Voxel-Based Heterogeneous Geometric Modeling for Surface Micromachined MEMS

Andy Perrin*, Venkat Ananthakrishnan**, Feng Gao**, Radha Sarma**, and G. K. Ananthasuresh*

*Mechanical Engineering and Applied Mechanics
University of Pennsylvania, Philadelphia, PA
**Mechanical, Industrial, and Manufacturing Engineering
University of Toledo, Toledo, OH
Contact: rsarma@eng.utoledo.edu or gksuresh@gksuresh.seas.upenn.edu

ABSTRACT

A heterogeneous geometric modeler for microelectromechanical systems (MEMS) is the focus of this paper. On the basis of a comprehensive formalism described in our earlier work, we present a 3-D implementation of the forward (from masks to a model) and inverse (from model to masks) problems. In order to solve the inverse problem, it is necessary to ensure that the geometric model is compatible with the chosen process. For this purpose, we have developed a feature-based constrained modeler that allows the MEMS designers to build models that are compatible with a process. We have chosen the surface micromachining process to develop the model as well as present a few illustrative examples.

Keywords: Geometric modeling, Heterogeneous modeling, MEMS, Feature-based modeling, Mask generation.

1 INTRODUCTION

The importance of the geometric design of MEMS is evident from the fact that almost all commercial MEMS CAD softwares [1-4, etc.] have this as an essential component. In this paper we address two aspects of geometric design of MEMS, viz. the *forward problem* of *art-to-part*, i.e., generating the geometric model of the part from the artwork of masks for a specified process; and the *inverse problem* of *part-to-art*, i.e., generating masks from the model of the part for a specified process. Existing software programs have the capability to generate 2-D and 3-D solid models from the mask and process data, but are limited in a number of ways. For example, in some of them only the most recently deposited layer can be etched and not all the exposed layers of a particular material. Depth-limited etches and multiple etches in the same layer are also not supported in many programs.

The inverse problem has received very little attention in the literature. Finding the masks for a single anisotropically etched silicon layers is discussed in [5] and determining the process for a given cross-section of a device is presented in [6]. In this paper, using the formalisms from our earlier work [7], we present a 3-D implementation of forward and inverse problems as well as a feature-based constraint modeler.

2 REVIEW OF MODELING FORMALISM

We developed a modeling formalism that is comprehensive for systematically implementing the forward problem and is also amenable for solving the inverse problem. We briefly review it here and refer to our earlier work [7] for a detailed description. In what follows, we assume that a full description of the process is known. This includes the number of layers (structural and sacrificial), number of process steps, type of deposit (conformal, stacked, via, or planar), type of etch (isotropic or anisotropic), type of doping (masked or unmasked), and the quantitative information such as thickness for deposits and depths and the list of affected layers for etching and doping.

2.1 Notation

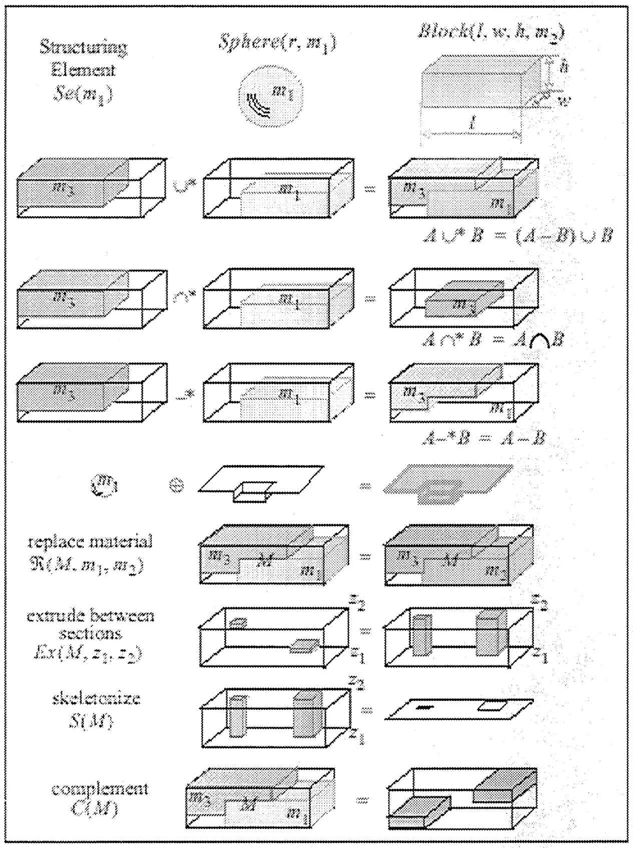
The heterogeneous geometric model for surface micromachined MEMS is a multi-material point set $M = \{P, m\}$ where P denotes the set of points and m the corresponding material tags. M is the union of several layers denoted by $M = \cup L_j$, where $L_j = \{P_j, m_j\}$ denotes jth layer. Furthermore, M_i denotes the device model up to the ith step. Thus, ith step (whether it is deposition, etching, or doping) operates on M_i to give the updated model to M_{i+1}.

2.2 Operators

We define two categories of operators viz. the *state change operators* and *query operators*, which are illustrated in Figures 1 and 2 respectively. The use of these operators is illustrated in Sections 3 and 4 for the forward and inverse problems.

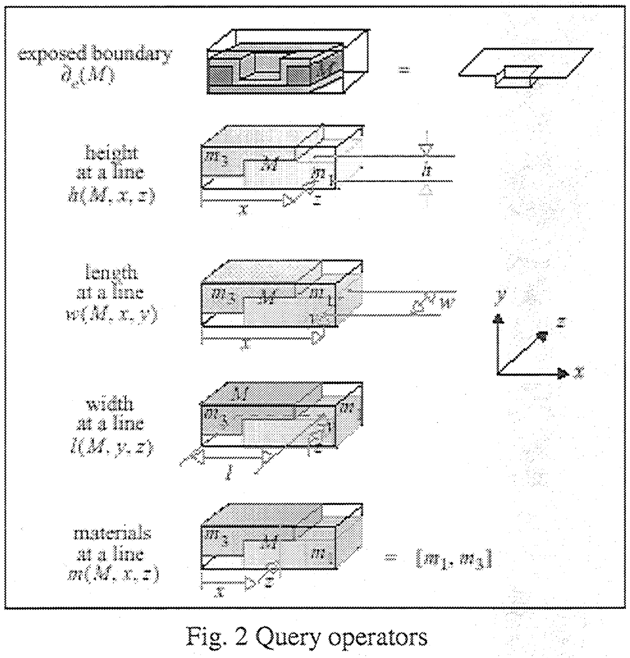
3 FORWARD PROBLEM

Given a process description and mask layouts, the solid model can be constructed by sequentially following the process steps. The state change and query operators help us define the steps unambiguously and comprehensively. For



Structuring Element $Se(m_1)$
$Sphere(r, m_1)$
$Block(l, w, h, m_2)$
$A \cup^* B = (A-B) \cup B$
$A \cap^* B = A \cap B$
$A -^* B = A - B$
replace material $\Re(M, m_1, m_2)$
extrude between sections $Ex(M, z_1, z_2)$
skeletonize $S(M)$
complement $C(M)$

Fig. 1 State change operators

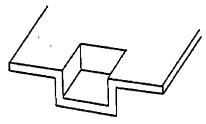
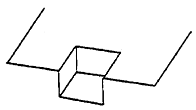


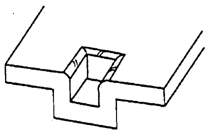
exposed boundary $\partial_e(M)$
height at a line $h(M, x, z)$
length at a line $w(M, x, y)$
width at a line $l(M, y, z)$
materials at a line $m(M, x, z)$
$= [m_1, m_3]$

Fig. 2 Query operators

example, the conformal deposit step is defined mathematically and is illustrated graphically as shown in Fiure 3. As another example, consider the description of the anisotropic vertical etch:

$S =$ sphere structuring element of radius $2d_i$ of material m_i $\textcircled{\#}$

$M_i =$

$\partial_e M_i =$



$S \oplus \partial_e M_i =$

$$M_{i+1} = deposit(M_i, confrmal, d_i, m_i) = (S \oplus \partial_e M_i) \cup^* M_i$$

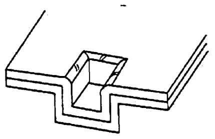


Fig. 3 Conformal deposit step

$$M_{i+1} = Etch(M_i, \tilde{L}, mask_i, d_i)$$
$$= for\, k = 1,2,....N$$
$$\Re(\tilde{L}_k \cup^* \{S \oplus (Ex(C(mask_i), 0, z_{\max}) \cap^* \partial_e M_i)\},$$
$$m_b, m_w)$$
$$where\ \tilde{L} = list\ of\ N\ affected\ layers$$
$$mask_i = mask\ for\ the\ i^{th}\ step$$
$$d_i = depth\ of\ etch\ in\ the\ i^{th}\ step$$
$$S = a\ vertical\ line\ structuring\ element\ of$$
$$height\ d_i\ made\ of\ material\ m_b$$
$$z_{\max} = maxium\ height\ until\ i^{th}\ step$$
$$m_w = white\ (empty)\ material \qquad (1)$$

It is important to note that actual implementation may differ from the way a step is mathematically described as long as the intended operation is carried out correctly. Therefore, the mathematical description should be viewed as a guideline for developing the implementation procedures.

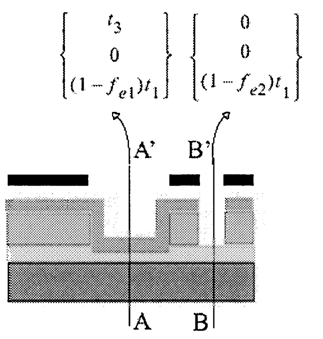
4 INVERSE PROBLEM

The inverse problem is solved by reconstructing an intermediate model R after each step and comparing it with the given model for which the masks are to be generated. In this process, we need to resolve some conflicts that arise, for example, when a layer is etched in two different steps. The mask openings are generated as follows. As shown in

Figure 4, a check line is scanned to determine the thicknesses of various layers along the check line. It is represented as a column vector and is recognized as the linear combination of "basis" vectors defined for each step. As an example, a three-layer process with three etches leads to:

$$\begin{Bmatrix} * \\ * \\ * \end{Bmatrix} = c_1 \begin{Bmatrix} 0 \\ 0 \\ t_1 \end{Bmatrix} + c_2 \begin{Bmatrix} 0 \\ 0 \\ -f_{e1}t_1 \end{Bmatrix} + c_3 \begin{Bmatrix} 0 \\ t_2 \\ 0 \end{Bmatrix} + c_4 \begin{Bmatrix} 0 \\ -t_2 \\ 0 \end{Bmatrix} + c_5 \begin{Bmatrix} t_3 \\ 0 \\ 0 \end{Bmatrix} + c_6 \begin{Bmatrix} -t_3 \\ 0 \\ -t_1 \end{Bmatrix} \qquad (2)$$

The left hand side is a result of a check line. The binary constants c_1 to c_6 determine if there is a mask opening corresponding to one of the three masks. This linear system of equations can be solved to obtain all possible solutions. See [7] for details.



$\begin{Bmatrix} t_3 \\ 0 \\ (1-f_{e1})t_1 \end{Bmatrix}$
$\begin{Bmatrix} 0 \\ 0 \\ (1-f_{e2})t_1 \end{Bmatrix}$
A'
B'
A
B

Figure 4 Checkline method for the inverse problem

5 IMPLEMENTATION USING VOXELS

The 2-D implementation using pixels was described in [7]. Here, we describe how the forward and inverse problems are implemented in 3-D. We use voxels to represent the point set $\{P, m\}$. The size, location, and material information about each voxel is stored in a data structure that is amenable for state change and query operators. Flood filling algorithm [8] and Boolean operations are the main tools in implementing various steps. We describe here a few generic operations.

For the conformal deposit, in one of our implementations, the exposed boundary query operation $\partial_e M$ is done as follows. A closed bounding box is created around the model with some empty space above it. A seed cell of black color is started in the 3-D space above the model and is recursively propagated until no more empty cells are found. In the recursive process, whenever a colored cell is encountered, it is marked as an exposed boundary. The Minkowski operation (denoted by $\oplus$) is done by scanning a sphere structuring element along the exposed boundary. A stack deposit is implemented by finding the top-most cell height and knowing the thickness of the deposited layer, a rectangular prism of voxels is created with appropriate material and added to the model. The via deposit operation uses the flood-fill algorithm. The isotropic etch is similar to the conformal deposit with the difference that the sphere of "empty" material is swept across the exposed boundary. For an anisotropic vertical etch, from a given mask, a rectangular prism of voxels of suitable size is created and is subtracted from the model.

Figures 5a-5d illustrate the modeling of sample process steps of a conformal deposit, anisotropic vertical etch followed by an isotropic etch.

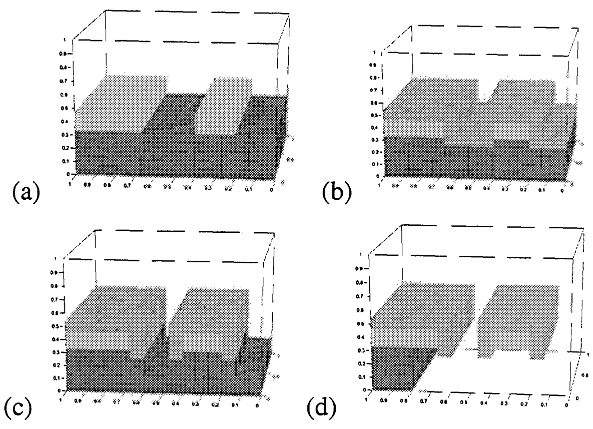


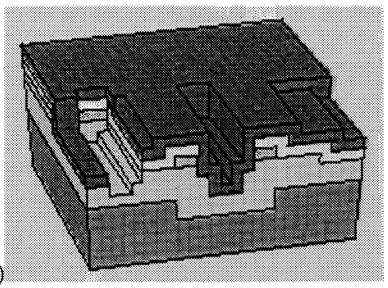
Figure 5: Illustration of the forward problem implemented in MATLAB

In the inverse problem, potential masks are generated by comparing layers (voxels) in model M to the corresponding layers in a reconstructed intermediate model R. A potential mask consists of a number of distinct, unconnected openings, which are extracted into separate masks using a flood-filling algorithm. Each mask is checked using the method described in Section 4 to ensure whether the mask needs to be retained or not. The masks are then used in reconstucting the model R until all masks have been generated. The inverse problem implemented in Microsoft Visual C++ and MFC is illustrated in Figure 6. Note in particular that two mask sets (Figure 6b) generate the same model shown in Figure 6a.

6 FEATURE-BASED MODELING

Our feature-based modeler comprises six modules: (1) the process generator module, (2) the user-defined feature generator module, (3) the design rule generator module, (4) the design-by-features module, (5) the manufacturing data generator module, and (6) geometry generator module. The surface micromachining fabrication steps are systematically mapped to the design features

The process generator module allows users to define a new surface micromachining processe or upload a known process into the process database. The user-defined feature generator module allows the user to define design features and specify the semantics of the defined features so that design intent is preserved. The feature database provides the designer with a set of intuitive design features with which to build the MEMS device. The design rule generator module allows users to define design rules for a given surface micromachining process or upload known design rules into the design rule database. The fabrication data generator module takes care of automatically generating the masks given a geometric model of the MEMS device and the geometry generator module takes care of archiving the geometric model. Finally, the design-by-features module, which is the core of the design tool, provides a user-interface that allows the user to intuitively build a model of the MEMS device without being concerned with the constraints imposed by the fabrication process.

(a)


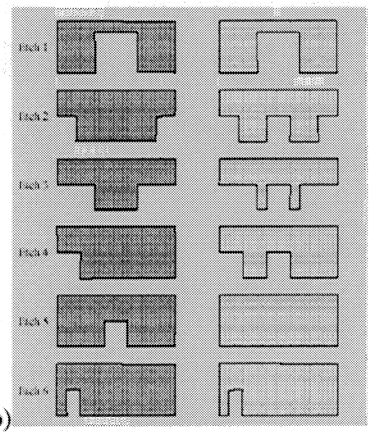
(b)


Figure 6: An example of the inverse problem (two mask sets generate the same 3-D model for the same process.)

Some examples of design features include protrusions, depressions, dimples, and undercuts, which a MEMS designer can easily relate to (Figure 7). A feature-based design tool was developed using Microsoft Visual C++ and MFC. A model of a MEMS micromotor to be fabricated using a three-layered surface micromachining process, designed using this tool, is shown in Figure 8.

7 CONCLUSIONS

In the geometric design of MEMS, building models from the mask (forward) and generating masks from a model (inverse) for a given process are both necessary. A voxel-based 3-D implementation of the forward and inverse problems is presented here. In order to solve the inverse problem, the model should be compatible with the chosen process. It is ensured in a feature-based modeler using which a MEMS designer can build models directly without first creating masks.

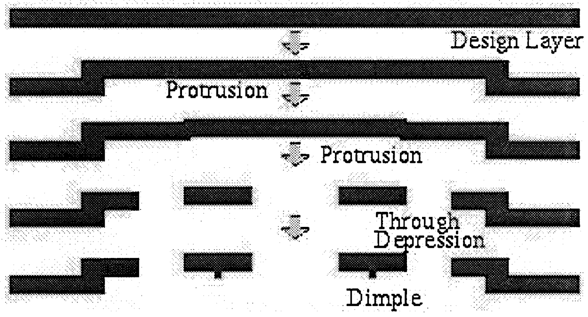


Figure 7 Feature-based design of a layer

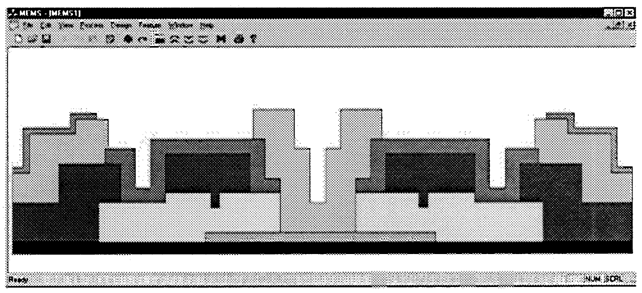


Figure 8: Micromotor built directly using features

ACKNOWLEDGMENTS

The authors are grateful to the National Science Foundation grants (DMI-997059 and DMI-9970021) that supported this work.

REFERENCES

[1] IntelliSuite, Intellisense, Inc., www.intellisense.com

[2] MEMCAD, Microcosm Technoogies, Inc., www.memcad.com

[3] MEMSCAP, Memscap, www.memscap.com

[4] MEMS-Pro, Tanner esearch, wwwtanner.com

[5] Ma, L. and Antonsson, E., "Automated Mask-Layout and Prcoess Synthesis for MEMS," MSM 2000, San Diego, March 27-29, pp. 20-23.

[6] Hasanuzzaman, M. and Mastrangelo, C.H., 1996, "Process Compilation of Thin Film Microdevices," IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.

[7] Venkataraman, V., Sarma, R., and Ananthasuresh, G.K., 2000, "Part To Art: Basis for a Systematic Geometric Design Tool for Surface Micromachined MEMS," CD-ROM proceedings of the 2000 ASME Design Engineering and Technical Conferences, Baltimore, Maryland, September 10-13, 2000. Paper number 14251.

[8] Lieberman, H., 1978, "How to Color in a Coloring Book," SIGGRAPH 78, ACM, Atlanta, GA, August, 1978, pp. 111-116.