating oxide layer on the first metal line, left on the bottom the topography after masking the second metal layer, and bottommost on the right the etched back second metal layer.

The following figures underline the efficiency of the improved algorithm. With the original spherical algorithm the computation time for the complete structure was 218 minutes, the isotropic deposition steps for the dioxide and the two metal layers took approximately 210 minutes. With the new algorithm and the use of spherical segments the structure was simulated in 10 minutes, 2 minutes thereof were needed for the deposition.

2.4 Resistance/Capacitance Extraction

This section gives an example for a complete simulation flow starting with lithography simulation and ending in electrical characterization. Fig. 8 shows a three-dimensional model used for an electrical simulation. The first input for the process flow is the layout. It is used for calculating an aerial image. By applying a threshold to the intensity of the aerial image a binary mask is extracted which is used by the solid modeler *mgc*. Simple solid modeling – accurate enough for the first layer over the planar substrate – is used for the first layer in Fig. 8. The deviation from the rectangular structures, results from the use of the aerial image and is observable in the rounding of the metal line edges.

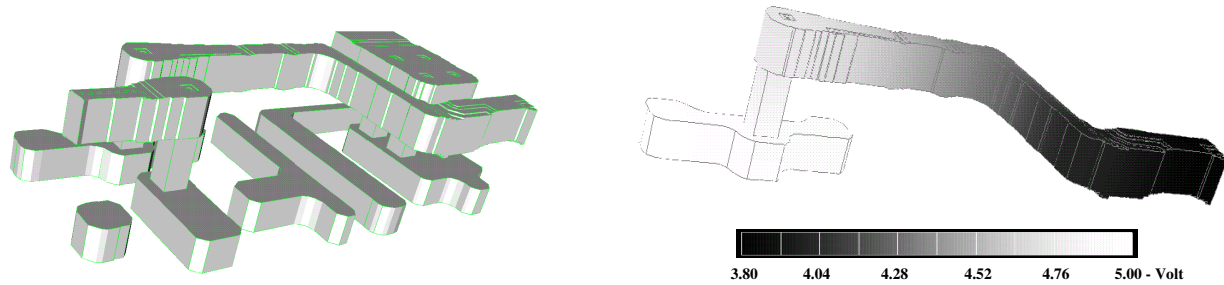


Figure 8: Three-dimensional structure including results from aerial image simulation on the left and simulated potential distribution in an interconnect wire on the right.

The pattern for the second metal layer again is extracted applying a threshold to an aerial image intensity. For the non planar structure over the insulating layer accurate topography simulation is necessary. On the right hand side of Fig. 8 the potential distribution of one wire extracted from the geometry is depicted. It was found, that the current density in some regions of the structure strongly differs from the values obtained from structures where topography and lithography is not taken into account.

3 Hydrodynamic Mixed-Mode Simulation

3.1 Introduction

Recent advances in development of semiconductor devices lead to more and more complex device structures. This concerns device geometry as well as the combination of different materials. Due to the rapid reduction of device geometries, the models describing the device physics increase in complexity. To gain additional insight into the performance of devices under realistic dynamic boundary conditions imposed by a circuit, mixed-mode simulation has proven to be invaluable. We present our approach of handling the complex situations arising from these problems. Since advanced SiGe Heterojunction Bipolar Transistors (SiGe HBTs) are currently amongst the fastest semiconductor devices, we demonstrate the capabilities of our simulator by simulating a 5-stage Current Mode Logic (CML) ring oscillator. Accurate simulation of HBT circuits must account for non-local effects such as velocity overshoot which calls for hydrodynamic (HD) mixed-mode simulation.

Our device simulator MINIMOS-NT is equipped with an extensive mixed-mode capability including HD modeling on distributed devices. In general the convergence of HD simulations is known to be poor. Therefore we enhanced MINIMOS-NT with an interface where different iteration schemes such as the full Newton scheme and various block iteration schemes based upon the Gummel scheme can be easily defined and augmented with special damping algorithms.

3.2 Segments

To allow for a flexible handling of distributed devices in a mixed-mode circuit simulation, their geometry is partitioned into independent regions, so-called segments. For these segments different sets of parameters, models and algorithms can be defined independently. As an example it is possible to solve only Poisson's equation on one segment, or the transport equation for only one carrier type in addition to Poisson's equation on another segment.

The segments are linked together by interface models which account for the interface conditions. This results in high flexibility which allows, for example, to use a HD model on one segment and a drift diffusion (DD) model on another segment. Furthermore, the explicit treatment of volume and interface models leads to a better condition of the linearized system compared to a method which simply reduces grid spacing in the vicinity of heterojunctions[8].

3.3 Model Assembly

During simulator development we frequently encountered the problem, that several different formulations for, e.g., the HD model are available (either in the model itself or in its discretization). To compare these formulations easily in a general manner, without adding a lot of keywords or recompiling the code, we developed a particular strategy. First, each partial differential equation (PDE) is split into its constituent terms. Second, these terms are combined to so-called groups, which then form the basic unit of model control. It is important to note that these groups can contain terms from more than one PDE, since the groups serve as an abstraction of the desired effect to model.

Finally, these groups are combined to build the complete equation set. For instance, in an oxide segment it is sufficient to use only the Poisson-group which contains the basic terms of the Poisson equation (discrete differential operator, contributions to the space charge density). In a channel of a HEMT it might

be sufficient to use only the Poisson and the electron group (contains the static DD continuity equation and the electron charge term of the Poisson equation). In the base of a HBT one should use the Poisson, electron and hole group. Furthermore, the Shockley-Read-Hall (SRH) group should be added to account for the recombination rates in the continuity equation and the trap charge in the Poisson equation. When considering transient analysis, the transient-groups add the time dependent terms to each equation. With this modular scheme it is easy to replace the DD electron group by the HD electron group or the static SRH group by the dynamic SRH without affecting the other groups. To guaranty model consistency a proper design of these groups is of course crucial.

3.4 Quantity Classes

Each equation term requires on its respective segment so-called quantity classes, such as potential, carrier concentrations and carrier temperatures. These quantity classes are automatically generated and initialized and can be independently marked for solving. An example might be a segment where a HD model should be solved. The HD model is selected by its appropriate group, but in many cases the influence of one carrier type turns out to be negligible. Now it can be decided to either completely ignore this carrier type by not selecting its model group, or to ignore only its temperature quantity by unmarking the latter for the solution process.

Despite of the quantity classes generated by the model groups, three classes are automatically generated when using the mixed-mode feature: the node voltage (NV), branch current (BC), and the fixed node voltage quantity class (FNV). The NV and the BC classes result from the modified node voltage analysis formulation, whereas the FNV class is actually part of the NV class and allows the user to, e.g., set an initial condition or to fix some of the node voltages to their respective old values at distinct timepoints.

3.5 Programmable Iteration Scheme

The features outlined above can be utilized when defining the iteration sequence. Each iteration block can contain other arbitrarily nested sub-blocks. First, for each block the groups to use are defined. Second, it must be specified which quantities are to be solved for and which are to be left to their respective old or initial value. The simulator will iterate until a block-specific termination criterion is satisfied. These criteria can be arbitrary expressions containing norms, iteration counters, time step information and much more. After each iteration the sub-blocks are entered recursively. Blocks can be empty to form a linear sequence of sub-blocks only.

3.6 Matrix Assembly

For each iteration all marked quantity-related equations are assembled into the system matrix. By unmarking for instance the node voltage quantity class, the single devices become decoupled. However, they are all stored in the same matrix which increases the time needed to find the solution, compared to several small matrices. To overcome this drawback, a two-level Newton algorithm can be specified by using a separate iteration block for each device. In contrast, by marking all quantities, a full Newton algorithm is achieved. Furthermore, it is easy to identify devices causing convergence problems. For these devices a separate sub-block employing a decoupled iterative scheme might be used.

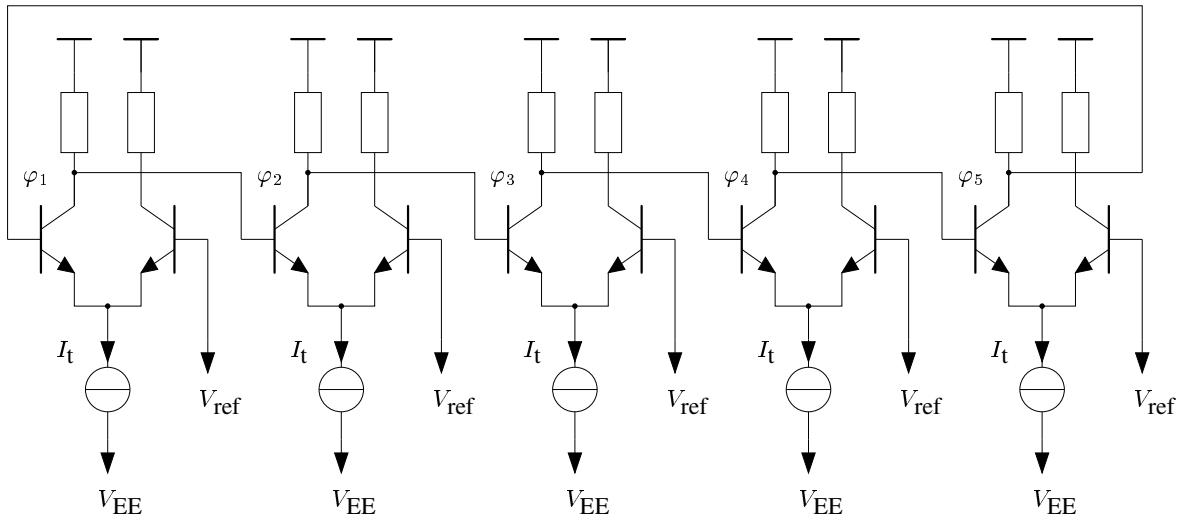


Figure 9: CML ring oscillator circuit.

3.7 Example

As a particular example, a 5-stage CML ring oscillator containing 10 SiGe HBTs (Fig. 9) was simulated. The two-dimensional HBT structure was taken from [9] and is shown in Fig. 10.

The physical models for the SiGe alloy are temperature and mole-fraction-dependent, the low field mobility model distinguishes between majority and minority electron mobilities on one hand, and between different dopant species on the other hand, both as a function of temperature and dopant concentration.

Fig. 11 shows a comparison of the DD and HD results of the ring oscillator given in Fig. 9. A full Newton scheme has been employed to both the DD and the HD model. Since the hole density in the collector and emitter is very low and the doping in the base is very high, the holes were modeled by a DD model only. The rank of the matrix was 29169 for DD and 36389 for HD. The circuit simulation was carried out for 50 timesteps and took 1h 56min for DD and 3h 26min for HD on a Pentium II 266 MHz LINUX workstation. The DD simulation shows a much lower inverter delay time compared to the HD simulation. This is due to the velocity overshoot in the base-collector space charge region which can not be modeled using a DD simulation as pointed out by [9]. Furthermore, since the current is higher in the case of HD simulation, the overall speed of the circuit increases. The error of the DD simulation in the inverter delay is in the range of 60% compared to the HD simulation, which proves the necessity for a HD model.

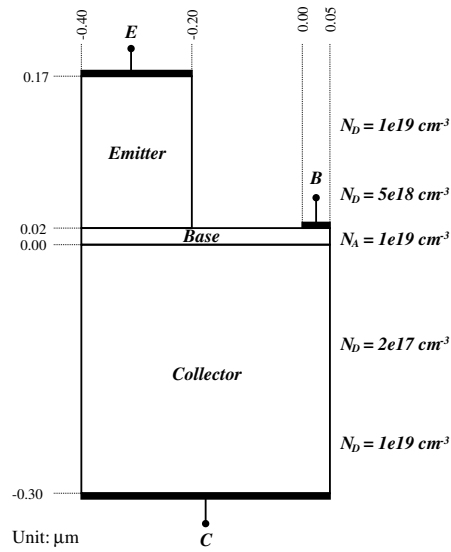


Figure 10: The HBT structure.

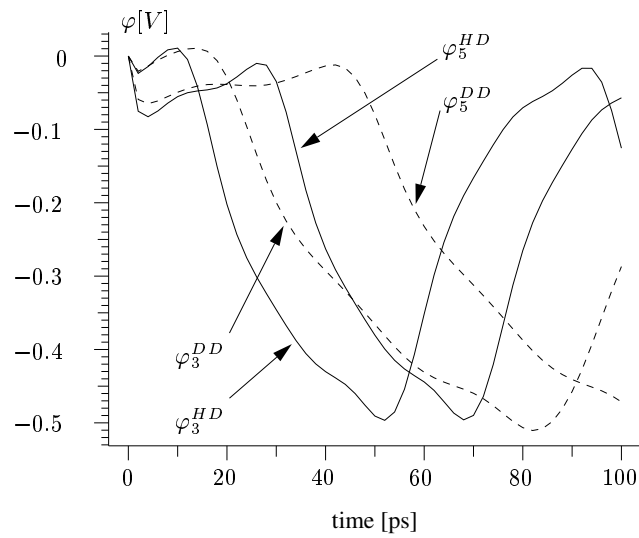


Figure 11: Comparison of the DD versus HD transient response.

4 Two-Dimensional Simulation of Ferroelectric Materials with MINIMOS-NT

4.1 Introduction

The development of non-volatile memory cells using ferroelectric materials leads to designs with two-dimensional hysteretic field properties. The simulation of the two-dimensional hysteresis curve leads to the nontrivial problem of field rotation and forces the calculation of a set of parameters for the nonlinear locus curve 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.

4.2 The Simulation Model

Primary focus was laid on the simulation of field rotation. This means that as described in [10] and [11] a constant rotation of a magnetic field will cause a lag angle χ of the induction. It is intuitive that the properties for ferroelectric problems will be similar. For a rigorous two-dimensional analysis, the simple approach to decrease the electric field first to zero, then to increase it to the value of the next operating point and to add the two derived polarization components cannot be employed, as it is inconsistent with the one-dimensional hysteretic properties. Even more the results strongly depend on the distance between the calculated operating steps. According to this we assume a straight trajectory between the vectors of the old and the newly applied electric field. This also assures a proper numerical behavior if the applied voltage steps will be increased. The basic principle of our model is to split the polarization \vec{P}_{old} and the electric field \vec{E}_{old} of the previous operating point into the components in the direction of the next applied electric field $\vec{P}_{old,\parallel}$ and $\vec{E}_{old,\parallel}$ as well as orthogonal $\vec{P}_{old,\perp}$ and $\vec{E}_{old,\perp}$. For each of these components a locus curve is calculated. These curves lead to the polarization in direction of the electric field and the remanent polarization in the orthogonal direction, thus forming a primary guess \vec{P}_0 for the next polarization.

As plotted in Fig. 12 \vec{P}_0 is divided into a component in the direction of the electric field \vec{P}_{\parallel} and into the orthogonal component \vec{P}_{\perp} . The scalar values of the two components are added and compared to the maximum polarization at the given magnitude of the electric field $\|\vec{E}\|$, forming an upper limit for the available number of switching electric dipoles. Due to the vanishing electric field in the normal direction making it easier to switch the dipoles in this direction than the dipoles held by the electric field, the orthogonal component \vec{P}_{\perp} is reduced appropriately in respect to this limit. This is shown schematically in Fig. 13 and it leads to the actual polarization vector \vec{P}_1 and the lag angle χ . As a first approach to the upper limit in the available number of switching dipoles the saturation polarization \vec{P}_{sat} was considered, but the simulator is already prepared to use any function of the magnitude of the applied electric field $\|\vec{E}\|$. The algorithm to handle field rotation is also capable to deal with three-dimensional problems.

For a general approach to two-dimensional hysteretic effects an inhomogeneous field distribution has to be assumed. This prevents the usage of a simple one-dimensional hysteresis model using the same locus curve for the complete ferroelectric region. According to the algorithm presented above two different locus curves have to be calculated for each grid point. Therefore, numerical methods to calculate the locus curves as described in [12] and [13] 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. The parameters of the locus curves are calculated using the projections of the old directions of the old electric field $\vec{E}_{old,\parallel}$, $\vec{E}_{old,\perp}$ and the old polarization

field $\vec{P}_{old,\parallel}$, $\vec{P}_{old,\perp}$. This hysteresis model was chosen in accordance to [14]. To adjust this model to the two-dimensional case the locus curves are not calculated due to the last turning point but to the saturation polarization, so the lancette curves will not exactly fit the one dimensional hysteresis model. The resulting locus curves plotted in Fig. 17 are the result of a transient simulation of a capacitor with a ferroelectric dielectric.

For a good numerical behavior it is calculated whether the electric field will be incremented or not before starting the actual solving process, because for each of these two cases a different set of parameters has to be calculated, thus leading to different locus curves and derivatives. This assures that the correct derivatives will be used in the Jacobian matrix. A suitable approach to receive this direction information is to vary only the linear part of the electric displacement and to keep the polarization constant.

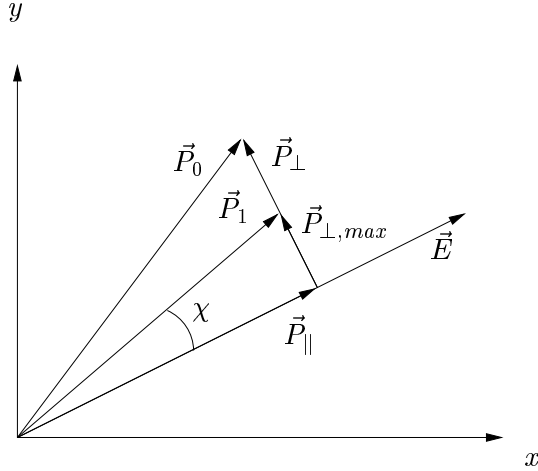


Figure 12: Calculation of the resulting polarization.

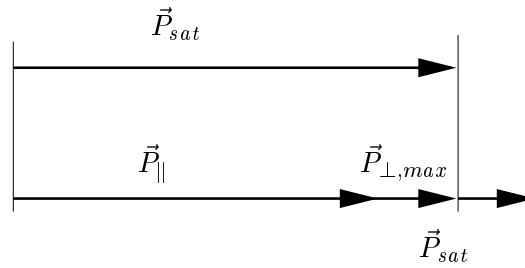


Figure 13: Reduction of orthogonal component.

4.3 Simulation of the FEMFET

Basically a FEMFET is an NMOS transistor modified by inserting a ferroelectric layer in the sub-gate area, as schematically outlined in Fig. 14. The FEMFET is based on the hysteretic properties of the polarization and the displacement of the ferroelectric material. According to Poisson's equation the displacement influences the charge at the surface of the substrate. The different contributions of the space charge density of a usual NMOS transistor and a FEMFET were calculated with MINIMOS-NT and are plotted in Fig. 15 and Fig. 16. The threshold voltage of the NMOS was 0.7 V and, due to the increased displacement, 0.6 V for the FEMFET. The operating point of the ferroelectric material was chosen on the initial polarization curve. In this case the ferroelectric polarization increases the displacement and as shown in the figures 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 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. As expected, the I-V characteristics of the transistor show also a hysteresis. Fig. 18 shows the simulated I-V characteristics for NMOS and FEMFET received by sweeping the gate voltage from zero to saturation and vice versa. The bulk voltage was set to 0.5 V, the drain voltage to 0.1 V. With a flat hysteresis curve we received a voltage shift of 0.1 V. This proves that it is possible to simulate materials with hysteretic properties even if they are included in complex structures like a FEMFET.

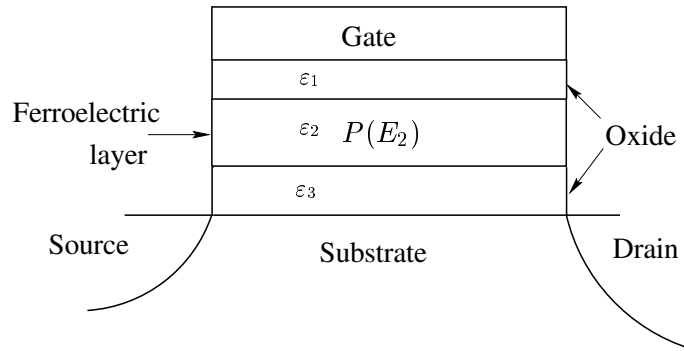


Figure 14: Ferroelectric non-volatile memory field effect transistor.

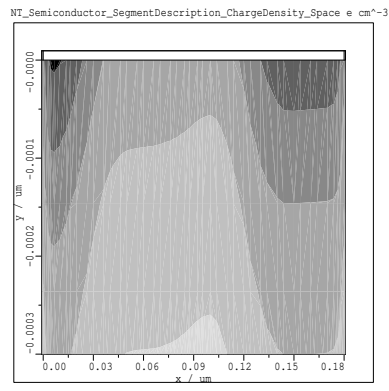


Figure 15: Space charge density of the NMOS.

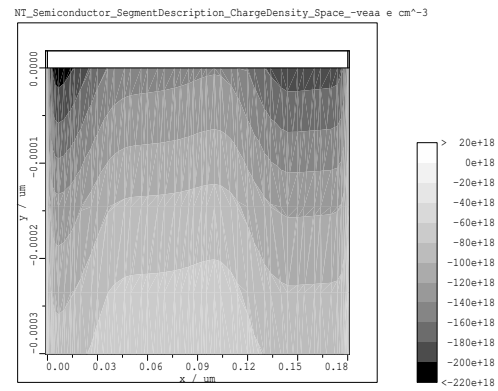


Figure 16: Space charge density of the FEMFET.

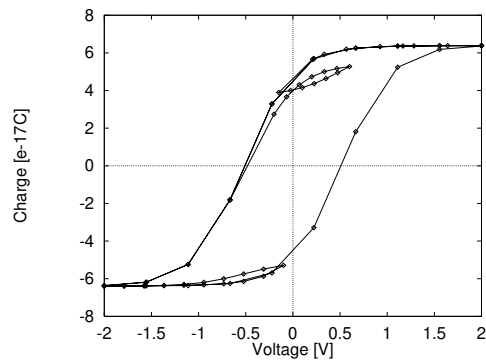


Figure 17: Simulated hysteresis curve.

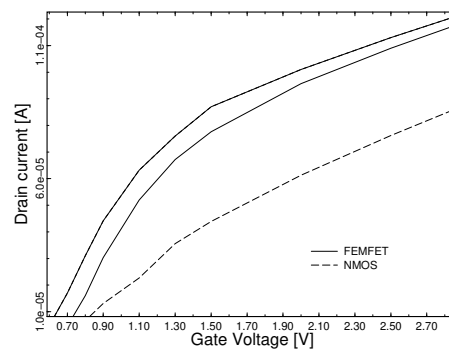


Figure 18: Simulated I-V characteristics of FEMFET and NMOS.

5 Simulation of Single-Electron Devices and Circuits

5.1 Introduction

Quantization of charge in metallic or semiconductor islands is usually not directly noticeable. However, when the size of such islands is in the nano-meter regime, that is when the total capacitance becomes very small and the charging energy is larger than the thermal energy, then the change in free energy associated with the addition or subtraction of a single electron from an island, or a quantum dot, becomes significant.

New phenomena, such as the Coulomb blockade which is a suppression of current flow at low bias, and Coulomb oscillations, a time or space correlated transfer of electrons through tunnel junctions, appear. With these new quantum-effects it is possible to control the movement and position of single electrons. Beside the wanted characteristics of controlled transfer of single electrons, unwanted effects arise, too. These are co-tunneling, a simultaneous tunneling of two or more electrons in different tunnel junctions, or the sensitivity to uncontrollable impurities that are dispersed through out the material, and which are disturbing the charge distribution and hence the Coulomb blockade.

Single-electron technology is a very promising alternative to state of the art ULSI technology. It enables:

- electronic circuits with ultimately low power consumption
- high integration densities due to the small feature size
- improvement of device characteristics with down-scaling

5.2 SIMON

A simulator for single-electron devices and circuits, named SIMON (simulation of nanostructures), has been developed [15][16]. A graphical circuit editor, which is embedded in a graphical user interface shown in Fig. 19, guarantees easy use and quick circuit design. Capacitors, tunnel junctions, constant voltage sources, piece-wise linearly time dependent voltage sources, and voltage-controlled voltage sources can be arbitrarily connected. One may choose between transient and stationary simulation modes and decide upon the order of co-tunneling ([17][18]) which should be included in the simulation. There are no a priori limitations on the number of tunnel junctions or circuit nodes.

From the circuit description SIMON extracts the capacitance matrix of the network. By exploiting the symmetry of the capacitance matrix, the electrostatic energy U of the circuit can be calculated efficiently. To simulate one time step in the course of a Monte Carlo simulation, the change in free energy for all possible tunnel events is calculated after (4). Then a tunnel rate, (5), is derived. Assuming a Poisson process, from the tunnel rate a concrete time intervall is calculated using (6). Finally, the event with the shortest time intervall is selected.

The Coulomb blockade is a manifestation of the tunnel rate dependence on the change in free energy. Free energy is the difference of electrostatic energy U stored in the network and work done by voltage sources W .

$$F = U - W \quad (4)$$

Every time an electron tunnels, the state of the circuit changes. The state of the circuit is determined by the set of all node charges and node voltages. A change in state causes a change in free energy. Since the circuit will tend to a lower energy state, events that reduce the free energy will be more likely.

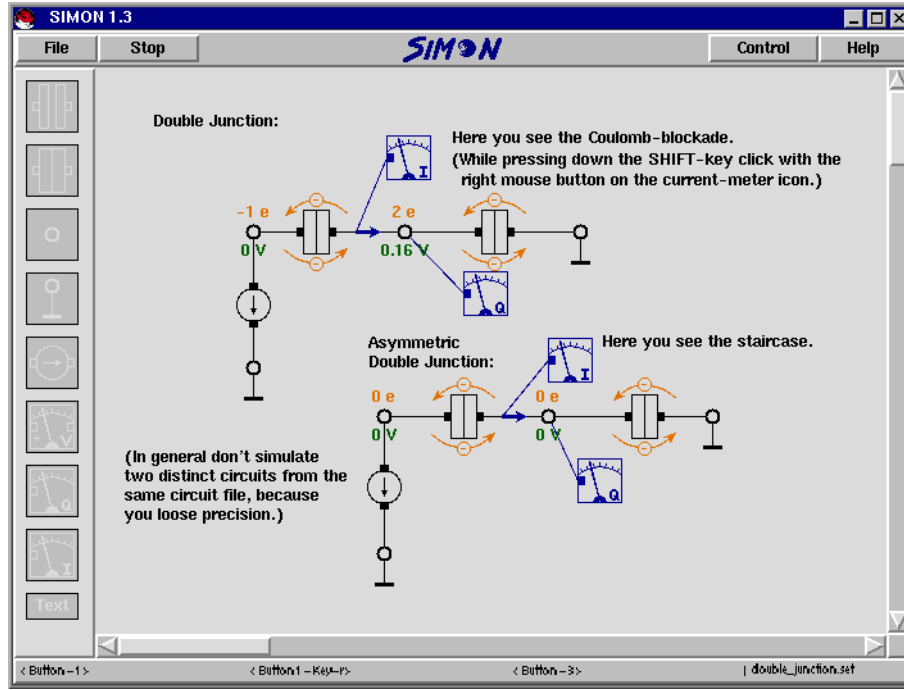


Figure 19: Graphical user interface of SIMON.

Once the change in free energy for all possible tunnel events is determined, the tunnel rates can be calculated by [19] [20]:

$$\Gamma = \frac{\Delta F}{q_e^2 R_T (1 - e^{-\frac{\Delta F}{k_B T}})}. \quad (5)$$

With this information the Monte Carlo part of the simulator is entered. Here ΔF is the change in free energy, k_B is the Boltzmann constant, T is the absolute temperature, R_T the tunnel resistance and q_e the elementary charge.

Tunneling is modeled as a Poisson process. To every possible event a duration after which an electron will tunnel is chosen randomly. From the probability distribution of the Poisson process one can deduce the following formula [21].

$$\Delta t = -\frac{\ln r}{\Gamma(T, R_T, \Delta F)} \quad (6)$$

Here r is an evenly distributed random number in the interval $[0, 1]$, $\Gamma(T, R_T, \Delta F)$ is the tunnel rate depending on temperature T , tunnel resistance R_T and change in free energy ΔF . The winner is the tunnel process with the shortest time Δt . This calculation of tunnel rates, time intervals and winners is done many times to either simulate the transient behavior of the network or to determine quasi stationary characteristics by averaging over many events.

During the process of calculating the electrostatic energy all node voltages and node charges are determined. The user specifies which one are output to data files. Only the branch currents that the user is interested in are calculated and output. By simulating many tunnel events the macroscopic device characteristics are obtained.

This simulation method relies on two basic assumptions. Firstly, voltage sources are considered to have no internal resistance. Consequently, charging and discharging of capacitances occurs instantaneously.

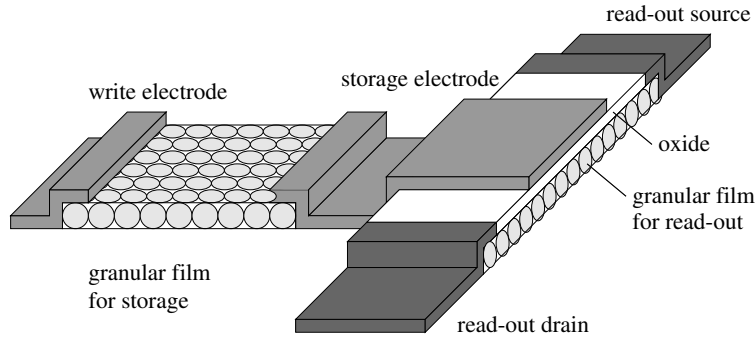


Figure 20: The T-memory cell

Secondly, the simulator treats electrons as point charges that hop from island to island via tunnel junctions. In other words, the electron states must be localized. This requires all tunnel resistances R_T have to be much bigger than the quantum resistance R_q [22].

$$R_T \gg R_q = \frac{h}{q_e^2} \approx 25.8\text{k}\Omega \quad (7)$$

Here h is Planck's constant and q_e is the elementary charge.

5.3 The T-Memory Cell

High density memories seem to be a very suitable application for single-electronics. The storage of binary information using Coulomb-blockade phenomena has been demonstrated in theory and in practice. However, several issues were identified which make a large scale integration of these single-electron memory cells with today's technology almost impossible. The most notable of these problems are operation at room temperature, which demands lithographic resolution of less than 10 nm, and the sensitivity to impurities and stray charges, which demands perfectly clean production processes. Considering the current trend in ULSI technology, neither of these issues will be resolved in the near future. Thus a new approach which combines new ideas and new insights is needed.

The main requirements to make single-electron circuits feasible for industrial application are:

- room temperature operation
- insensitivity to random background charge
- mass production with optical lithography

These requirements can be met by using granular films (see e.g. [23]). Their usage eliminates the necessity for nano-lithography, since nanoscopic tunnel junctions and granules are formed naturally. Additionally, granular films provide an averaging effect of the Coulomb-blockade, which dramatically reduces the sensitivity to random background charges.

Several memory designs were analyzed with SIMON. In consequence of these investigations, we proposed a new memory design called T-memory (Fig. 20).

This cell consists of two granular film batches which are arranged in T-form and which are capacitively coupled at the junction. It turns out that using two granular films, one for storing charge, and the other

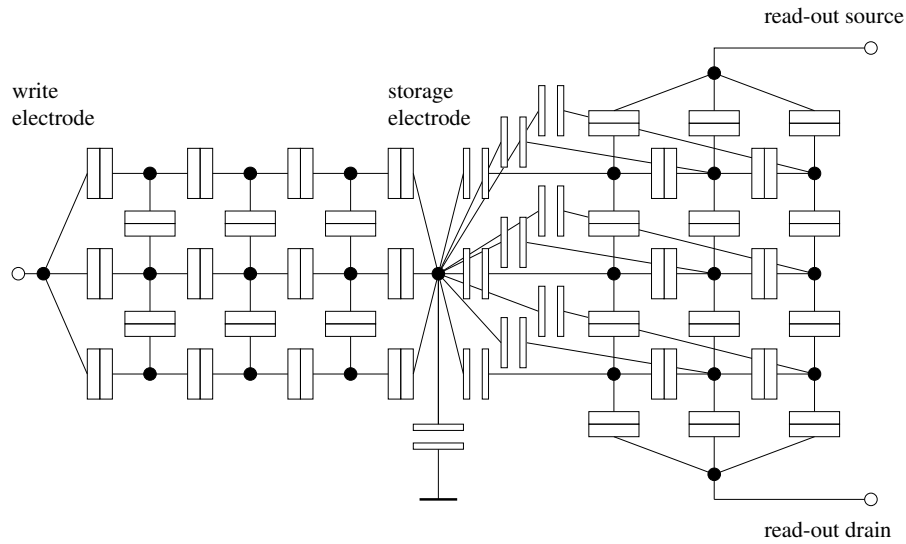


Figure 21: Equivalent circuit of the T-memory cell used in the simulation

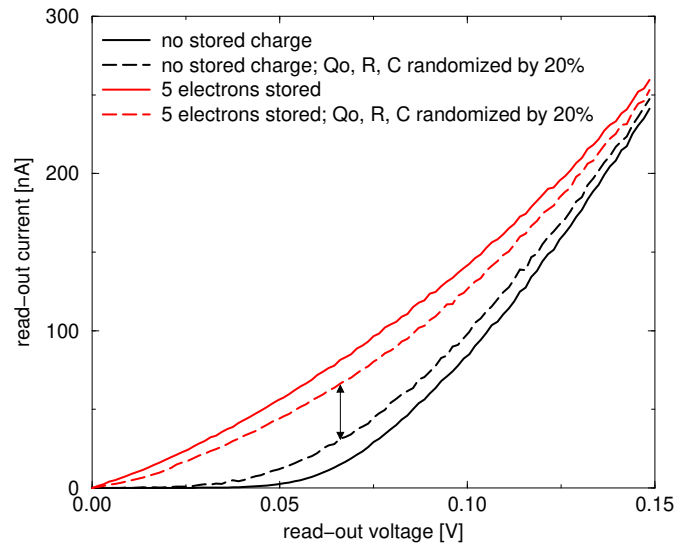


Figure 22: Read-out I-V characteristics of T-memory cell for two different storage charges.

to read-out the stored information, gives the designer great flexibility in optimizing the cell. This enables to tune storage time, to achieve, for example, a non-volatile memory, and at the same time to optimize read-out characteristics for a reliable information retrieval.

The equivalent circuit of the T-memory cell used for the simulation with SIMON is shown in Fig. 21.

Two different read-out mechanisms are imaginable. The first one makes use of the dependence of the static IV-characteristics of the read-out transistor on the stored charge (Fig. 22).

The other mechanism is a destructive read-out. One attempts to discharge the storage node. If with the read-out transistor current oscillations are sensed, the cell was charged (state “1”), otherwise it was empty (state “0”). Accordingly, the contents of the cell has to be restored. Fig. 23 shows the signals when reading a “1” and a “0”.

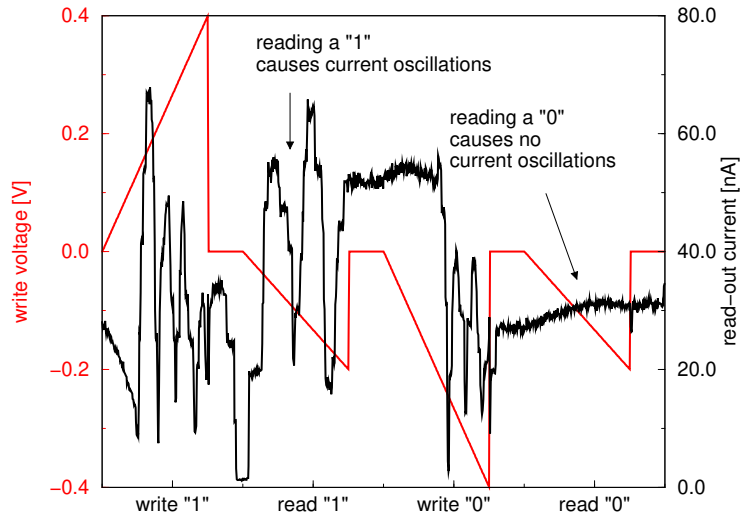


Figure 23: Coulomb oscillations occuring in the read-out transistor

In conclusion, high density memories are supposed to be a promising field for applying single-electronics in the near future. The use of granular films enables fabrication with optical lithography. A multi purpose single-electron device and circuit simulator, SIMON, has been developed which is capable of simulating static and transient circuit behavior using the Monte Carlo method. With the simulator various single-electron memory designs have been investigated with respect to the maximum operating temperature and sensitivity to random background charges. With this experience the new T-memory cell has been proposed which has the potential for high volume production with today's process technology. The cell can be operated at room temperature and is insensitive to uncontrollable impurities.

References

- [1] J.K. Ousterhout. *Tcl and the Tk Toolkit*. Addison Wesley, 1994.
- [2] D.W. Yergeau and R.W. Dutton. Alamode: A Layered Model Development Environment for Simulation of Impurity Diffusion in Semiconductors. 1997. Documentation for release 97.06.18.
- [3] J. Litsios. *A modeling language for mixed circuit and semiconductor device simulation*. Hartung-Gorre, 1996.
- [4] M. Radi, E. Leitner, E. Hollensteiner, and S. Selberherr. AMIGOS: Analytical Model Interface & General Object-Oriented Solver. In *International Conference on Simulation of Semiconductor Processes and Devices*, pp 331–334, Cambridge, Massachusetts, 1997.
- [5] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, 1986.
- [6] H. Puchner. *Advanced Process Modeling for VLSI Technology*. Dissertation, Technische Universität Wien, 1996.
- [7] E. Strasser and S. Selberherr. Algorithms and Models for Cellular Based Topography Simulation. *IEEE Trans. Computer-Aided Design*, 14(9):1104–1114, 1995.
- [8] T. Simlinger, R. Deutschmann, C. Fischer, H. Kosina, and S. Selberherr. Two-Dimensional Hydrodynamic Simulation of High Electron Mobility Transistors Using a Block Iterative Scheme in Combination with Full Newton Method. In G.L. Baldwin, Z. Li, C.C. Tsai, and J. Zhang, editors, *Fourth Int. Conf. on Solid-State and Integrated-Circuit Technology*, pp 589–591, Beijing, China, 1995.
- [9] B. Neinhüs, P. Graf, S. Decker, and B. Meinerzhagen. Examination of Transient Drift-Diffusion and Hydrodynamic Modeling Accuracy for SiGe HBTs by 2D Monte-Carlo Device Simulation. In H. Grünbacher, editor, *27th European Solid-State Device Research Conference*, pp 188–191, Stuttgart, Germany, 1997. Editions Frontieres.
- [10] H. Pftzner. Rotational Magnetization and Rotational Losses of Grain Oriented Silicon Steel Sheets—Fundamental Aspects and Theory. *IEEE Transactions on Magnetics*, 30:2802–2807, September 1994.
- [11] P. Weiss and V. Planer. Hysteresis dans les champs tournants. *Journal de Physique (Theor. et Appl.)*, 47:5–27, 1908.
- [12] S. L. Miller and P. J. McWhorter. Physics of the Ferroelectric Nonvolatile Memory Field Effect Transistor. *Journ. of Apl. Phys.*, 72:5999–6010, 1992.
- [13] 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.
- [14] 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.
- [15] Ch. Wasshuber, H. Kosina, and S. Selberherr. SIMON— A Simulator for Single-Electron Tunnel Devices and Circuits. *IEEE Trans. Computer-Aided Design*, 16(9):937–944, 1997.
- [16] C. Wasshuber and H. Kosina. A Single-Electron Device And Circuit Simulator. *Superlattices & Microstructures*, 21(1):37–42, 1997.

- [17] C. Wasshuber and H. Kosina. A Single Electron Device and Circuit Simulator with a New Algorithm to Incorporate Co-tunneling. In *International Conference on Simulation of Semiconductor Processes and Devices*, pp 135–136, Tokyo, Japan, 1996. Business Center for Academic Societies Japan.
- [18] D.V. Averin and Y.V. Nazarov. Virtual Electron Diffusion during Quantum Tunneling of the Electric Charge. *Physical Review Letters*, 65(19):2446–2449, 1990.
- [19] H. Grabert, G.L. Ingold, M.H. Devoret, D. Esteve, H. Pothier, and C. Urbina. Single Electron Tunneling Rates in Multijunction Circuits. *Zeitschrift für Physik B - Condensed Matter*, 84:143–155, 1991.
- [20] H. Grabert and M.H. Devoret, editors. *Single Charge Tunneling – Coulomb Blockade Phenomena in Nanostructures*, volume 294 of *NATO ASI Series B*. Plenum Press, 1992.
- [21] M. Kirihara, N. Kuwamura, K. Taniguchi, and C. Hamaguchi. Monte Carlo Study of Single-Electronic Devices. In *Extended Abstracts of the International Conference on Solid State Devices and Materials*, pp 328–330, Yokohama, 1994.
- [22] K.K. Likharev. Correlated Discrete Transfer of Single Electrons in Ultrasmall Tunnel Junctions. *IBM J.Res.Dev.*, 32(1):144–158, 1988.
- [23] K. Yano, T. Ishii, T. Sano, T. Mine, F. Murai, and K. Seki. Single-Electron-Memory Integrated Circuit for Giga-to-Tera Bit Storage. In J.H. Wuorinen, editor, *IEEE Int. Solid-State Circuits Conference*, pp 266–267, 458. IEEE, 1996.