George M. Koppelman
Michael A. Wesley

OYSTER: A Study of Integrated Circuits as Three-Dimensional Structures

This paper presents a design for a software system (OYSTER) for the parametric simulation and analysis of the fabrication steps of very large scale integrated circuit devices. The system is based on a solid geometric modeling approach in which the component parts of an integrated circuit are represented at any step as three-dimensional solid objects in a geometric data base. The simulation of a fabrication step transforms the data base representation of the geometry and the relations among component parts from their state before the step to their state after the step. At any step, and particularly after the final step, the component parts may be analyzed automatically to determine geometric, mechanical, thermal, and electrical properties. Statistical effects may be incorporated to allow investigation of alignment tolerance build-up and yield. A prototype study is described in which an existing geometric modeling system is used to transform a set of planar masks for an FET device through 28 process steps into 3-D models which are used to compute device capacitances.

Introduction

The design of VLSI structures may be considered at several levels—individual device, chip, wafer, module, etc. In this paper we describe a design for a tool to be used in the design and analysis of individual VLSI devices. The VLSI device design process may be defined as follows:

Given a set of desired device characteristics (size, shape, electrical, mechanical, statistical, etc.), define a set of parameterized (time, temperature, concentration, etc.) process steps to produce such a device.

In the past the execution of such design tasks has been based on the skill and prior knowledge of the designer, using only rather simple design tools. However, advances in VLSI device fabrication technology, both in lithography and in device complexity, now make it even more difficult for designers of experimental circuits to

1. Understand how the physical shapes of the various material layers are actually formed in relation to each other and, as a consequence, to
2. Predict the electrical relationships between the levels that determine the operating characteristics and performance of the devices,
3. Analyze the thermal and mechanical properties of a device,
4. Analyze the effects of tolerances on performance and yield.

Although designers have commonly defined devices in essentially two dimensions by specifying a sequence of process steps and drawing a set of two-dimensional planar mask patterns, the devices themselves must be thought of as three-dimensional (3-D) objects. This is because, in an integrated circuit process, each new material layer must conform to the ones already deposited, taking into account their various thicknesses, discontinuities, and overlappings. Typical processes for even simple devices have required sequences of 25 or more process steps. The resulting material layers have become very difficult for the designer to visualize, let alone analyze, without computerized tools.

These complexities have made it desirable to try to extend Computer Aided Design (CAD) techniques to the study of the 3-D structure and properties of integrated circuit devices. In this paper we describe a system (OYSTER) which provides the device designer with the means of simulating

close approximations to the 3-D shapes that result from a sequence of actual process steps combined with a planar mask set. OYSTER provides a software simulation tool to answer questions of the form:

Given a sequence of parameterized process steps and planar masks, what device structures, operating characteristics, and performance will be produced?

This tool may then be embedded in an iterative system for VLSI design optimization.

It appears that the timing is right for OYSTER. The literature contains many references to work aimed at deep understanding of the physics of individual fabrication steps in VLSI processes [1–4]. Much work has also been done on the derivation of operating characteristics and performance of known device structures [5–7]. In a totally different domain, namely mechanical engineering, solid geometric modeling has become accepted as an interactive design and analysis tool [8–10]. Solid modeling has also been the basis for much work on application algorithms for manipulating models in domains such as robot programming and machine tool control. The OYSTER system described here is aimed at integrating these separate approaches to the study of VLSI technology into a single tool for simulation and analysis of complete VLSI device designs. We think that the general concepts of solid geometric modeling are applicable also to the other levels of VLSI design.

In the following section we describe the OYSTER system. Then a feasibility study is described and illustrated in which the Geometric Design Processor (GDP) [9] was used to simulate and analyze 28 fabrication steps of an FET process. Finally, areas of future work on OYSTER are discussed.

The OYSTER system

The software simulation tool OYSTER (*O*ff-line *Y*orktown *S*ystem for *T*hree-dimensional *E*mulation of VLSI *R*esearch) is intended to aid in the design and analysis of complex VLSI devices. OYSTER has two main components:

1. A fabrication step simulator that allows representation of the geometric effects of parameterized physical processes by means of
   a. 3-D geometric objects in a geometric data base,
   b. Relationships among these geometric objects, and
   c. Parameterized procedures that simulate process steps by performing transformations on the objects and relationships contained in the geometric data base.

   It is expected that these operations will be performed interactively, i.e., in essentially real time.
2. A device analyzer that uses the geometric data base to generate input to analysis procedures to derive properties of the device. Many of these operations will be run interactively, but some of the more complex ones will have to be run in batch mode and the user will experience significant delays before seeing the results of an operation such as calculation of capacitance.

A system such as OYSTER may be implemented as an extension to a solid geometric modeling system which provides the means to

1. Represent the shape, location, and orientation of rigid solid objects,
2. Represent family tree relationships among objects, for example, that object *a* is a son of object *b* and a brother of object *c*,
3. Perform transformations on solid objects, in particular, the Boolean operations of union, intersection, and difference,
4. Perform operations interactively with both command and graphics interfaces,
5. Execute user procedures which define new operations and which may internally call other system operations.

The Geometric Design Processor (GDP) [9, 10] is a geometric modeling system that provides these facilities. GDP is the basis for the discussion of simulation of process steps and analysis of device properties, and also for the implementation of the feasibility study given in this paper.

Process step modeling

The aim of OYSTER process step modeling is to simulate the geometric effects of the various process fabrication steps currently used in chip technologies. The major choice to be made is the selection of the level of modeling complexity. In this paper we define two domains of user interaction, process and structure, which represent high and low levels of complexity, respectively.

In the *process domain* of user interaction, a process step is described in terms of process parameters, for example, time, temperature, and solution. In the process domain, the simulation of a process step applies appropriate transformations, based on physical processes, to appropriate structures, to produce model structure states at the end of the step.

In the *structure domain* of user interaction, a process step is described in terms of structure parameters, for example, structures affected and thicknesses developed. Thus, in simulation of a process step, the model structures at the end of the step can be determined geometrically.

The selection of the level of geometric detail at which to represent the physics has yet to be determined. A wide range of level of detail is possible, from very simple (and cheap) with, for example, sharp corners, no undercutting, no lifting,

etc., to representations accurate to the level of understanding of the process physics [11] (and significantly more expensive). At present, modeling at the molecular and crystallographic imperfection level does not seem warranted but may eventually become necessary as device dimensions shrink. In any case the subject of imperfect crystals is an interesting application of geometric modeling in its own right.

The choice of a modeling point or level of detail may be biased to rather high levels of detail by current technology trends. As advances in lithography continue to lead to finer line widths and smaller devices,

1. Elements that used to be parasitic now tend to dominate in determining device performance, and it becomes essential for the designer to take them into account, and
2. Alignment tolerances between the various mask levels become relatively more critical, and control of these tolerances is necessary to ensure that a given statistical yield of manufactured chips will occur within the nominal design window.

Both these trends indicate a need for finer levels of detail than shown in the feasibility study. Fortunately, an evolutionary approach is possible, starting with simple models and evolving to more complex. Even when the highest level of detail is available, the user may wish to be able to select a lower level of detail appropriate to his current application.

Based on the fabrication steps arising during the FET feasibility study and shown in Fig. 1, OYSTER primitive process steps will include at least those shown in Table 1. This is not intended to be an exhaustive list; rather, it indicates the class of primitive processes envisaged for OYSTER. Other device technologies, for example, bipolar and Josephson families, will require further fabrication steps. Recent developments, such as self-aligned stencil and oblique deposition processes [12], call for explicit 3-D step modeling rather than the implicit 3-D nature of the steps tabulated above. Eventually, a common set of steps may be established.

The occurrence of these primitive process steps in the overall process shown in Fig. 1 is shown in Table 2. It can be seen that the count of 28 steps is rather arbitrary and that in many cases several process simulation primitives consistently occur in the same step (e.g., Develop and Wash) or consistently occur in successive steps (e.g., Expose and Develop). At an even higher level of clustering of steps, the sequence Apply photoresist, Expose with mask, Develop, Wash, Etch occurs five times, once for each mask. To preserve generality and maintain flexibility in experimentation with process steps, we have chosen not to exploit this clustering and to represent this FET fabrication process in terms of these nine primitive process steps.

Table 1 Primitive process steps.

Fabrication step	*Physical process*
1. Grow oxide	Heating in oxidizing atmosphere
2. Deposit material	Deposition of vapor
3. Apply photoresist	Dripping on, spinning out flat
4. Expose with mask	Directed irradiation of photoresist
5. Develop	Partitioning photoresist into exposed and unexposed volume regions
6. Etch	Dissolving in acid
7. Wash (or Strip)	Dissolving in solvent
8. Lift-off	Removal of photoresist and supported material
9. Implant	Directed selective irradiation and diffusion into material

Table 2 Primitive process steps in overall process of Fig. 1.

Process step	*Steps 1–10*	*Steps 11–20*	*Steps 21–28*
Grow oxide	-x-----x-x	----------	--------
Deposit material	--x-------	x---------x	------x-
Apply photoresist	---x------	-x---x----	x---x---
Expose with mask	---x------	-x---x----	x---x---
Develop	----x-----	--x---x---	-x---x--
Wash	----x-x-x-	--x-x-x-x-	-x-x-x--
Etch	-----x----	---x------	--x-----
Lift-off	----------	----------	-------x
Implant	----------	-------x--	--------

The simulation of each process step is an OYSTER system primitive operation based on geometric interpretations of simplified forms of the physics. Each process step is parameterized, for example, in terms of reaction coefficients (times, concentrations, temperatures, reactants, etc.) or layer thicknesses depending on the domain. The output of an OYSTER simulation of a process step is a set of 3-D geometric models of and relationships among the structures generated by the step. Of course, each step is simulated in the context of its place in the entire sequence of process steps. Thus, execution of an identical fabrication step will produce different geometric shapes at different stages of the overall process sequence. By simulating all steps in the correct sequence, models of all the material shapes making up an operating device on a chip can be created. The prototype study discussed later gives graphic examples of the shapes pro-

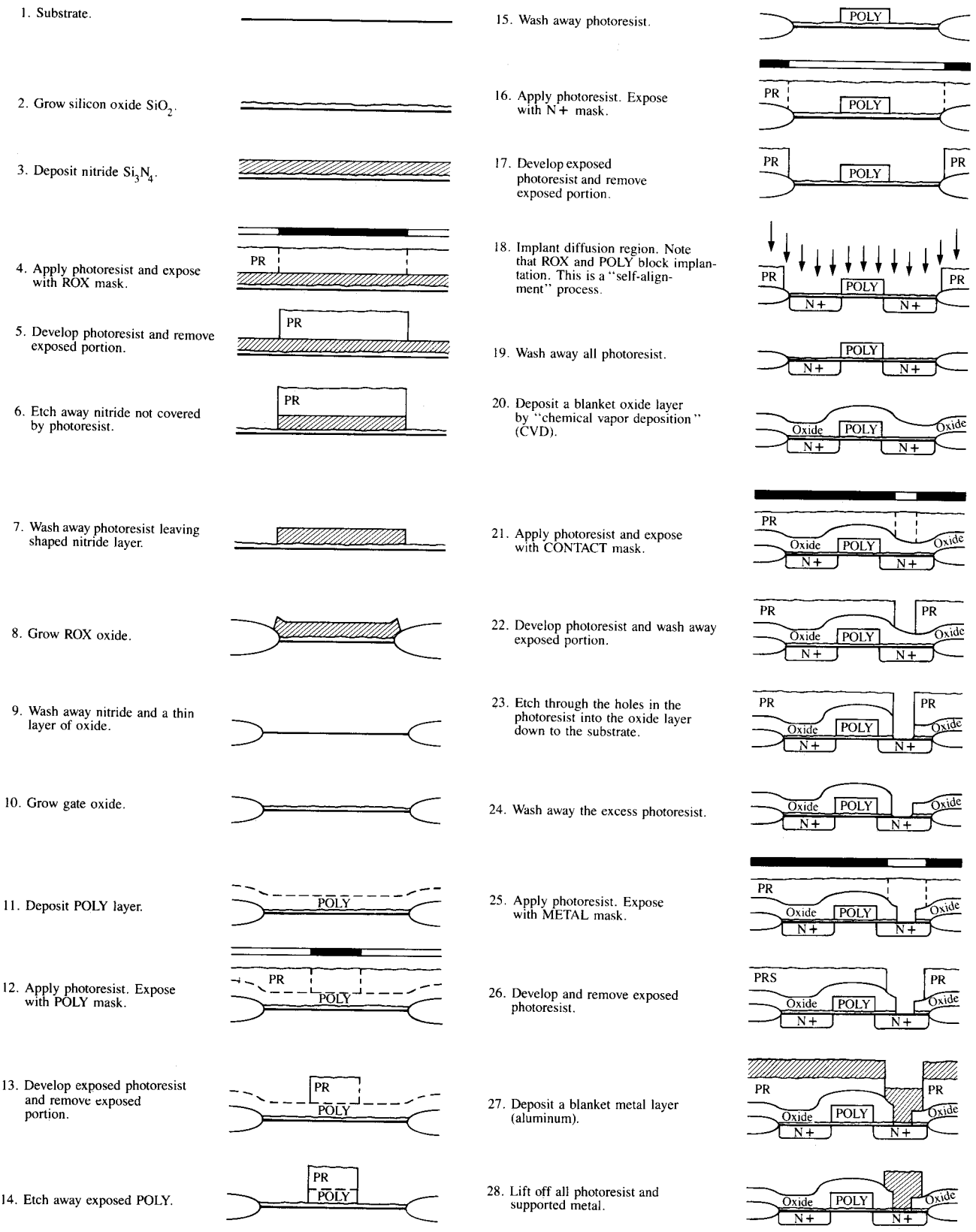

Figure 1 FET process steps shown in cross section through the gate area.

duced in such a sequence of steps using only a very simple interpretation of the physics. These simple methods give the basic shape outlines, but detail features at vertical edges, corner roundings, etc., need more accurate modeling.

We now discuss the nature of the procedures to implement each of these primitive process steps. The descriptions illustrate the use of parameters and Boolean operations between geometric objects. The primitive process step functions are described as procedure calls in a geometric modeling system and data base environment. Data base nodes may contain geometric objects and other data, and links among nodes may represent relationships among objects. Simulation of a sequence of fabrication steps generally starts with an initial process node, which is the parent of other nodes defining inputs to the system, such as a substrate node and a set of mask nodes. As the simulation steps proceed, nodes and relationships are built up or destroyed until, after the final step, the data base contains nodes that represent all the 3-D structures of the complete device.

- *Process domain modeling*

Process domain modeling implies a system with *knowledge* of VLSI processes. A typical example of knowledge in the VLSI process domain is that silicon is transformed to silicon dioxide in the presence of a suitable oxidizing atmosphere, at a rate determined by the relevant rate equations and process parameter values. The knowledge required may be considered in two parts, the structures affected and the nature of the transformations to be applied. Various methods of knowledge representation have been developed [13], for example, by *rules* of the form

IF there is material of type = silicon,
AND there is an atmosphere of type = oxidizing,
AND there are surfaces common to the material and the atmosphere,
THEN the material is transformed using function Form-oxide (model, material, surfaces, temperature).

Knowledge-based systems of this form are known as *rule-based Expert systems*. The use of a knowledge base also permits checking for process errors (e.g., developing a nonexistent photoresist) and for enforcement of good design practices. We do not consider the development of a VLSI knowledge base further here, other than to note that even though tools have been developed for constructing such systems [14], much work will be necessary to develop an effective VLSI knowledge base.

Grow oxide
Command syntax:
Grow-oxide('oxide-name', atmosphere, temperature, time)

Command semantics:
A new data base node is created and the parameter values are stored in the node. The parameter 'oxide-name' is assumed as the name of the newly created node and allows subsequent reference to the oxide structure by name. An overall oxide layer is grown isotropically over all the sensitive upper surface exposed to the atmosphere at a rate that is a function of the material, the atmosphere, and the temperature. The growth rate is integrated over the specified time period. The 3-D structure representing the new oxide layer is stored at the new node. The node is linked to the substrate node as a brother and to the process node as a son. The accuracy of the isotropic growing operation is dependent on the level of detail at which the simulation is specified.

Note that isotropic growing (and shrinking) operations are algorithmically very similar to transformations used in model-based numerically controlled (NC) machine tool cutter path generation and also in robot collision-free path planning [15], as are the Deposit material, Wash, Etch, and Implant fabrication steps.

Deposit material
Command syntax:
Deposit-material('material-name', atmosphere, temperature, time)

Command semantics:
Essentially the same semantics as for Grow-oxide, except that rules appropriate to deposition are applied and the material deposited is not necessarily an oxide.

Apply photoresist
Command syntax:
Apply-photoresist('photoresist-name', type, thickness)

Command semantics:
A new named node is constructed in the data base, and all parameter values are stored there. The node is linked as a brother to the substrate node and son to the process node. The photoresist thickness is expected to be significantly larger than the grossest vertical detail present on the upper surfaces of the structures formed so far. It is assumed that the photoresist flows into all exposed regions on the structure upper surface. The geometric solid representing the photoresist is stored at the node and is formed by creating a wafer-sized planar sheet of the specified thickness positioned above and parallel to the substrate, with its lower surface at the same height as the lowest upper surface point on the current structure. The existing structures are subtracted from its lower surface. The photoresist type parameter allows later recognition of sensitivity to particular develop and wash operations.

Expose with mask
Command syntax:
Expose-with-mask('mask-name', radiation-type, direction)

Command semantics:
Parallel radiation from the specified direction and filtered by the specified mask is used to partition the photoresist present in the model into two volume regions containing those points in the photoresist that receive radiation and those points that do not. These components are stored in two new nodes created as sons of the original photoresist node. The accuracy of representation of occultations of the radiation by surface topology depends on the level of detail being represented. The simplest approach might generate the prism formed as the projection of the mask in the specified direction and intersect the prism with the substrate structure to form one of the partitions. This simple approach ignores occultation effects. More detailed algorithms could construct the visible volumes allowing for shadowing and also allow for nonparallel radiation. The partitions are labeled as *latent*.

Develop
Command syntax:
Develop(developer-type)

Command semantics:
The latent portions of photoresist sensitive to developer-type are labeled *visible*, i.e. selectively sensitive to wash solutions. It is assumed that development is to completion; otherwise, a time parameter would be specified.

Wash
Command syntax:
Wash(solution-type, temperature, concentration, time)

Command semantics:
All objects soluble in the solution-type are subjected to an isotropic shrinking on all exposed surfaces, at a rate specified by appropriate parameterized rate equations, integrated over the specified time. Generally the time specified permits complete removal of any soluble object. If an object volume becomes zero, the object node is deleted.

Etch
At a coarse level of detail, the parameterized procedural description of Etch is essentially the same as for Wash. At a more detailed level, anisotropic effects can be included, for example, undercutting.

Lift-off
Command syntax:
Lift-off(solution-type)

Command semantics:
All remaining photoresist material sensitive to solution-type is completely dissolved (i.e., deleted). Any structures supported by the photoresist are also deleted. Thus, metal layers in the final fabrication steps may be removed. Note that this operation implies a knowledge base that knows about support relationships.

Implant
Command syntax:
Implant('implant-name', radiation-type, direction, intensity, time)

Command semantics:
The wafer is irradiated with parallel radiation of the specified type, direction, and intensity for the specified time. A new named node is generated (as the brother of the substrate, etc.) to hold the implantation region to be generated. The implantation region is generated by propagating the penetration of radiation into exposed regions of material at a rate governed by radiation type and intensity and by buildup of new material. The propagation rate is integrated over the specified time. Thus, the new region is grown, and some material is deleted from the structures into which the implantation is embedded. Note also that the implantation region may be modified (e.g., by spreading and undercutting) as a side effect of other fabrication steps.

It is the implantation region which determines the operating characteristics of the device (e.g., the I-V relationship). Thus, to predict these relationships the implantation propagation process must be modeled to sufficient accuracy; an analytic model of the implantation process has been given in [2].

Structure domain modeling

The intent of structure domain modeling is to allow simulation of process steps to be performed without a VLSI knowledge base and to allow implementation directly by known geometric transformations. The parameterized treatment of the simulation of process steps given above is in terms of process parameters of the form temperature, time, etc., and assumes a VLSI knowledge base. The simpler approach described here is based on structure parameter values, for example, layer thickness deposited, on explicit assertion by the user of the structures affected by a process step, and on transformations producing idealized boundaries, for example, with sharp corners. The acceptance of idealized boundaries means that many of the transformations required are implementable largely in terms of operations such as Boolean union, intersection, and difference, which are available on an existing geometric modeling system such as GDP. In this section we describe a representation of the primitive process steps in these simpler and directly implementable terms.

The description is again in terms of a data structure in which nodes represent objects and links represent family relationships between objects. The simulation starts with a tree-structured data base with three sub-trees:

1. *Structures* formed during execution of the process steps, commencing with just the substrate and finishing up with the complete set of device structures,

2. *Masks* supplied as input,
3. *Utility structures* entered initially and/or generated as the simulation proceeds. These structures are transparent to the user and are intended to simplify the simulation process. Utility structures may include a block representing the overall shape of the device cell, a prototype photoresist layer with son nodes to contain the positive and negative components after exposure, and structures with special profiles to allow more accurate representation of critical device components.

The operations are described in terms of the Boolean operations, a swept volume operator that may be implemented explicitly or indirectly with Boolean operations, an implicit vertical direction, and a number of geometric operations that can be defined in a limited domain. These operations include the determination of common surfaces between objects, the determination of portions of surfaces visible in a line-of-sight sense from a given point or direction, and the determination of the current upper surface layer.

Grow oxide
Command syntax:
Grow-oxide('oxide-name', o-type, o-base, v-othick, h-othick)

Command semantics:
A new named data base node is created in the *structures* sub-tree. The oxide structure is built from the volume swept as the free upper surfaces of the oxide-base are translated vertically by v-othick. When nonzero, the parameter h-othick allows representation of deposition on lateral surfaces. In the case of essentially rectangular structures, the lateral effects can be represented by lateral translations in each of the coordinates of the horizontal plane. The implementation can recognize special oxide types and invoke special utility structures when necessary, for example, when representing the profile of a recessed oxide layer.

Deposit material
Command syntax:
Deposit-material('material-name', m-type, v-mthick, h-mthick)

Command semantics:
Essentially the same semantics as for Grow-oxide, except that the deposition is over the whole cell upper area rather than being restricted to a specified base region.

Apply photoresist
Command syntax:
Apply-photoresist('photoresist-name', thickness)

Command semantics:
A new named data base node is created in the *structures* sub-tree. The photoresist thickness is generated as a cuboid with horizontal dimensions matching the device cell, and of specified vertical thickness, with all the existing *structures* subtracted.

Expose with mask
Command syntax:
Expose-with-mask('photoresist-name', 'mask-name')

Command semantics:
The prism volumes swept as the mask is translated vertically are used with union and difference operations to partition the photoresist into two son substructures, the volume regions in the photoresist that receive radiation (pos) and those points that do not (neg). The partition is latent in that the photoresist is not selectively sensitive to washing.

Develop
Command syntax:
Develop('photo-resist-name')

Command semantics:
Latent photoresist is marked as being selectively sensitive to washing.

Wash
Command syntax:
Wash('structure-name')

Command semantics:
The named structure is erased, typically the pos or neg portion of a photoresist.

Etch
Command syntax:
Etch('etched-structure', 'masking-structure')

Command semantics:
The masking structure is swept vertically and intersected with the structure to be etched. Note that with this definition of the primitive instruction, a single actual process step may be represented by several primitive steps, one for each etched-structure.

Lift-off
Command syntax:
Lift-off('photoresist-name', 'supported-structure-name')

Command semantics:
The named photoresist material and supported structures are deleted.

Implant
Command syntax:
Implant('implant-name', 'implant-target', i-depth)

Command semantics:
The free upper surface of the implant target is swept down by i-depth to generate a new named structure. The new structure is subtracted from implant-target.

The parameterized structure domain description of the FET process step sequence shown in Fig. 1 is given in Table 3.

Analysis functions

The analysis functions to be included in the OYSTER system fall into the following main categories:

1. Visualization of geometric objects, e.g., line drawings,
2. Derivation of properties found as the result of simple geometric operations, e.g., volume,
3. Derivation of properties found as solutions to differential or integral equations, e.g., capacitance between conductors,
4. Derivation of properties based on statistical parameters such as alignment tolerances.

Visualization of objects

The simplest visualization of objects that is readily available on geometric modeling systems is line drawings with hidden lines suppressed and with user selected viewing parameters: eyepoint, gazepoint, and scale. Our experiences with the feasibility study confirm our opinion that line drawings of unfamiliar objects are hard to interpret. Thus OYSTER will also provide 3-D color shaded graphics so that a 3-D picture of a device can be constructed from the original process specification and displayed on a color shaded graphic terminal as each layer or process step is completed. The designer can interact with the display and view the device structure from any desired angle or display the cross section at any desired plane. In either the line drawing or color-shaded graphic display approach, the user can visualize directly the results of process steps and can hunt interactively for anomalies and unexpected geometric happenings.

Examples of errors in the geometric structures produced can be seen in the illustrations from the feasibility study shown below. In this case, these errors were caused by faulty modeling techniques but can readily be recognized as errors by interactive visual inspection of the model structures built.

Properties as simple geometric results

Properties that may be directly and speedily calculated from the 3-D geometric structures generated by process simulation steps include

1. Surface area,
2. Contact area,
3. Volume,
4. Minimum (maximum) cross-section area,
5. Minimum (maximum) path length contained within an object,
6. Minimum (maximum) distance between objects.

These functions can be implemented simply (indeed, they may already be available) in any complete modeling system and can be expected to be executed interactively. Their implementation is not considered further here.

Properties as solutions to differential or integral equations

Many physical properties may be expressed as the solutions to differential or integral equations over the volumes or boundaries of the components of devices and the media in which they are embedded. Such properties include resistance, inductance, capacitance, mechanical stiffness and strength, heat transfer, operating characteristics, etc. The important thing to note is that the solid object structures and their relationships generated by the process step simulation stages are the basic inputs required for these calculations. Thus we expect to be able to generate these properties directly from the simulation.

In general these equations are solved by finite element methods and, fortunately, suitable finite element packages have already been developed, or are currently active areas of research, for most of the properties of interest. It remains to devise means for the automatic construction of interfaces from the geometric data base models generated by the OYSTER simulation to the finite element analysis package. The interface generally requires the partitioning of the solid models into finite element meshes of the form required by the problem package. In general, this is a nontrivial problem in geometric modeling and is still not completely automated. However, almost automatic methods can be expected to be available in the near future. In the feasibility study discussed below, *ad hoc* methods based on the mask dimensions were used to generate the required form of mesh for capacitance calculation.

Statistical effects

The representation and analysis of stochastic variables in manufacturing operations is still in its infancy. However, we believe that the parametric procedural approach based on geometric modeling presented here is the approach most likely to succeed. Thus, although general results on statistical effects are not yet available, preliminary work has been done [16, 17], and further results can be expected.

In OYSTER, statistical variations can be applied to coordinate transformations (e.g., alignments of masks) and process parameters (e.g., thicknesses of layers, dimensions of objects). A pseudo-random number generator can also be used to trigger observed sporadic effects (e.g., surface irregularities). In the case of surface defects, the OYSTER approach allows the effect of defects occurring at one level to be propagated up through covering layers as they are laid down. We expect OYSTER eventually to be able to compute the overall distributions of the final shapes and positions of device structures. These results will be used for calculating distributions of properties such as capacitance, and also to provide input to models predicting yield.

Table 3 Structure domain simulation of FET process steps shown in Fig. 1.

Simulation step	*Comments*
1 FETCH FET	load model with substrate, masks, and utility structures
2 GROX TSIO2 SIO2 SUBSTR T1 0	grow temporary SiO2 layer (TSIO2) on substrate with T1 vertical thickness and with no horizontal thickness
3 DEP TSI3N4 SI3N4 T2 0	deposit nitride layer (TSI3N4) with T2 vertical thickness and with no horizontal thickness
4 APR PR1 T3	apply photoresist (PR1) T3 thick
EXP PR1 M-ROX	expose with ROX mask
5 DEV PR1	develop PR1
WASH PR1.POS	wash away exposed portion of PR1
6 ETCH TSI3N4 PR1	etch away nitride not covered by PR1
7 WASH PR1.NEG	wash away remainder of PR1
8 SROX ROX ROX SUBSTR T4 T5	grow ROX on substrate T4 thick with T5 undercut
9 WASH TSI3N4	wash away nitride
WASH TSI02	wash away TSI02
10 GROX GOX GOX SUBSTR T6 0	grow gate oxide on SUBSTR with T6 vertical thickness and with no horizontal thickness
11 DEP POLY POLY T7 0	deposit polysilicon layer with T7 vertical thickness and with no horizontal thickness
12 APR PR2 T8	apply photoresist PR2 T8 thick
EXP PR2 M-POLY	expose with POLY mask
13 DEV PR2	develop exposed PR2
WASH PR2.POS	wash away exposed portion
14 ETCH POLY PR2.NEG	etch away exposed polysilicon
15 WASH PR2.NEG	wash away remaining PR2
16 APR PR3 T9	apply photoresist PR3 T9 thick
EXP PR3 M-NPLUS	expose PR3 with NPLUS mask (N.B., the NPLUS mask nearly protects other chip regions)
17 DEV PR3	develop PR3
WASH PR3.P05	wash away exposed portion of PR3
18 IMP NPLUS SUBSTR T10	implant diffusion layer (NPLUS) into SUBSTR T10 thick
19 WASH PR3.NEG	wash away remainder of PR3
20 DEP BOX BOX T11 T12	deposit blanket oxide layer with T11 vertical thickness and with T12 horizontal thickness
21 APR PR4 T13	apply photoresist PR4 T13 thick
EXP PR4 M-CONT	expose PR4 using CONT mask
22 DEV PR4	develop PR4
WASH PR4.POS	wash away exposed portion of PR4
23 ETCH GOX PR4.NEG	etch through holes in PR4 into blanket oxide
ETCH BOX PR4.NEG	etch through holes in PR4 into gate oxide
24 WASH PR4.NEG	wash away remaining PR4
25 APR PR5 T14	apply photoresist PR5 T14 thick
EXP PR5 M-METAL	expose PR5 with METAL mask
26 DEV PR5	develop PR5
WASH PR5.NEG	wash away UNexposed portion of PR5
27 DEP METAL METAL T 15 0	deposit metal layer with T15 vertical thickness and with no horizontal thickness
28 LOFF PR5 METAL	remove PR5 and lift off METAL it supports

Feasibility study: an FET example

In order to demonstrate the feasibility of the OYSTER geometric modeling based approach to studies of the VLSI fabrication process, a sample device was selected from an actual test chip. GDP was first used to simulate the sequence of fabrication steps, which was followed by the automatic preparation of input data to a capacitance calculation program. The simulation in this prototype study is not parametric and uses only existing GDP functions, in particular Boolean (the GDP MERGE) operations on solid objects. Combinations of existing GDP functions (again, mostly MERGE) were used empirically to decompose the resulting models of the poly, metal, and implant layers into finite elements with the topology required by an IBM standard

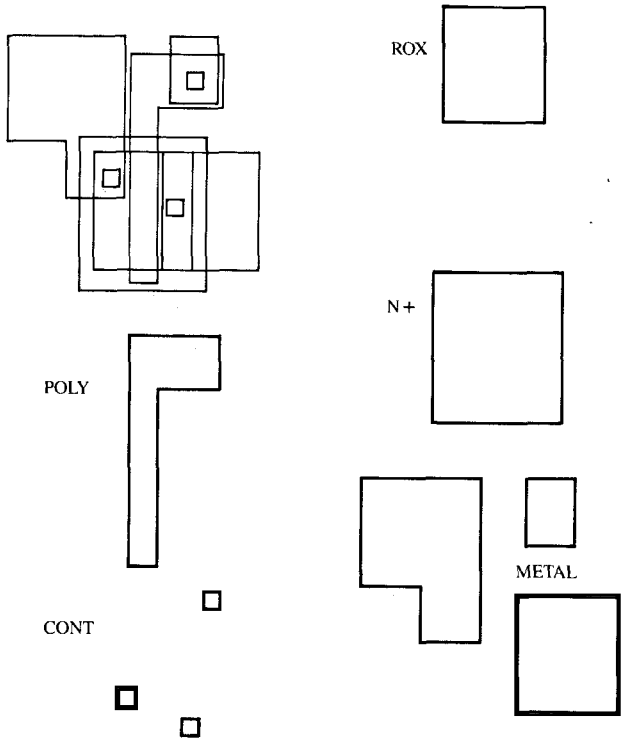

Figure 2 Composite FET mask set showing five mask levels.

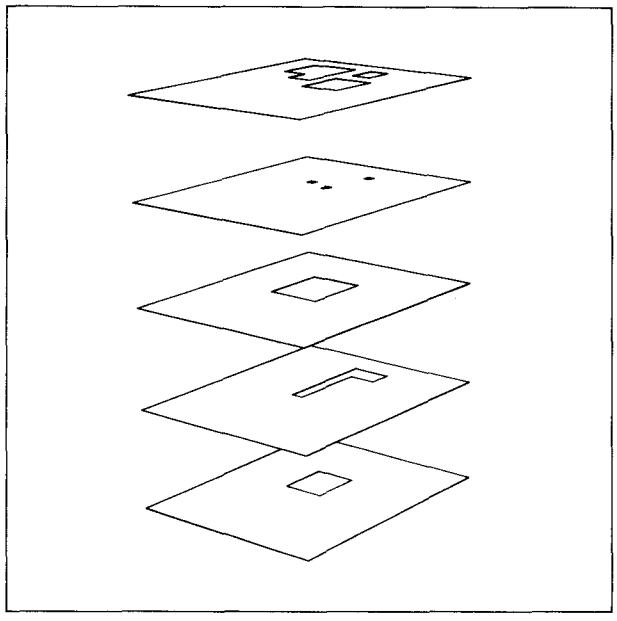

Figure 3 GDP mask set generated from masks in Fig. 2.

package for calculation of device capacitances. The capacitance package was then run successfully to find the capacitances among the polysilicon, implantation, and metal structures.

The FET device used in our example is built using five mask levels. The masks are ROX, POLY, NPLUS, CONTACT, and METAL. Seven structures are produced above the substrate: recessed oxide, gate oxide, polysilicon, implant, blanket oxide, contact, and metal.

The input data to our study consisted of

1. Planar masks defining the sample device, and
2. Definition of a sequence of 28 process steps that are used in conjunction with the mask set to build the device.

The next two sections describe these inputs. Subsequent sections then describe the GDP based simulation of the process steps and the calculation of conductor capacitance.

- *An FET device mask set*

The patterns that define the sample FET device, shown in Fig. 2, are taken from an actual mask set of a CMOS test development chip. Figure 3 shows an exploded 3-D view of the five masks produced by GDP from the mask input data. An automatic interface could easily be written between the standard production mask description data files and GDP; for this initial study, the procedural description provided to GDP was used, and the conversion was performed manually by writing a GDP source procedure.

- *A sample FET process step definition sequence*

Figure 1 shows a series of schematic cross-section drawings which represent the series of 28 process steps used to build the chip device from the masks shown in Fig. 2. The cross section was taken through the gate area to show a portion of the device that exhibits the most complex intersection of the five mask levels. Note that the drawings are not to scale.

The process steps begin with a blank substrate and end, after the last step, with models representing the seven material layers making up the FET device.

- *Process step modeling*

The 28 steps were modeled in the structure domain in GDP to *approximate* the results of the actual process simulation steps discussed earlier. The GDP system allows modification of objects through an interactive command interface and has a macro capability that enables strings of interactive commands to be composed. In the implementation of OYSTER new, special purpose transformations will be provided as interactive commands. In this feasibility study, the existing functions were used, and a separate macro sequence was written for each of the process steps. These macro command strings are not yet parameterized as required by the OYSTER system. Rather, structure domain parameter values (e.g., layer thickness) have here been embedded as constants appearing in the macros. Each GDP macro step has been defined to save the state of the process at its completion,

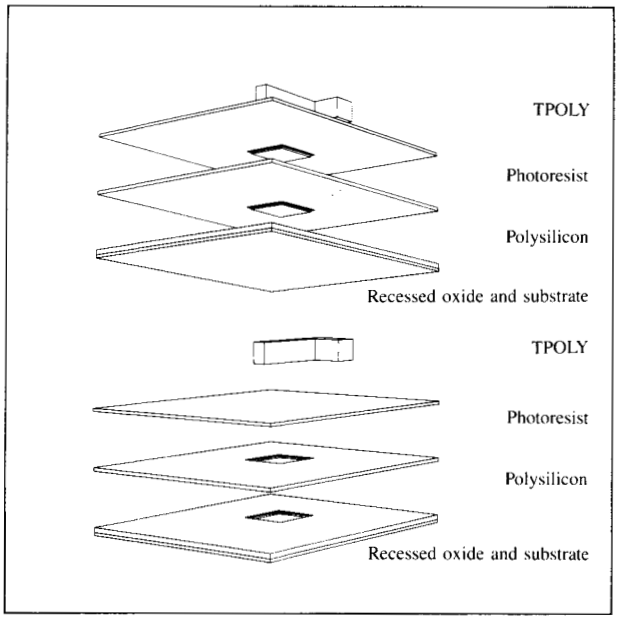

Figure 4 State of model at end of step 12, i.e., at start of step 13.

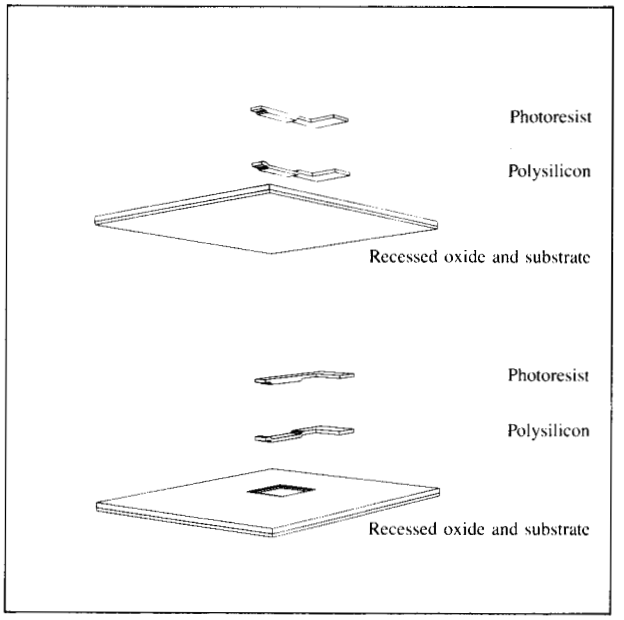

Figure 5 State of model at end of step 14.

which is then used as the input state of the next step. In this way, the simulation can be restarted at any of the steps with changed values in the command strings.

In order to obtain a better approximation to an important physical feature than was possible using the straightforward approach of Boolean operations on masks, the profile of the recessed oxide "bird's beak" was entered directly and used in the ROX deposition steps to represent the curvature that the actual process creates. Subsequent steps, which build layers above and below the beak, reflect this same beak curvature through Boolean operations involving the beak shape.

To illustrate the GDP simulation used throughout the prototype study, a sequence of steps in the simulation are described in some detail. Figure 1 shows the full sequence of process steps; here we consider the steps involved in the construction of the polysilicon component of the FET device.

At the end of step 10, the data base contains models of the substrate, ROX, and gate oxide structures. In step 11 a polysilicon layer is deposited over the whole cell area. The simulation generates a substrate-shaped cuboid of suitable thickness and forms the union with the ROX structure at its upper surface, translates vertically, and subtracts the ROX from the lower surface. This represents a constant vertical thickness deposition process but does not incorporate any lateral thickness. Note that in the blanket oxide deposition in step 20, the constant thickness property on vertical faces is approximated more carefully using lateral as well as vertical translations.

In step 12 the second photoresist layer is applied and exposed using the POLY mask. Again, a substrate-shaped cuboid of the correct thickness is positioned over the current structure, and the polysilicon is subtracted from the under side. A temporary object, TPOLY, is formed as the right prism with the cross section of the polysilicon mask. This is used in later steps.

In step 13 the photoresist is developed. This is simulated by forming the intersection of the photoresist structure with the TPOLY prism. Note that this structure is not actually used in any functional manner in the geometric simulation, but is included for completeness. It is erased in step 15.

Step 14 etches away the exposed polysilicon by replacing the polysilicon structure with its intersection with the TPOLY prism.

In step 15 the remaining photoresist is washed away.

Figure 4 shows the structures created by the end of step 12. The substrate (much too thin) and recessed oxide layer are drawn in position with the polysilicon, photoresist, and TPOLY drawn in raised position. Figure 5 shows the effect

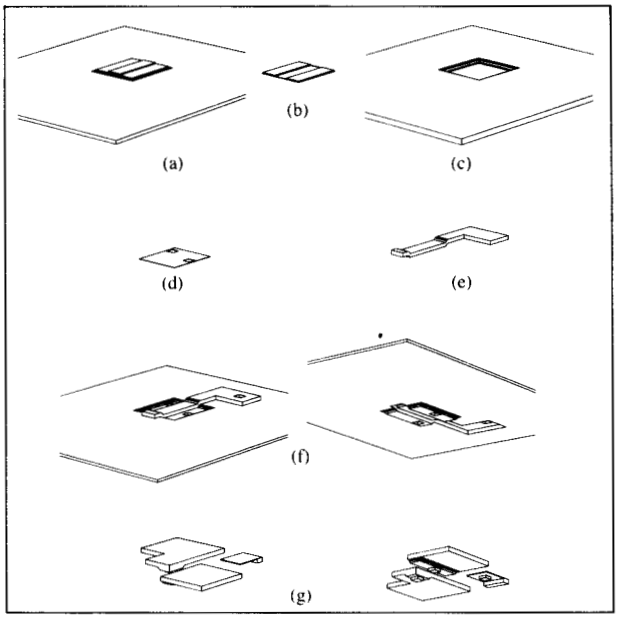

Figure 6 Close up views of FET component structures generated by OYSTER.

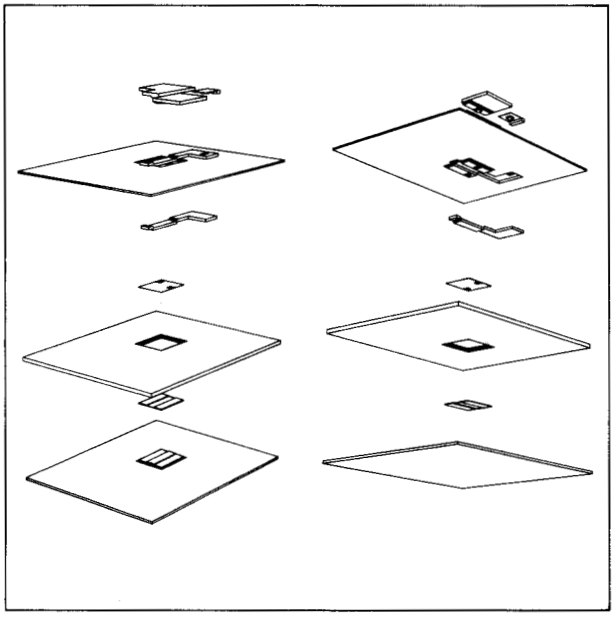

Figure 7 Exploded view of FET device structures generated by OYSTER.

of exposure and development on the photoresist (step 13) and the result of the etching away of the exposed polysilicon (step 14).

Figure 6 shows 3-D views of the individual shapes that are deposited on the chip to form the FET device. The substrate is shown at (a), the implant region at (b), the recessed oxide layer at (c), the gate oxide at (d) (note the two square portions that are etched away when the contact regions are formed), the polysilicon layer at (e), the blanket oxide from above and below at (f) (note the square shape contact hole where the metal layer will contact the poly), and the metal layer from above and below at (g) in three separate volume regions. Note the underside view of the metal layer where the top shape, as indicated in the 2-D mask pattern, is joined to the square contact shape by an intermediate shape formed by the recessed portion of the blanket oxide. This intermediate shape is not easily envisioned by looking at the juxtaposition of the 2-D masks. Figure 7 shows some of the other views that can be obtained from the GDP model.

At this point we draw attention to some particular errors in the structures generated in the simulation of process steps. In the study the errors were caused by mistakes in the macro-instruction strings, but in a real situation they might have been caused by mistakes in mask design or alignment. Two errors in the structures can be seen in Fig. 6:

1. The implantation region should be two disjoint, roughly cuboidal regions, but the simulation shows two thin triangular prisms linking the main regions.
2. The metal layer that goes over a high region of the blanket oxide has been cut off too thinly. This error was caused by giving all the metal a flat top rather than constant deposition thickness.

The simulation could readily have been changed to have corrected these errors, but we preferred to leave them as examples of the ability that OYSTER provides to correct design errors at design time by interactive manipulation and inspection of the model structure.

• *Direct capacitance calculation using a standard package*
Electrical analysis packages which compute capacitances between idealized 3-D models of conductors located within infinite dielectric flat regions have been available for a number of years. Heretofore there has been no means to capture the 3-D geometry directly from the device design specification. Using OYSTER, once the 3-D models of the conductors, such as the polysilicon layer, have been formed, the input data for the package can be generated directly. The input data involves a decomposition of idealized forms of the 3-D conductors into suitable finite elements, the specification of the thicknesses of dielectric layers, and the preparation of an input description file for the package.

In this feasibility study, the generation of the finite elements was performed by a set of GDP macro-instruction strings. The macros are again not parametric, though they have been written in terms of input values such as mask aperture dimensions, and these values are embedded as constants in the macro strings. A new GDP instruction was

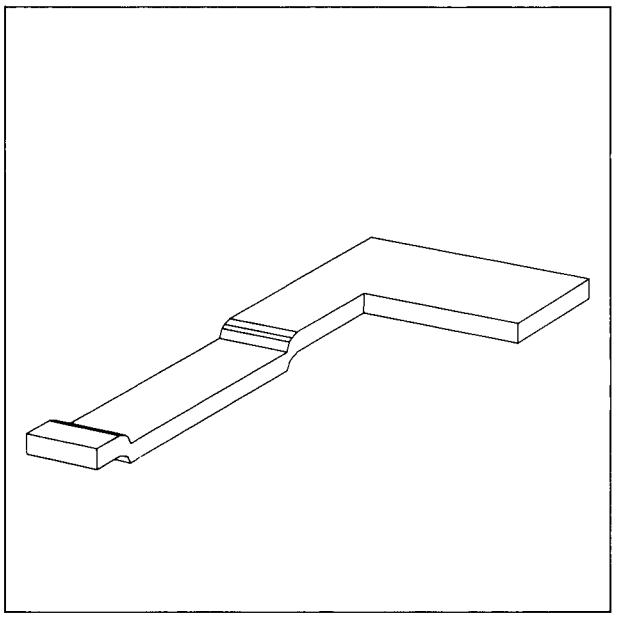

Figure 8 Model of polysilicon structure created by OYSTER.

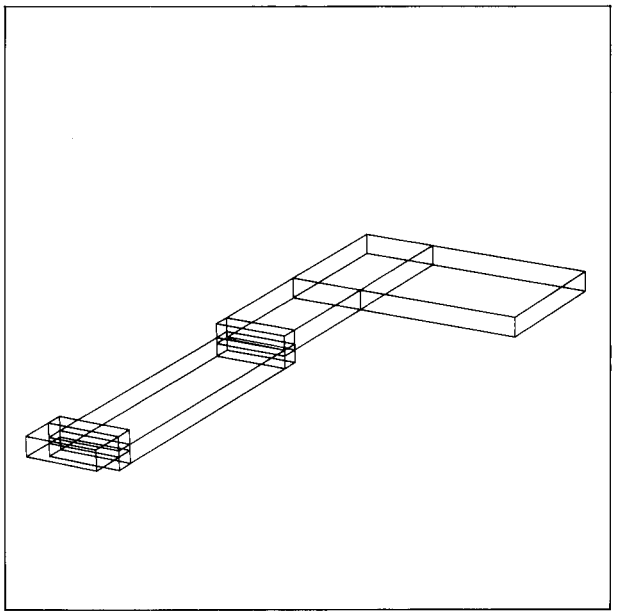

Figure 9 Finite element decomposition of polysilicon structure.

used to convert the finite elements into the input description file for the package, inserting nominal values for the dielectric layer thicknesses and asking the user to modify them interactively if desired.

Since capacitance is a surface effect, the interface to the package requires each surface of each of the cuboids making up the 3-D conductors to be labeled as to whether the surface is *charge-free*, i.e., interior, or *charged*, i.e., exposed. This operation is, of course, now performed automatically, greatly simplifying the job of preparing input data for the package. Figure 8 shows the polysilicon model created by OYSTER, and Fig. 9 shows the cuboidal finite element approximation that OYSTER created to represent the polysilicon shape to the package. Similar decompositions were performed on the implantation region and the metal. The data generated automatically from this finite element model were used successfully to find the resulting capacitance matrix for the three conductors—polysilicon, implantation region, and metal.

Summary and conclusions

In this paper we have presented a design for a solid geometric modeling based tool (OYSTER) for parameterized simulation of the fabrication steps and analysis of the structures produced in a VLSI process. The system takes as input a description of a set of planar masks and a specification of a set of parameterized process steps, and it generates solid geometric models of the device structures produced at each step in sequence of the fabrication process.

Two domains of user interaction have been described, process and structure. In the process domain, process steps are described in terms of process parameters and the system contains a VLSI knowledge base. In the structure domain, fabrication process steps are described in terms of the device components to be changed and the geometric parameters of the changes to the component structures. A process domain implementation with its knowledge base is seen as a long term research activity. A structure domain implementation is feasible now and is at present under way.

The technical feasibility of the design has been demonstrated by a study in which the GDP system was used to simulate 28 steps of an FET process and then generate input to a capacitance calculation package. We believe that this prototype study has also demonstrated the practicality of the OYSTER system.

Our sequence for further work is

1. Complete an initial structure domain implementation, based on idealized representations of device structure steps and using existing geometric operators,
2. Extend the initial structure domain implementation to more detailed representations with new geometric transformations to enable shape boundaries to match physical effects better,
3. Define and implement new analysis functions such as minimum cross-section area, closest approach, path length. Operations such as these should not present major technical problems,

4. Invent and implement algorithms to generate finite element meshes for direct input to analysis packages. This is an area of active research, and algorithms may well be obtained from other workers. The availability of detailed geometric models and automated input generation may in turn allow more precise analysis to be performed,
5. Adapt the structure domain transformation models to process domain,
6. Develop a VLSI knowledge base,
7. Develop representations for stochastic parameters, such as alignment tolerances, and algorithms for producing useful analyses, such as tolerance buildup and inputs to yield models.

A major factor in the successful development of a design tool such as OYSTER is the development of an appropriate user interface. We have discussed here two domains of user interaction. The details of the user interaction, in terms of interactive graphics, command interfaces, procedural interfaces, ability to construct macros, must be developed in the context of the VLSI design environment.

The discussion so far has considered OYSTER as a system that simulates process steps in a discrete manner and has considered its use as an off-line design and simulation tool. An implementation based on continuous, real time integration of rate dependent phenomena would enable it to be used in a real time process control system as the predictor part of the control loop.

In the application of an existing technology (solid geometric modeling) to a new area (VLSI), the transfer of ideas can be expected to proceed in both directions. Thus, the availability of detailed shape representations may allow better finite element analyses to be performed, and the need to represent the transformations occurring in fabrication process steps may lead to new features in geometric modeling systems.

Aknowledgments

We would like to thank A. Appel for suggesting the possibilities of solid modeling in VLSI, G. Hu for patiently explaining details of the FET process, and V. Di Lonardo for supplying us with the mask data for our sample FET device. Helpful discussions concerning analysis packages were held with B. Landman, A. Ruehli, P. Brennan, and P. Wolff, Sr.

References

1. H. Ryssel, K. Haberger, K. Hoffmann, G. Prinke, R. Dumke, and A. Sachs, "Simulation of Doping Processes," *IEEE J. Solid-State Circuits* **SC-15,** 549–557 (August 1980).
2. D. Chin, M. R. Kump, H.-G. Lee, and R. W. Dutton, "Process Design Using Two-Dimensional Process and Device Simulators," *IEEE Trans. Electron Devices* **ED-29,** 336–340 (February 1982).
3. Savvas G. Chamberlain and Asim Husain, "The Need For Three-Dimensional Simulations for VLSI MOSFETs, CCD Imagers and CCD SPS Memories," *Research Report RC-9410,* IBM Thomas J. Watson Research Center, Yorktown Heights, NY, June 7, 1982.
4. A. Yoshi, H. Kitazawa, M. Tomizawa, S. Horiguchi, and T. Sudo, "A Three-Dimensional Analysis of Semiconductor Devices," *IEEE Trans. Electron Devices* **ED-29,** 184–189 (February 1982).
5. E. M. Buturla, P. E. Cottrell, B. M. Grossman, and K. A. Salsburg, "Finite-Element Analysis of Semiconductor Devices: The FIELDAY Program," *IBM J. Res. Develop.* **25,** 218–231 (July 1981).
6. A. E. Ruehli and P. A. Brennan, "Efficient Capacitance Calculations for Three-Dimensional Multiconductor Systems," *IEEE Trans. Microwave Theory Tech.* **MTT-21,** 76–82 (February 1973).
7. A. E. Ruehli and Pierce A. Brennan, "Capacitance Models for Integrated Circuit Metallization Wires," *IEEE J. Solid-State Circuits* **SC-10,** 530–536 (December 1975).
8. M. A. Wesley, "Construction and Use of Geometric Models," *Computer Aided Design,* Chapter 2, *Lecture Notes in Computer Science* No. 89, Springer-Verlag New York, Inc., 1980.
9. M. A. Wesley, T. Lozano-Pérez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," *IBM J. Res. Develop.* **24,** 64–74 (January 1980).
10. W. Fitzgerald, F. Gracer, and R. Wolfe, "GRIN: Interactive Graphics for Modeling Solids," *IBM J. Res. Develop.* **25,** 281–294 (July 1981).
11. F. Jones and J. Paraszczak, "RD3D (Computer Simulation of Resist Development in Three Dimensions)," *IEEE Trans. Electron Devices* **ED-28,** 1544–1552 (December 1981).
12. G. J. Dolan, "Off-Set Masks for Lift-Off Photoprocessing," *Appl. Phys. Lett.* **31,** 337–339 (1977).
13. P. H. Winston, *Artificial Intelligence,* Addison-Wesley Publishing Co., Reading, MA, 1977.
14. W. van Melle, "A Domain-Independent Production Rule System for Consultation Programs," *Proceedings of the International Joint Conference on Artificial Intelligence,* 1979.
15. T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *Commun. ACM* **22,** 560–570 (October 1979).
16. R. C. Hillyard and I. C. Braid, "Analysis of Dimensions and Tolerances in Computer Aided Mechanical Design," *Computer Aided Design* **10,** 161–166 (May 1978).
17. D. D. Grossman, "Monte Carlo Simulation of Tolerancing in Discrete Parts Manufacturing and Assembly," *Stanford Artificial Intelligence Laboratory Memo AIM-280,* Stanford University, Palo Alto, CA, May 1976.

Received September 23, 1982; revised October 25, 1982

George Koppelman *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Koppelman is a Research staff member in the Computer Sciences Department at the Thomas J. Watson Research Center. He received his B.S. from the University of Chicago, Illinois, in 1961 and joined IBM at the Research Center in 1965, working in pattern recognition, graphics, and interactive document scanning. He then spent a number of years with the Josephson development effort on circuit simulation, design automation tools for master image logic chips, and microprocessor design. These interests led to Mr. Koppelman's current position as a member of the VLSI design group.

Michael A. Wesley *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Wesley is manager of the geometric modeling project in the Manufacturing Research Center at the Thomas J. Watson Research Center and is an Adjunct Associate Professor of Computer Sciences at the New York University Graduate School of Arts and Sciences. He was educated at

Dulwich College, London, England, and the University of Cambridge, England, from which he received a B.A. in mechanical sciences in 1960 and a Ph.D. in control engineering in 1966. Since 1966, he has been a Research staff member at the IBM Thomas J. Watson Research Center and for the past nine years has worked on the automation research project. He has also worked on application programming for industrial robots, the Autopass high-level robot language proposal, the geometric design processor geometric modeling system, and algorithms for path planning. More recently, Dr. Wesley has worked on methods of generation and conversion of geometric data bases and their application throughout the CAD and CAM process.