VISTA Status Report
June 1994

W. Bohmayr, S. Halama, C. Pichler, S. Selberherr, T. Simlinger,
E. Strasser, W. Tuppa

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

Contents

Summary

The etch simulator supports now different physical models for etching and deposition. This is done by a superposition of isotropic and anisotropic reactions of directly and indirectly incident particles which is used to calcuate the resulting velocity vector of surface points.

The monte carlo ion implantation module supports now in addition to amorph crystalline targets. Implantation damage is calculated by the modified Kinchin-Pease model.

A wrapper to the process simulation tool TSUPREM-4 of TMA was written. The wrapper takes PIF input and converts it to a TSUPREM-4 save file or a TIF (Technology Interface Format) to run the simulator. The results are converted back to PIF.

The gridding tool VORONOI as described in the last VISTA status letter was enhanced to provide an interpolation function to the generated grid. This speeds up interpolation to the new grid by many times, since all related information is already available within voronoi.

A full process was simulated using the SFC (Simulation Flow Controller) and the new TIF wrapper. The voronoi gridding tool is used for grid generation and interpolation.

The visualization allows now displaying multiple attributes at the same time. This is useful if e.g. a concentration is defined with an own attribute for each segment because of the modeling of segregation. One more new feature allows the analysis of field vectors.

The SFC was enhanced to support split point in a simulation flow. Only necessary steps are calculated to minimize simulation time. The run data management stores the output of all simulator calls for later review (may be discarded on request).

A byte code compiler was developed for the XLISP interpreter. This reduces the needed amount of nodes significantly and speeds up execution of LISP code. Byte code compiled code can be mixed at any level with uncompiled code.

1 Modeling of Etching and Deposition Processes

1.1 Introduction

A lot of topography processes are strongly affected by the shape of the surface. Many successful two-dimensional simulation programs for etching and deposition use macroscopic point advancement models which consider information about particle fluxes and surface reactions to calculate etch or deposition rate distributions along the exposed surface. This approach is extremely desirable and has also been applied in our simulator, since a variety of process models for etching and deposition in the literature already exists and quantities such as etch or deposition rates are easily measurable in semiconductor technology.

1.2 Process Models for Etching and Deposition

To determine the rate contributions of incoming particles both in etching and deposition the simulator must be capable to calculate the resulting particle flux incident at a surface point. Therefore the region above the wafer is divided up into several patches and the incident flux is then integrated over those patches of the "sky" which are visible from the surface point. To determine if a patch is visible from a point on the surface a shadow test has to be performed along a given direction which is within the cellular material structure of our simulator simply the matter of following a discretized line of cells from the surface cell to the boundary of the simulation area. If any cell on this line is a material cell, then the surface cell is shadowed.

As a basic concept for modeling etching processes we consider a linear combination of isotropic and anisotropic reactions of directly and indirectly incident particles to calculate the resulting velocity vector of a surface point. The isotropic reaction is mainly a chemical reaction affected by a reactive gas, in which the reactive particles have short mean free paths and move randomly. The etch rate has no orientation or flux dependencies and is described as a constant. The anisotropic reaction is a physical or chemical reaction, where the particles have long mean free paths compared to the device dimensions and angular particle fluxes must be considered. Rate contributions from directly incident ions, directly incident neutral particles, and from reflected particles are taken into account to calculate the resulting etch rate of a surface point. Deposition modeling is based on the original work of Blech who developed a model for describing two-dimensional profiles of evaporated thin films over steps [1]. The growth vector of a surface point can be calculated by considering the angular flux distribution of incoming particles at a surface point. Different assumptions on the flux distribution function result in evaporation or sputter deposition processes.

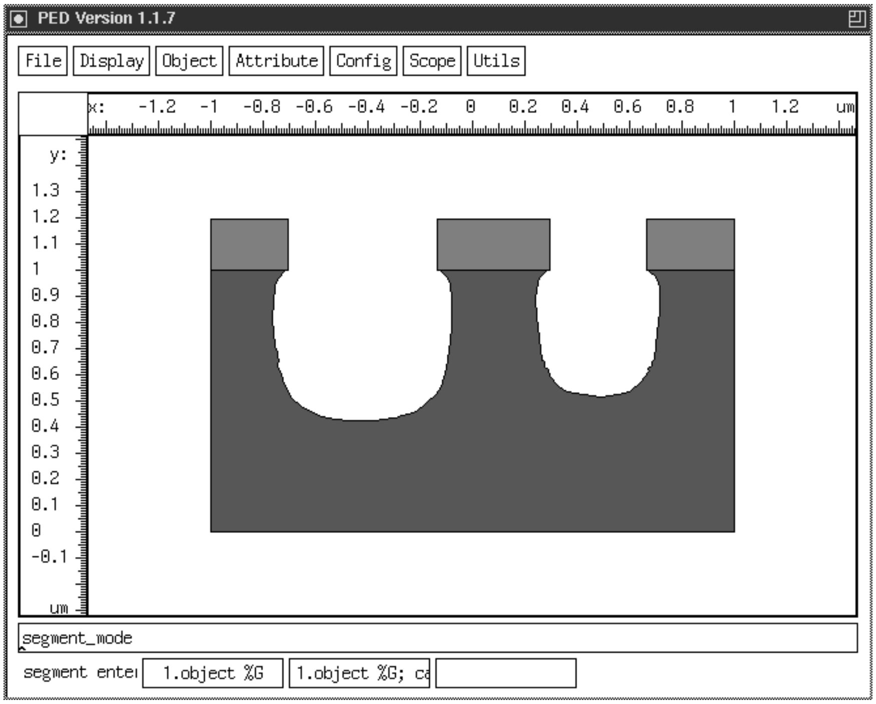
1.3 The Surface Advancement Method

The algorithm for surface advancement is based on morphological operations which provide a well defined method for altering a given image with respect to some predetermined geometric shape known as *structuring element* [2]. The basic idea behind the surface evolution algorithm is to consider the simulation geometry as a black and white image and to relate the spatial dimensions of the structuring element at a surface point to a precalculated local etch or deposition rate [3] [4].

The material is represented using an array of cells each characterized as vacuum or material cell. During the simulation a structuring element moves along the exposed surface and depending on the simulated process cells are either removed or added which are located within the structuring element (midpoint of the cell is considered). The way how the topography evolves at a point on the surface depends on the spatial dimension and on the orientation of the applied structuring element.

1.4 Simulation Results

Fig. 1 shows the typical barreling phenomenon which results in ion enhanced plasma etching due to high energetic ions that increase the etch rate where they hit the surface (mainly at the bottom of the trench) and due to reactive neutrals which also attack the sidewalls. The picture also shows the well known aperture effect (etch rate decreasing due to limited delivery of ions and radicals) resulting in a deeper trench where the mask size opening is larger.



segment_mode

segment ente 1.object %G 1.object %G; c

Figure 1: Reactive ion etching of trenches.

Fig. 2 shows etching caused by reactive neutrals only leading to a poor anisotropy of the trenches and Fig. 3 shows the result of a sputter deposition process using a cosine distribution function for the incident particles.

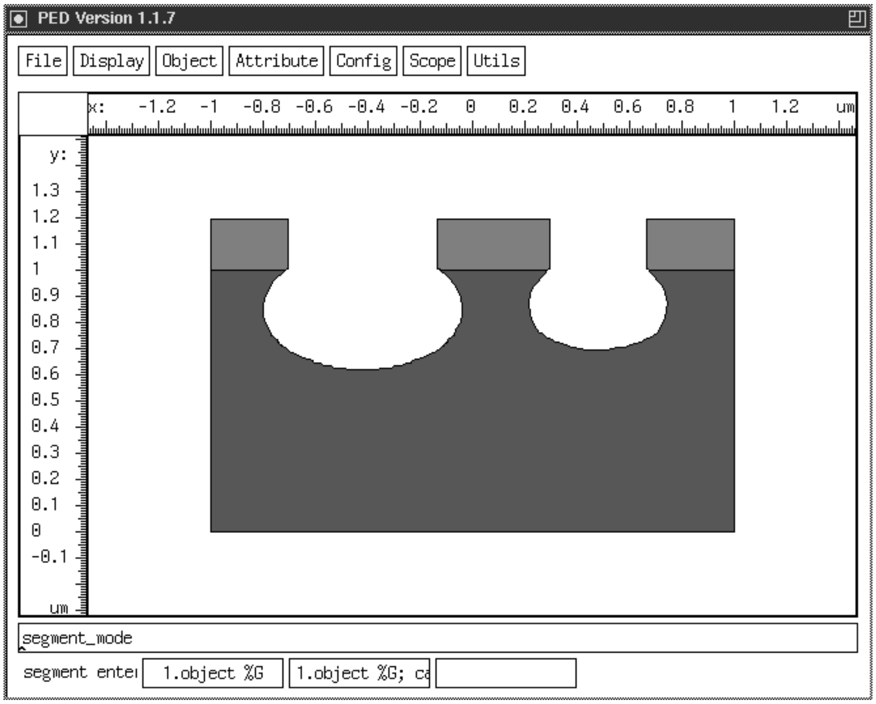

PED Version 1.1.7
File
Display
Object
Attribute
Config
Scope
Utils
x: -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1 1.2 um
y: 1.3 1.2 1.1 1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 -0.1 um
segment_mode
segment ente
1.object %G
1.object %G; ca

Figure 2: Chemical etching caused by reactive neutrals.

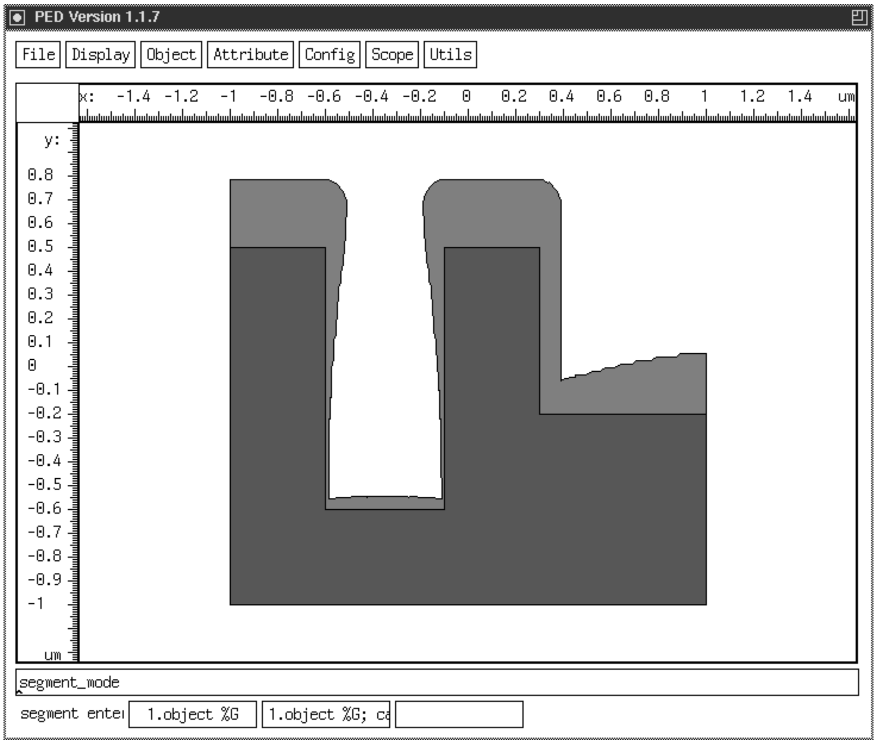

PED Version 1.1.7
File
Display
Object
Attribute
Config
Scope
Utils
x: -1.4 -1.2 -1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1 1.2 1.4 um
y: 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 -0.1 -0.2 -0.3 -0.4 -0.5 -0.6 -0.7 -0.8 -0.9 -1 um
segment_mode
segment ente
1.object %G
1.object %G; ca

Figure 3: Sputter deposition over nonplanar structure.

2 The VISTA Monte Carlo Ion Implantation Modules

2.1 Introduction

As a part of the VISTA Process Simulation Tools a program for ion implantation has been developed. The program can handle arbitrary geometries containing regions of amorphous and crystalline materials.

The miniaturization of semiconductor devices imposes an accurate description of the implantation profiles which depend on the properties of the ions (atomic number, energy, and incidence angle with respect to the wafer surface) as well as on those of the target (morphology, atomic composition, amorphous/crystalline, etc.). Thus, the consideration of special physical characteristics of crystalline structures is essential. To meet all requirements for realistic simulations general two-dimensional geometry modules and advanced physical models for crystalline targets have been developed.

2.2 Two-Dimensional Geometry Modules

The program gets its initial geometry from a PIF Binary File. The initial geometry can either be a simulation result from a previous simulation run (i.e. with the Etching and Deposition Module which is also a part of the VISTA Process Simulation Tools) or a geometry constructed with the interactive PIF Editor. The program uses the VISTA Two-dimensional Geometry Support Library to read the input geometry from a PIF Binary file and to perform the point location used for the ions. Geometry checks in two-dimensional targets are performed by using the slab method. The planar polygons are partitioned into slabs by cutting each vertex with a vertical or horizontal line. In this way several convex sections, trapezoids and triangles, are created within each slab. The advantage of our geometry routines can also be seen in the possibility of scaling the structures according to the grid spacing of the crystal.

2.3 Crystalline Targets

In the Monte Carlo method, ion trajectories are simulated collision by collision. It therefore allows the physics of ion scattering and slowing down to be taken into account in a direct way but crystalline targets with regular arrangements of atoms need advanced physical models. The atom rows or planes line up so that there are long-range open spaces through which the ions can travel without significant scattering. Ions are steered down these channels (*channeling*) by glancing collisions with the atom rows or planes, extending the final ion distribution deeper into the target. As each ion travels through the target, it undergoes a series of nuclear collisions and a fraction of its energy is transferred to a target atom which is displaced from its original position. The resulting implantation damage is calculated by the modified Kinchin-Pease model. Its influence on ion trajectories in an originally crystalline layer is considered by employing the amorphous mode with a probability proportional to the local number of displaced atoms.

2.4 Examples

These examples started from a simulation result obtained from the Etching and Deposition Module. The first simulation result (Fig. 4) shows the concentration of implanted boron into an amorphous structure. The implantation dose and the implantation energy for this case were $1 \cdot 10^{14} cm^{-2}$ and $10 keV$ respectively. Fig. 5 shows the obtained implantation profile if a crystalline target is assumed. In this example damaging is not considered. In comparison to the amorphous case the projected range is almost four times higher due to the channeling effect. If we use the modified Kinchin-Pease model to calculate the transient amorphization of the crystalline bulk the projected range decreases but the maximum penetration range remains unchanged as shown in Fig. 6.

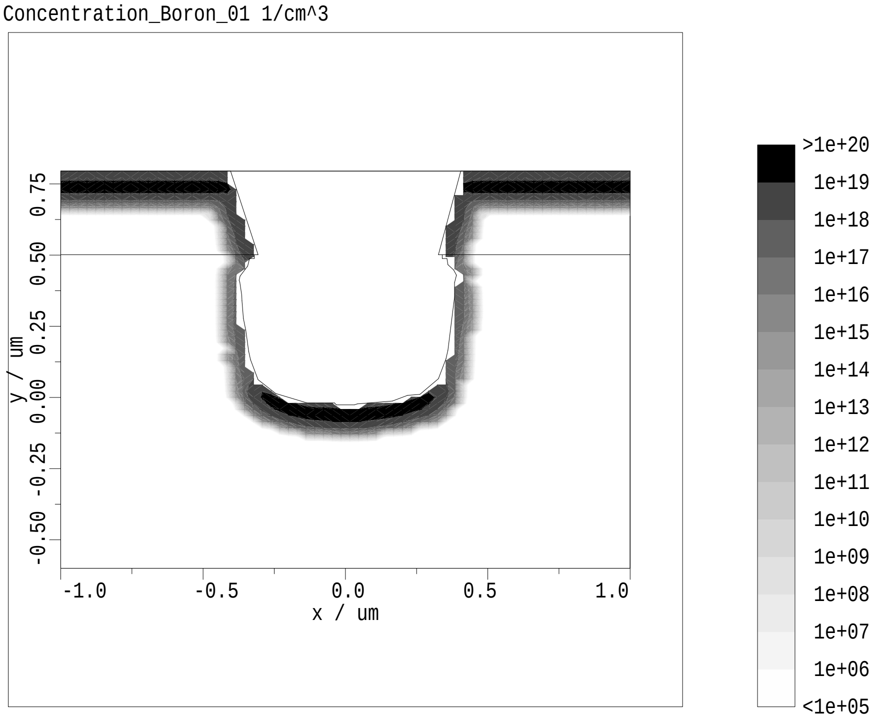


Figure 4: Simulation Result after Ion Implantation into an Amorphous Structure

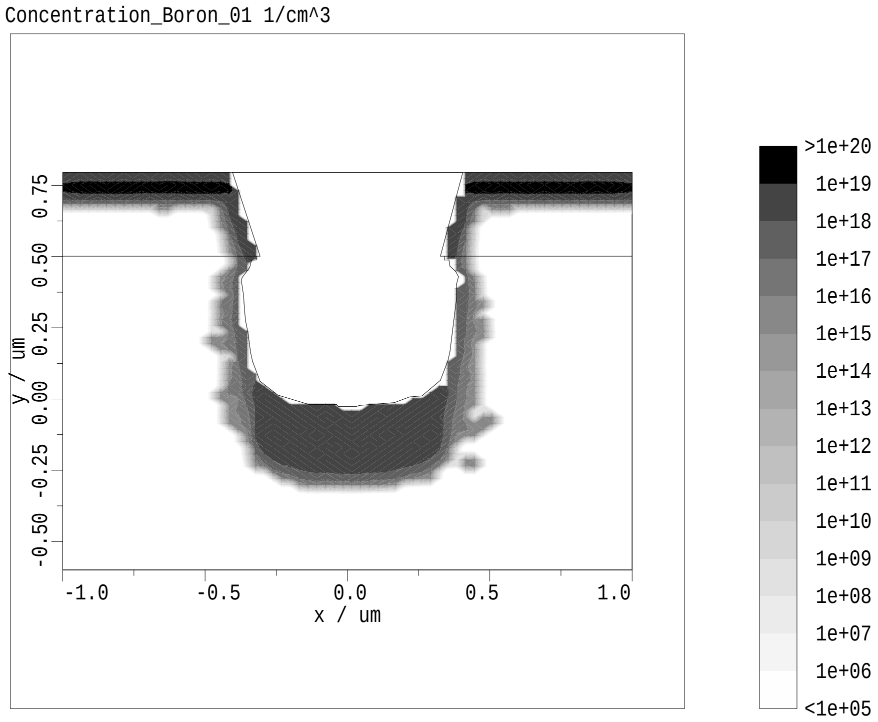


Figure 5: Simulation Result after Ion Implantation into a Crystalline Structure

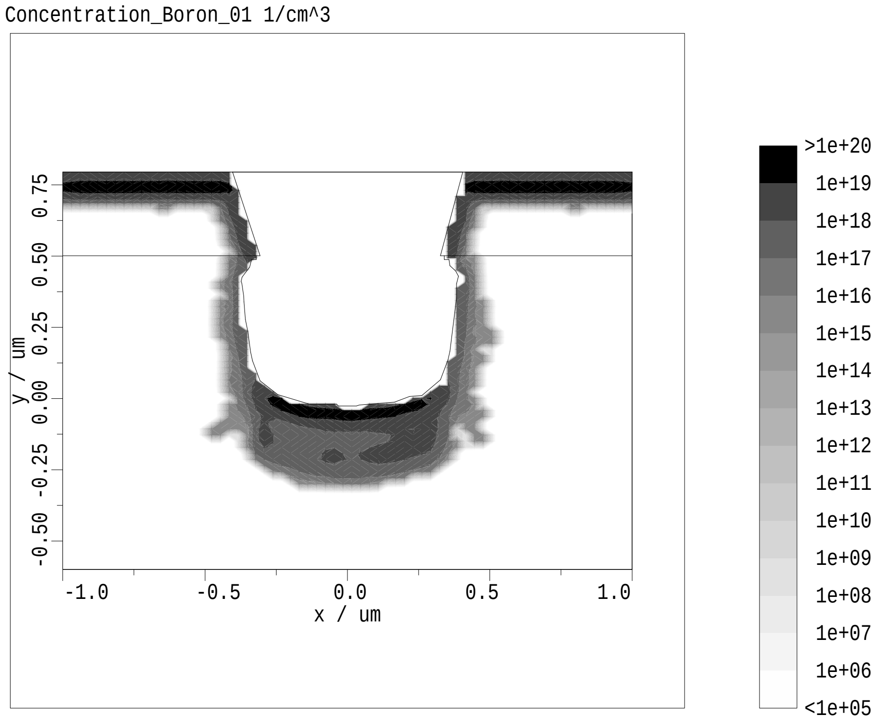


Figure 6: Simulation Result after Ion Implantation into a Crystalline Structure with Consideration of Damaging

3 The TIF Wrapper

All process simulators developed at the Technical University of Vienna can be used under the VISTA project with the Simulation Flow Controller (SFC). Since currently not all process steps can be simulated by internal simulators, there is a need to allow usage of external simulators not developed in Vienna. One early example was the sample wrapper Samwrap for SAMPLE of the University of Berkeley.

To allow a full process simulation under the SFC a binding to the TSUPREM-4 [5] process simulator of Technology Modeling Associates (TMA) was built under VISTA. The TIF (Technology Interface Format) Wrapper takes a PIF binary input file with a geometry and attributes (mostly doping profiles) defined on a triangular grid (e.g. generated by the VISTA gridding and interpolation tool VORONOI) and converts it to a TSUPREM-4 save file (used by older versions) or a TIF input file which is known by the new versions of all TMA simulation tools. The TIF file format is quite similar to the older TSUPREM-4 save format but has some enhancements in material and flow information. In addition to the PIF file an input deck is needed for simulation. This must be provided from an external source (e.g. generated by the SFC). After the simulation has finished the output is converted back into a PIF binary file which can be used for further processing within VISTA.

The TIF Wrapper uses the VISTA Material Database to convert the materials found in the Segment Descriptions of the PIF binary file to a material known by TSUPREM-4. If a material is unknown by TSUPREM-4 the TIF wrapper looks for a super material of the PIF material to match a TSUPREM-4 known material. This method can lead into problem because on back conversion the material from TSUPREM-4 is used for generating the Segment Description in the output PIF binary file and some information may be lost.

All input attributes defined on the input grid are converted only if they are known by TSUPREM-4. An internal conversion table is used for attribute type translation. This table holds the units which have fixed values under TSUPREM-4, but may be given in different units on the PIF binary files. Unknown attributes are silently discarded during conversion (they still remain in the input PIF binary file).

On output the TIF wrapper generates a PIF binary file with all information from the TSUPREM-4 output file. The geometry is converted back into a PIF geometry. The attributes are written on different grids for each segment, because doping values may be different on material boundaries. All information from a TIF file can be converted into the PIF binary file although not all information may be used by the following simulators. The VISTA visualization now supports simultaneous display of multiple attributes (e.g. all attributes of one type, but defined on different segments).

The usage of this wrapper can be seen in the simulation of a fully planarized $0.25\mu m$ CMOS Technology (see Section 5).

4 The VORONOI Re-Gridding and Interpolation Service

The VORONOI gridding service has already been described in the last report (VISTA status report, Dec. 1993). The modular architecture of VORONOI is shown in Figure 7. Compared to the status of December 1993, another module for the interpolation of attributes has been added and work on a point cloud generation and the grid coarsening module is in progress. Note that VORONOI's highly modular architecture with a common data backbone is ideally suited for extensions (see Figure 7).

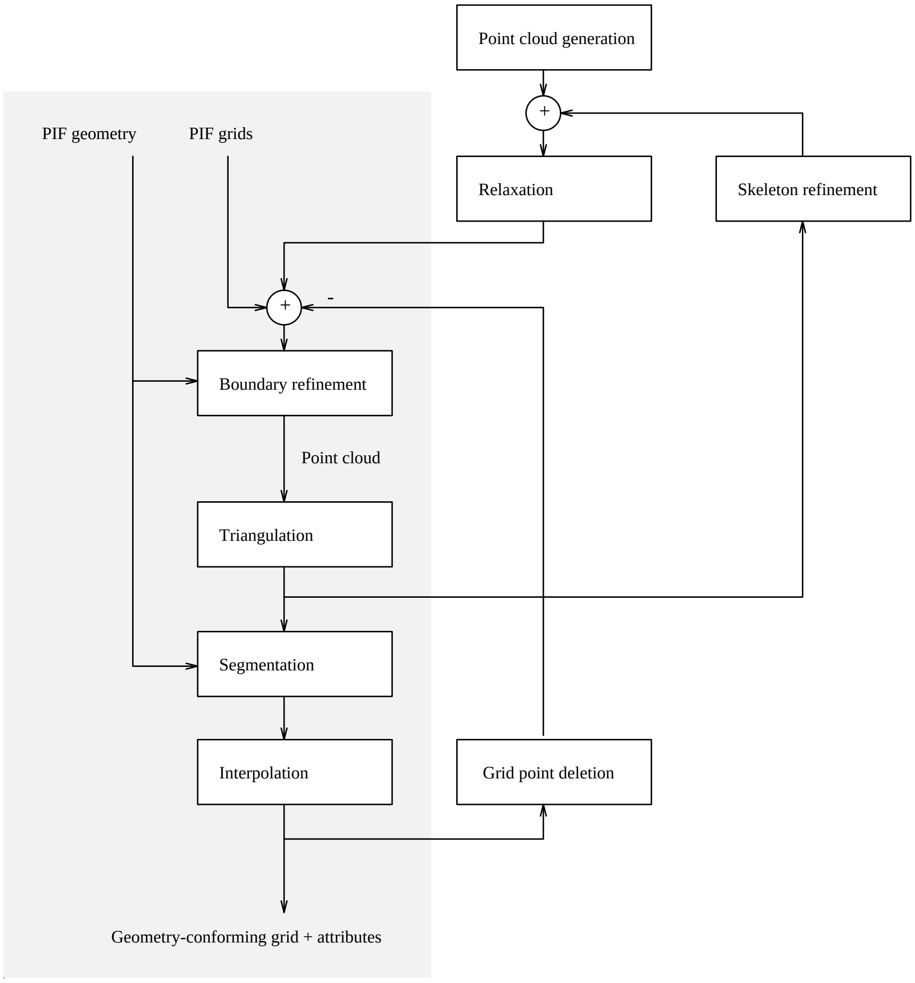


Figure 7: Modules and data flow within VORONOI. The shaded area represents the present status of VORONOI, other modules are planned extensions.

4.1 Interpolation

The need for interpolation of attributes arises for two different reasons. Firstly, input attributes may be given on entirely different, but spatially overlapping grids which have been merged and re-triangulated to form a single grid for the overlap region. Secondly, additional grid points may have been created as a result boundary refinement, which do hence not carry attribute information.

In any case, one ends up with a single Delaunay grid for a given region, where attribute values are defined for certain points and are not defined for all others. The percentage of undefined values can be rather large (90% are no exception), depending on the spatial density and number of different grids that have been merged.

VISTA offers grid-type specific interpolation services which are accessed via a generic opaque interface (*GRid Support*, `GRS`) so that the application does not have to bother with any details of the interpolation. These interpolation methods are well-suited for situations where the attribute information that is requested by an application is defined on a grid which is unknown or which can not be handled by the application. By means of the generic interpolation service the application can simply interpolate all the input attributes onto a preferred (internal) grid.

From a rigorous point of view, finite-difference results should not be interpolated at all, as the values of a quantity are only defined on the grid points. From previous considerations it is, however, obvious that an interpolation method is needed for coupling simulation tools, also when no element-specific interpolation function which is consistent with the application's discretization method is given.

VORONOI is in the advantageous position to utilize target grid information for this interpolation problem. The target (Delaunay) grid is a "super-grid" of the input grid (the input grid is a "sub-grid" of the Delaunay grid) and the output grid. This means that the Delaunay grid contains (nearly) only grid points that are also grid points of the source grids. For those points which are *not contained* in the source grid on which an attribute to be interpolated is defined, some means to obtain "reasonable" values is required (these are the *unknown* values).

A natural initial approach for interpolation is to solve the Laplace equation

$$\triangle\psi = 0 \tag{1}$$

with the "internal Dirichlet boundary condition"

$$\psi_j = \psi_j^D \tag{2}$$

for all source grid points j where the attribute value ψ is known, and the Neumann boundary condition

$$grad\psi_j = 0 \tag{3}$$

for all other (non-Dirichlet) boundary points.

When only boundary values are known and all of the inner grid points are unknown, the problem is the classical *Dirichlet problem.* In our case, however, there are known values on inner grid points as well. Let the grid contain N grid points comprised by N_U unknown and N_D known

(Dirichlet) values, $N = N_U + N_D$. The discretization of Equation 1 will then lead to to a system of N_U equations for the unknown ψ_i.

The Delaunay graph and the Voronoi polygons can be readily used to discretize Equation 1. With a finite-difference approximation and a box integration over the Voronoi regions, one directly obtains a linear system for the N_U unknown values ψ_j. The Neumann boundary conditions are fulfilled implicitly by the box integration. The Dirichlet values cause a non-zero right-hand side of the linear system.

As Equation 1 is generally not fulfilled at the Dirichlet points (the source grid points), spikes may form at the source grid points where $\triangle\psi \neq 0$ originally.

An alternative approach is to solve the biharmonic equation

$$\triangle\triangle\psi = 0 \tag{4}$$

(with analogous boundary conditions) to achieve a "higher order" more smooth interpolation scheme.

Figure 8 shows a result of the application of VORONOI to a geometry which has the shape of the VISTA logo. Only three inner gridpoints labeled A, B, and C are given to define an attribute in Figure 8. The specified attribute values are $\psi(A) = 0.0$, $\psi(B) = 1.0$, and $\psi(C) = 0.0$, all other values have been interpolated. For this example, VORONOI performed both triangulation and interpolation (by solving the biharmonic equation), which led to the solution shown in Figure 8 (the shaded quasi-three-dimensional surface) along with the initial geometry.

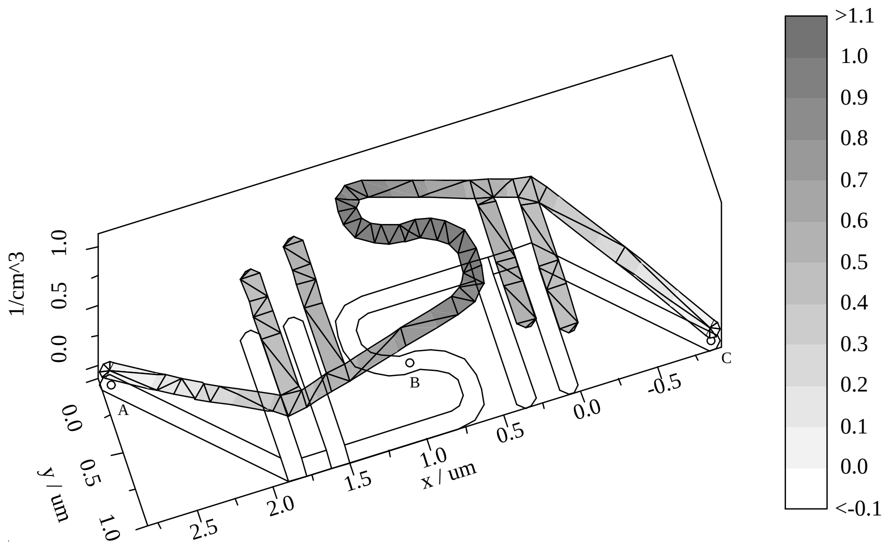


Figure 8: Application of VORONOI to the complex singly connected geometry of the VISTA logo. Attribute values have only been defined on the three inner grid points A, B, C. All other grid points are boundary points with interpolated values.

The example shown in Figure 8 exceeds the complexity of current TCAD problems, but it clearly proofs the ability of VORONOI to construct a smooth ("physically reasonable") interpolation in very complex geometries from a minimum of information required (in this case just three points). VORONOI processes this example in 1.8 CPU seconds (including PIF output) on a PC 486 (DX2, 66Mhz) running Linux 1.0.

4.2 Grid Coarsening

The need for coarsening an existing grid arises due to repeated grid merge operations. The number of grid points in the wafer state grid may grow considerably so that the removal of grid points becomes necessary. The coarsening of the grid is accomplished by measuring the quality of the final triangular grid, by removing appropriate uncritical grid points from the point cloud ("Grid point deletion" in Figure 7) and by re-triangulating the remaining grid points.

5 Simulation of a Fully Planarized, $0.25\mu m$ CMOS Technology

Motivation and technical details for the fully planarized $0.25\mu m$ CMOS technology with shallow trench isolation are given in [6]. A schema of the the resulting device structure is shown in Figure 9.

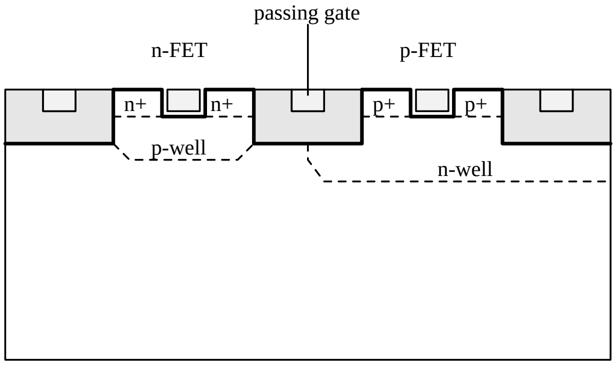


Figure 9: Final CMOS structure after junction formation

From a simulation perspective, this technology is highly demanding with regard to the integration of topography and doping definition steps. The challenge lies in the alternating etching/deposition and implantation/diffusion steps. For each step, a specialized process simulation tool must be used and the wafer state must be kept consistent, although each of these specialized tools focuses on a certain aspect and neglects others.

The simulators Etch [7], TSUPREM-4 [5], the Monte Carlo module of Promis for the simulation of ion implantation [8] and the auxiliary tool Sketch for modeling spin-on, exposure, and mask strip steps are used to simulate the process.

This example process also demonstrates the use of VORONOI within the VISTA framework. The simulation flow employs three entirely different process simulation tools and VORONOI is used repeatedly to achieve consistency between grid and geometry, and to transform the structural and doping information between tool-specific and wafer state compliant form.

Figure 10 shows the Phosphorus doping of the n-well after the trenches for the MOS gates and for the passing gates have been etched. This etch step removes part of the geometry and the grid and doping is “repaired” by VORONOI to produce a consistent wafer state.

The Promis Monte Carlo module always writes its result on a non-boundary-conforming tensor product grid, which extends over all segments of the geometry. TSUPREM-4 expects a geometry-conforming grid, where the geometry is actually defined over grid points. Etch changes just the geometry without taking care for the grids and attributes which are defined over the geometry. The auxiliary tool Sketch also adds segments to the geometry, without considering the grid and attribute information.

Hence, VORONOI is called after each step which modifies (only) the geometry without consistently adapting the grid and after each step that adds a new grid to the current wafer state. VORONOI needs to be run theoretically after the following steps: 13, 18, 19, 20, 24, 32, 37, and

step	simulator	SFC operation	parameters
1	Sketch	create-subdomain	”Si”, $5.6 \times 2.5\mu m$
2	Sketch	spin-on	”Resist”, $0.2\mu m$
3	Sketch	expose	
4	Sketch	strip-material	”Exposed”
5	Etch	plasma-etch	misc. etch rates, 320 seconds
6	Sketch	strip-material	”Resist”
7	Etch	iso-depo	”SiO2”, 60 seconds at $125 nm/s$
8	Sketch	spin-on	”Resist”, $0.25\mu m$
9	Etch	iso-etch	misc. etch rates, 330 seconds
10	Sketch	spin-on	”Si3N4”[1], $1\mu m$
11	Sketch	expose	n-well mask
12	Sketch	strip-material	”Exposed”
13	Promis	monte-carlo-implant	”Phosphorus”, $5 \cdot 10^{11} 1/cm^2, 60keV$
14	Sketch	strip-material	”Si3N4”
15	Sketch	spin-on	”Si3N4”[2], $1\mu m$
16	Sketch	expose	p-well mask
17	Sketch	strip-material	”Exposed”
18	Promis	monte-carlo-implant	”Boron”, $5 \cdot 10^{11} 1/cm^2, 60keV$
19	Sketch	strip-material	”Si3N4”
20	TSUPREM-4	diffusion	inert, 3.5 minutes at 1000°C
21	Sketch	spin-on	”Resist” $0.5\mu m$
22	Sketch	expose	
23	Sketch	strip-material	”Exposed”
24	Etch	plasma-etch	misc. etch rates, 200 seconds
25	Sketch	strip-material	”Resist”
26	Etch	iso-depo	”SiO2”, 60 seconds at $1.1 nm/s$ [3]
27	Sketch	spin-on	”Poly”, $0.25\mu m$ [4]
28	Etch	iso-etch	misc. etch rates, 310 seconds
29	Sketch	spin-on	”Si3N4”, $1\mu m$
30	Sketch	expose	nMOS SD mask
31	Sketch	strip-material	”Exposed”
32	Promis	monte-carlo-implant	”Arsenic”, $2.5 \cdot 10^{14} 1/cm^2, 33keV$
33	Sketch	strip-material	”Si3N4”
34	Sketch	spin-on	”Si3N4”, $1\mu m$
35	Sketch	expose	pMOS SD mask
36	Sketch	strip-material	”Exposed”
37	Promis	monte-carlo-implant	”Boron”, $2.5 \cdot 10^{14} 1/cm^2, 30keV$
38	Sketch	strip-material	”Si3N4”
39	TSUPREM-4	diffusion	inert, 1 minute at 970°C

Table 1: Process simulation flow for the $0.25\mu m$, fully planarized, shallow trench isolated CMOS process.

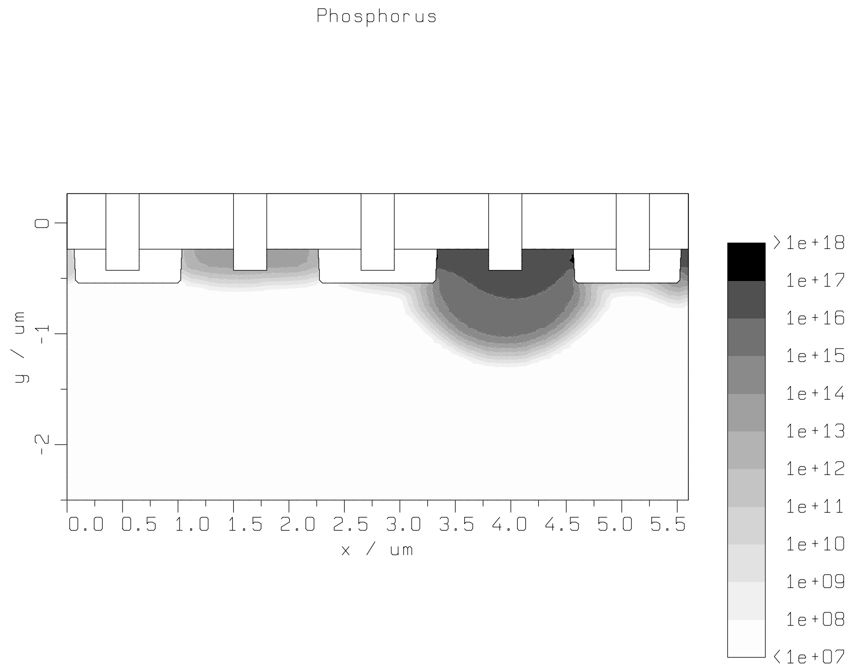
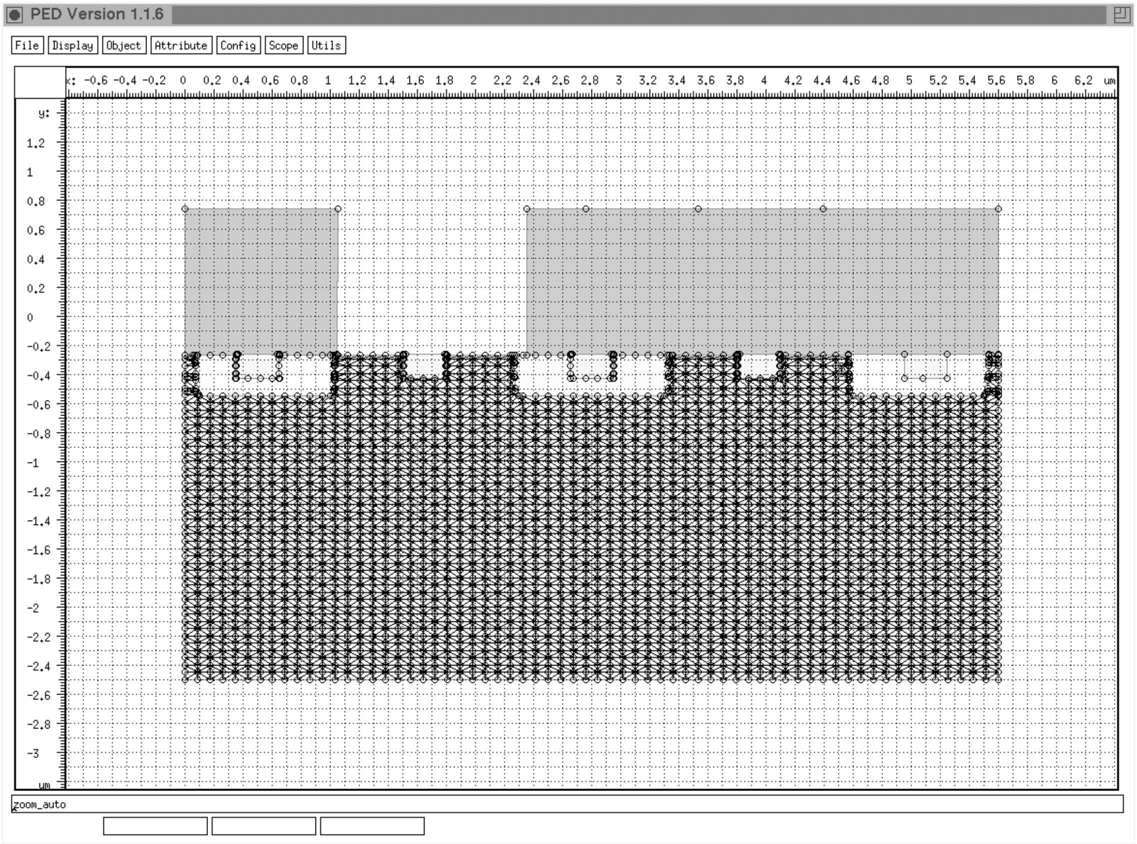


Figure 10: N-well doping profile after etching the trench for the planarized poly gate. Result of step 24

38. In the practical application, it is run even more often in order to maintain a safe wafer state. Compared to the effort of the actual simulation steps, the required CPU time is still negligible.

Note that the diffused doping profiles produced by TSUPREM-4 (Figure 10) show a slight distortion in (orthogonal) X and Y directions. This effect is due to the numerically enhanced diffusion caused by the tensor-product constellation of the triangulated grid points. An initial adaption of grid points (as depicted in Figure 7) will help to bypass this tool-specific problem.


PED Version 1.1.6
File
Display
Object
Attribute
Config
Scope
Utils
x: -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2 2.2 2.4 2.6 2.8 3 3.2 3.4 3.6 3.8 4 4.2 4.4 4.6 4.8 5 5.2 5.4 5.6 5.8 6 6.2 um
y:
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
-2.4
-2.6
-2.8
-3
um

zoom_auto

Figure 11: PED showing the entire CMOS structure with the grid on the silicon segment, triangulated by VORONOI. Result of step 31

6 The PIF Visualization

The PIF visualization is a comfortable tool to display two dimensional data stored onto a PIF data file. This two dimensional data, so called PIF-attributes, are sets of data values of a function of two independent variables like doping profiles or potential distributions valid for a bounded 2D domain (s. Fig. 12).

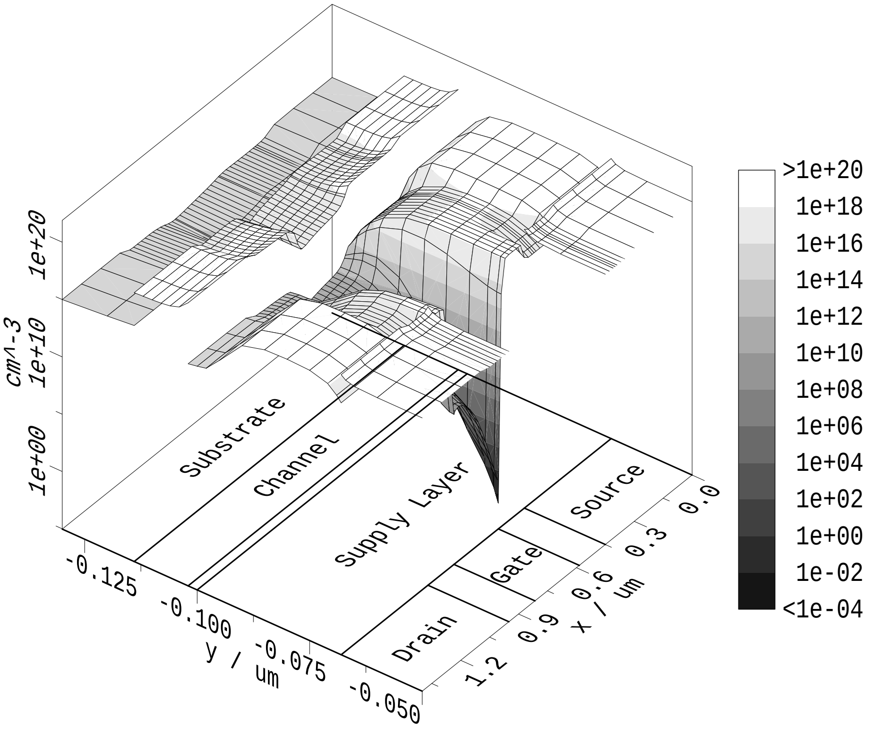


Figure 12: View of an electron concentration of a hetero structure device in logarithmic mode with slices and a grey scale. It can easily be seen, that for each segment a particular attribute is specified.

The SVG data format is used as a general ASCII data format to store a specific view, from which postscript data or xfig data can be obtained using the VISTA data converters *svg2ps* and *svg2fig*. xpif2d provides an interactive mode by the standard Vista User Interface as well as a batch mode which is useful for automatic data conversions.

6.1 Data Selection - The *File* Menu

The *File* menu provides to select data, to save data and to terminate xpif2d.

The *open* entry allows the user to select an arbitrary PIF file by the standard VISTA PIF file selection menu. After the correct logical PIF is chosen, an attribute selection menu is shown, which lists all attributes available in the PIF in the left column. The attribute type (e.g. *Concentration* or *Potential*) as well as the attribute name can be preselected where a star '*' as simple wildcard feature is provided (e.g. **_C*n_C**). When the preselected attribute type or name is changed a rescan must be done by pressing the RESCAN button or simply pressing the ENTER key on the keyboard when the string is changed.

Clicking on an item in the left list marks it as 'selected' and it will appear in the right list as well. Any further different attribute selected in the left list is added to the right list. The right list contains the so called attribute set, which is displayed when the selection is confirmed by the OK button. An previously selected item of the attribute set can be removed if it is clicked on in the right list. The item then disappears from the list and can be selected again by clicking on it in the left list.

Only attributes of identical type are displayed, e.g. electron and hole concentrations or attributes defined on a segment set. As different attributes can be chosen, abrupt value changes can be displayed. This makes it possible to visualize results of simulating e.g. hetero structure devices (s. Fig. 12).

By the *attribute* entry a previously attribute set can be modified.

The current view can be saved using the *save* entry. The data are copied onto a SVG file with the name of the first attribute of the attribute set extended by `.svg`. An already existing SVG file with identical name is lost.

6.2 Data Mode - The *Scale* Menu

As attributes of different type must be displayed, several data modes are available in order to give a maximum on information. The data modes are *absolute*, *negative*, *logarithmic* and *square*. The *square* data mode scales the domain of the independent variables so it is viewed as a square. This is especially useful if the domain is a small slice. By selecting the *limits* entry a limits adjust menu is popped up (see *Domain Selection* for further details).

6.3 View - The *Viewpoint* Menu

Five predefined viewing directions are available. Four from above the attributes with the directions from southeast, southwest, northeast and northwest and the negative z-direction (top view, s. Fig. 13). Front, back and two side views are provided as well which can be used as an, albeit awkward, method for generating horizontal or vertical one dimensional cuttings of attributes defined on ortho grids. At first a small horizontal or vertical slice including one grid line must be chosen and than the front/back or side view must be selected. If the slice is not small enough, the result is strip rather than a one dimensional curve. If no grid line is inside the slice, nothing is displayed.

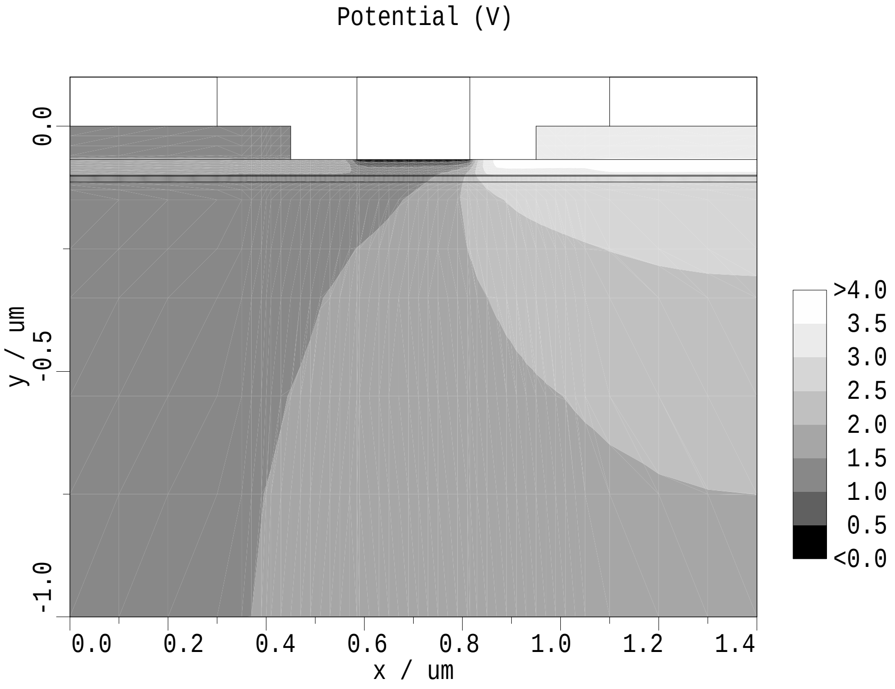


Figure 13: Top view of a potential distribution.

6.4 Display Mode - The *Color* Menu

For additional information the grid, the geometry and, for vector attribute types, e.g. current density or electric field, arrows can be shown.

To obtain an efficient data representation xpif2d provides several color modes as *color slices*, *color isolines* and *front/back shading*. The *color slices* and *color isolines* modes also show a color scale.

For the color scale rainbow colors, the colors of hot iron or a grey scale can be chosen. The grey scale is useful when a black and white postscript printer is used as output device as it has been done for this document.

To adjust the color scale for a particular screen, a hue correction and a gamma correction are provided. Furthermore the color ordering can be reversed.

6.5 Domain Selection

With a crosshair cursor running in the xy-plane (xy is a placeholder for the 2 independent variables which the attribute to view depends on) and the crosshair axes parallel to the xy-axes the domain can be selected by click and drag. When the left mouse button is pressed the actual position of the cursor indicates the first corner of the desired domain. Then the mouse is moved with the left button still pressed a selection window is drawn. Finally, when the left mouse button is released the actual cursor position indicates the second corner of the new domain. If a new domain is selected the attribute is cut and displayed using the full display area. So arbitrary details can be viewed.

If the left mouse button is pressed and released immediately, so that no section window is drawn, a limits adjust menu is popped up where the ranges for each coordinate can be independently set, inclusive the z-value range and the number of decades for the logarithmical mode. This menu can also be selected by the last entry of the *Scale* pull down menu.

Is the domain selected by command line, than this is the largest selectable domain and is the upper bound for all interactively specified subdomains. Is the crosshair cursor inside the domain, the current position and the value of the attribute is displayed in the upper left corner of the display area. If the attribute is a vector type, the value of the first component is shown. If there is one attribute above an other attribute, the value of the first attribute found in the attribute set is shown.

7 The VISTA Simulation Flow Control Module

7.1 Overview

The introduction of the VISTA Simulation Flow Control Module (SFC) (cf. VISTA status report December 1993) has resulted in considerable progress with respect to tool integration, automatic process flow simulation, and simulation data management. It is now possible to use tools like `SAMPLE`, `TSUPREM`, `PROMIS` interchangeably without having to take special care for any tool control or tool data exchange details. Simulation results are stored together with the recipes used to generate them and are easily retrievable upon request. An automatic split detection finds split points in related simulation tasks and avoids unnecessary recomputations.

7.2 Graphic Simulation Flow Editor

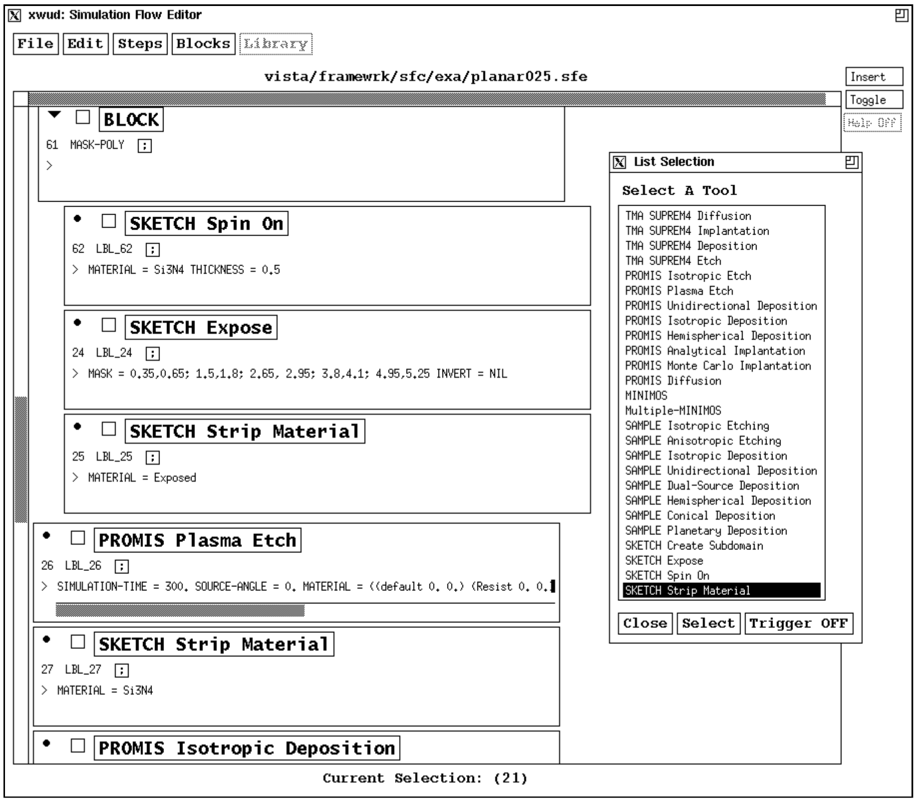
During the course of the last year, the migration from a text-based flow representation to a graphic-oriented one was completed. Figure 14 shows the definition of the 0.25 μm planar CMOS process using the Simulation Flow Editor. Note that all available tools may be used in any order; in this example, `TSUPREM-4` was used together with the `PROMIS` and `SKETCH` modules for simulating the two highly non-planar transistors. (As a matter of fact, a planar CMOS process results in the device's silicon segment cross-section – cf. figure 9 – being planar not at all).

7.3 Split Point Detection

To avoid recomputation of data already available, an automatic split detection mechanism has been integrated that starts the execution of a flow at the first statement that differs either in its type or in its arguments from the corresponding statement of a previously computed flow. Iterative trial-and-rework loops most widely observed during the early stages of process design are thus effectively supported. Figure 15 gives an impression of how split points are detected and split branches are generated during experimenting with a flow design. Note that branches are computed in parallel whenever possible.

7.4 Persistent Run Data Management

To support the analysis of data gathered from a large number of computations over a long period of time, especially when exploring the limits of the simulation tools involved, it is deemed to be of utmost importance to be able to retrieve results, logging information, and diagnostic messages generated by a tool call at any later time. A run data management system was devised to properly collect, store and present these data to the user. In essence, a database entry is generated for each simulation step which can be read by simply selecting the step symbol (cf. figure 15). As all data are written to the file system, they stay available until actively removed.



vista/framewrk/sfc/exa/planar025.sfe

BLOCK
61 MASK-POLY

SKETCH Spin On
62 LBL_62
> MATERIAL = Si3N4 THICKNESS = 0.5

SKETCH Expose
24 LBL_24
> MASK = 0.35,0.65; 1.5,1.8; 2.65, 2.95; 3.8,4.1; 4.95,5.25 INVERT = NIL

SKETCH Strip Material
25 LBL_25
> MATERIAL = Exposed

PROMIS Plasma Etch
26 LBL_26
> SIMULATION-TIME = 300. SOURCE-ANGLE = 0. MATERIAL = ((default 0. 0.) (Resist 0. 0.

SKETCH Strip Material
27 LBL_27
> MATERIAL = Si3N4

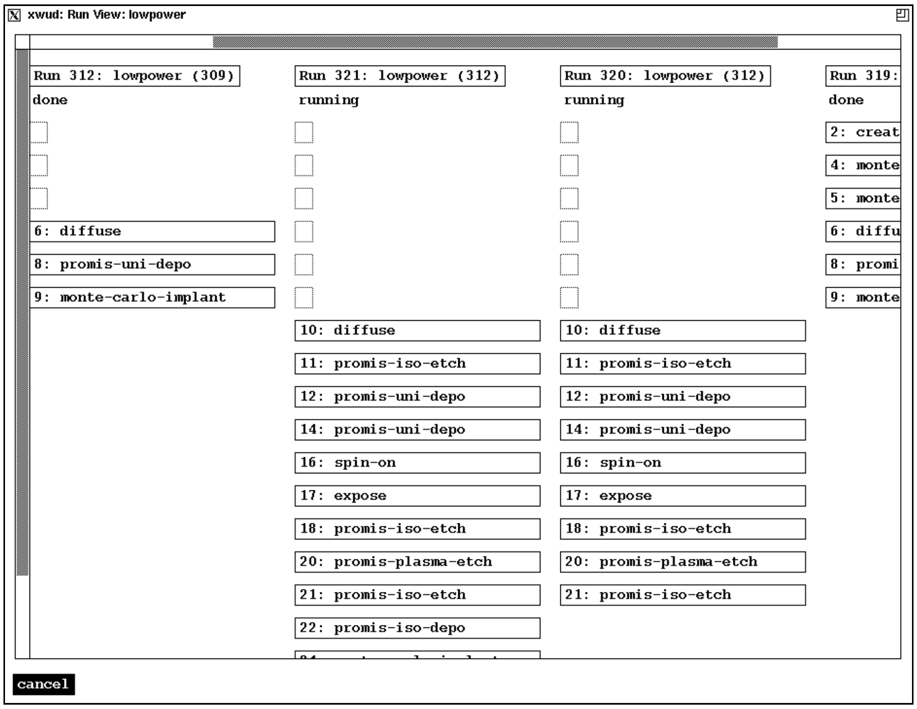
PROMIS Isotropic Deposition

Select A Tool

- TMA SUPREM4 Diffusion
- TMA SUPREM4 Implantation
- TMA SUPREM4 Deposition
- TMA SUPREM4 Etch
- PROMIS Isotropic Etch
- PROMIS Plasma Etch
- PROMIS Unidirectional Deposition
- PROMIS Isotropic Deposition
- PROMIS Hemispherical Deposition
- PROMIS Analytical Implantation
- PROMIS Monte Carlo Implantation
- PROMIS Diffusion
- MINIMOS
- Multiple-MINIMOS
- SAMPLE Isotropic Etching
- SAMPLE Anisotropic Etching
- SAMPLE Isotropic Deposition
- SAMPLE Unidirectional Deposition
- SAMPLE Dual-Source Deposition
- SAMPLE Hemispherical Deposition
- SAMPLE Conical Deposition
- SAMPLE Planetary Deposition
- SKETCH Create Subdomain
- SKETCH Expose
- SKETCH Spin On
- SKETCH Strip Material

Current Selection: (21)

Figure 14: Simulation Flow Editor



Run 312: lowpower (309)
done
6: diffuse
8: promis-uni-depo
9: monte-carlo-implant

Run 321: lowpower (312)
running
10: diffuse
11: promis-iso-etch
12: promis-uni-depo
14: promis-uni-depo
16: spin-on
17: expose
18: promis-iso-etch
20: promis-plasma-etch
21: promis-iso-etch
22: promis-iso-depo

Run 320: lowpower (312)
running
10: diffuse
11: promis-iso-etch
12: promis-uni-depo
14: promis-uni-depo
16: spin-on
17: expose
18: promis-iso-etch
20: promis-plasma-etch
21: promis-iso-etch

Run 319:
done
2: creat
4: monte
5: monte
6: diffu
8: promi
9: monte

Figure 15: Split Branches

8 Byte Code Compiler for XLISP

The XLISP interpreter is a modified version of the LISP interpreter written by D. Betz [9]. There were many enhancement and new functions added to the small original interpreter as they were needed for the project. Due to the easy prototyping major parts of VISTA are written in LISP code (e.g. the Simulation Flow Controller, the PIF file browser and many User Interaction Panels). Due to the huge amount of LISP code there are very many LISP nodes which have to be checked during the garbage collection which took about 80% of the used time in the interpreter. To lower this time and reduce the number of LISP nodes, a byte code compiler was written.

The byte code compiler reduces significantly the number of LISP nodes used by compiled functions because most parts of the code are compiled into a byte code string. Each byte of this string defines an opcode of the byte code interpreter and has a similar LISP interpretion. E.g. the LISP function *(car ...)* can be compiled to the opcode *OP_CAR*. Many of the standard LISP functions have associated byte code opcode. Some of the more complex functions are implemented by a sequence of instructions as shown below.

In the example below the LISP code

```
(member val list :test #'equal)
```

is compiled into the byte code

```
#y3(#h23(0925002501650015A23800139110A53800041108130064) val list)
```

The symbolic byte code output in this case is:

```
0:        BC::ENTRY
0:    25 00         (op-sgetv val)       ; get value of variable val
2:    25 01         (op-sgetv list)      ; get value of variable list
4:        LABEL1
4:    65 00 15      (op-member LABEL3)   ; special operation for member
7:    a2            (op-equal)           ; compiled in :test
8:    38 00 13      (op-brt LABEL2)      ; found match
11:   91            (op-cdr)             ; next element of list
12:   10            (op-dup)
13:   a5            (op-listp)           ; check for dotted pair at end of list
14:   38 00 04      (op-brt LABEL1)
17:   11            (op-pop)             ; no match found
18:   08            (op-nil)             ; return NIL
19:       LABEL2
19:   13 00         (op-clap 0)          ; remove val from stack
21:       LABEL3
21:       BC::EXIT
21:   64            (op-bcend)           ; end of byte code
```

One big advantage of byte coded `LISP` in addition of the reduced number of `LISP` nodes is that the number of `LISP` frames for function calls is much smaller too. Since the binding of the `LISP` parameters is the second CPU expensive operation in the `LISP` interpreter. This speeds up the execution of the programs. In the example you can see that the function call for the :test parameter of assoc is byte compiled directly in, so no additional byte code is required for the test function. This also mean, that there will be no `LISP stack` frame for each function call to the test function. This also works form *#'(lambda ...)* test functions which compile into a *(let ...)* for the parameter. This optimization is valid, since the function is defined and used in the environment of the member call.

Currently the byte code compiler generates byte code which evaluates only a small amount faster. This is because no optimization is done on the generated byte code. Later versions will analyze data flow and may remove variable binding from *(let ...)* statements if possible. In addition many protections of values can be removed if they are not used (e.g. a *(dolist ...)* allows a *(return ...)* in it and needs special code for this. If this feature is not used, some byte code can be removed. This will speed up the interpretion of byte code compiled code by an additional factor 2 to 3.

Since the byte code compiler analyzes all `LISP` code it compiles it is able to find some kinds of coding error which would normally only occur at runtime, e.g. too many parameter to `(car ...)`.

Byte code compiled `LISP` code and non compiled code can be intermixed at any level for functions. All macros have to be known at compile time because of parameter evaluation. Any unknown call is implicitly taken as function call and will produce later an error if it is defined as macro. Another difference is that during the evaluation of byte code opcodes no tracing is available (still for the byte coded function).

The byte code compiler was tested with `vmake` (Vienna Make Utility) which is coded in `LISP`. Also the number of garbage collection is a little bit higher, the total numbers of checked nodes is much smaller since the byte code compiled utility needs only about 60% of the nodes of the uncompiled version. The reduction is not higher because the utility used big data lists.

References

[1] I.A. Blech. Evaporated Film Profiles Over Steps in Substrates. *Thin Solid Films*, 6:113–118, 1970.

[2] C.R. Giardina and E.R. Dougherty. *Morphological Methods in Image and Signal Processing.* Prentice-Hall, New Jersey, 1988.

[3] E. Strasser, K. Wimmer, and S. Selberherr. A New Method for Simulation of Etching and Deposition Processes. In *1993 International Workshop on VLSI Process and Device Modeling*, pp 54–55, 1993.

[4] E. Strasser, G. Schrom, K. Wimmer, and S. Selberherr. Accurate Simulation of Pattern Transfer Processes Using Minkowski Operations. *IEICE Transactions on Electronics*, E77-C:92–97, 1994.

[5] Technology Modeling Associates, Inc. *TMA TSUPREM-4, Two-Dimensional Process Simulation Program Version 6*, 1993.

[6] D.S.Wen, W.H. Chang, Y. Lii, A.C. Megdanis, P. McFarland, and G.Bronner. A Fully Planarized $0.25\mu m$ CMOS Technology. In *Proc.Symposium on VLSI Technology*, pp 83–84, 1991.

[7] E. Strasser and S. Selberherr. A General Simulation Method for Etching and Deposition Processes. In S. Selberherr, H. Stippel, and E. Strasser, editors, *Simulation of Semiconductor Devices and Processes*, volume 5, pp 357–360. Springer, 1993.

[8] H. Stippel. Simulation der Ionen-Implantation. *Dissertation, Technische Universität Wien*, 1993.

[9] D.M. Betz. *XLISP: An Object-Oriented Lisp, Version 2.1*, 1989.