THE AUTOMATIC SYNTHESIS OF PLANAR FABRICATION PROCESS FLOWS FOR SURFACE MICROMACHINED DEVICES

B. Gogoi, R. Yuen, and C. H. Mastrangelo

Center for Integrated Sensors and Circuits
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109-2122, USA

ABSTRACT

A method which automatically generates fabrication sequences for surface micromachined structures starting from a two dimensional geometrical description is presented. This method translates the device geometry into layers and a mathematical representation of layer order. It is shown that all possible process sequences are extracted from the layer order in terms of fundamental processing steps like deposition, lithography, and etching using topological sorting techniques. In general, the fabrication sequence is not unique. Hence, an optimal sequence is selected from the set using a cost function based upon a database of materials and processes. A synthesis program that implements the sequencing algorithm and optimization was developed. The output of the synthesizer is a complete optimal fabrication sequence in human-readable form.

INTRODUCTION

Current computer simulation tools for planar microfabrication such as SUPREM, SIMPL, MEMCAD, and OYSTER [1, 2, 3, 4] take a description of the fabrication process flow and a mask set of a device as inputs and generate accurate simulated representations of the finished device. These design tools are undoubtedly useful aids to the designer allowing him to correct potentially expensive mistakes before fabrication begins. Nevertheless, these programs are aimed at design verification, and require the input of a known fabrication sequence. This sequence is typically developed by an experienced designer which lacks formal methods. Systematic procedures for designing process flows do not exist to date.

In this paper a systematic means for the construction or synthesis of a fabrication sequence for surface micromachined devices is developed. This operation in effect is the reverse of simulation, as shown in Fig. 1. The synthesis problem consists of finding a correct sequence of process steps that converts a bare wafer onto the desired device using fundamental steps (lithography, deposition, etching). Mathematically, the structure on the wafer at any given step can be regarded as the state of a system \( S_i \) as shown in Fig. 2(a). The synthesis problem then reduces to that of finding the intermediate states \( S_i \) and the operations (or steps \( L_i \)) that transform one state onto another (Fig. 2(b)). Such problem is known as a path finding problem in artificial intelligence literature.

The internal organization of the program is shown in Fig. 3. The program is divided into an input processor, a synthesis core, and an output processor. The process synthesis core is composed of three main modules (a) the geometry smasher which decomposes the device structure into layers, (b) the sequence generator, (c) the process flow generator, and (d) the flow optimizer.

GEOMETRIC INPUT AND DECOMPOSITION

A graphical interface is provided to the user to input the geometrical description of the desired surface micromachined structure to the process synthesizer. The current version of the interface accepts manhattan structures. The interface checks that all input polygons are anchored (i.e. rest on other polygons) and automatically inserts sacrificial layers if required. The sacrificial layer geometry is determined using the dipping algorithm due to Weiler[5] because of its capability of handling both convex and concave polygons.

The augmented graphical information is fed to the smashing algorithm which test for polygons which can
be represented by conformal layers of uniform thickness. The conformality test is performed by dilating [6] the bottom profile of each polygon and using it as a template for the top profile. If there is no match, then the polygon is nonconformal, and it is resolved into elemental polygons of constant thickness using scan-line algorithms described in Foley[7]. The expanded set of polygons of constant thickness is the output of the polygon smasher.

**DIRECTED GRAPH AND PARTIAL ORDER REPRESENTATION**

The order of the output polygons from the geometry smasher can be represented as a directed acyclic graph (DAG). Each of the polygons corresponds to a node of the graph, and its arcs are determined by precedence relations established between the polygons. If in the original geometry, polygon a is above polygon b, then an arc is drawn in the graph from node b to node a. When such a precedence relationship is established, it is written in shorthand as b≺a. Each polygon is compared to each other until a complete DAG is established. The order of the polygons is then completely specified by the graph.

The structure of the DAG and the corresponding order relationship is described by its adjacency matrix A with elements \( a_{ij} \). Element \( a_{ij} \) is set to one if there is an arc from node i to node j in the DAG and zero otherwise. A simple example outlining this mathematical procedure is shown in Fig. 4(a). The adjacency matrix

\[
A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  

specifies the order only between adjacent nodes. Additional order relations are inferred from the transitive property of the DAG i.e. if node \( a \prec b \) and node \( b \prec c \), then node \( a \prec c \). All these relations are included in the transitive closure matrix \( T \) [8].

\[
T = \sum_{i=1}^{N-1} A^i
\]  

where the sum and power are Boolean operations. In graph-theoretic terms, this is equivalent to economi-
cally representing the path information in the directed graph [9, 10].

![Directed graph representation of a simple microstructure.](image)

The synthesis problem thus reduces to finding a sequence of nodes that includes all the polygons subject to these constraints. In mathematical terms, the DAG represents a partially ordered set (poset) of objects. The synthesis problem is equivalent to finding linear extensions of the poset. In this context, the DAG represents a combinatorial object. Once this connection between device geometry and partial orders is established, two very important observations are established: (a) there exist at least one process sequence for every 2D device, and (b) the number of process sequences \( n(S_N) \) is \( 1 \leq n(S_N) \leq (N-1)! \) where \( N \) is the number of nodes in the DAG.

The existence of linear extensions for a poset has been postulated by Szpilrajn [11] which is stated here - Any partial order on a set can be extended to a linear order of the same set. Since any 2D device cross-section can be represented as a poset, any device structure can be fabricated using a planar process. The enumeration and generation of linear extensions has been studied extensively [12, 13, 14, 15, 16, 17]. For the general case, the counting of the number of linear extensions is a NP-complete problem [18]. However, for finite posets, estimates have been made for the enumeration of all the linear extensions [12]. A reasonable upper estimate for \( n \) is

\[
n \leq \prod_{i=1}^{N-1} (N - i - C(i)) \tag{3}
\]

where \( C(i) \) is the sum of upper and lower constraints of node \( i \). In general, the number of sequences \( n \) is large. The set of all possible sequences is called the design space \( \mathcal{D}(S_N) \) of cardinality \( n(D) \). Finding the complete design space is highly useful since it allows us to select an optimal sequence and the most profitable process flow. The number of process flows increases with the number of polygons, but decreases with increased constraints. For some structures, the number of flows may be exceedingly large. Hence, the sequencing problem may become intractable. In order to detect these special cases, it is convenient to estimate the number of linear extensions ahead of time. The number of linear extensions of the poset \( P \) is

\[
n = N! \text{vol}(B(P)) \tag{4}
\]

where \( B(P) \) is an \( N \)-dimensional convex polytope defined by the constraints of the poset, and \( \text{vol} \) is the polytope volume [12]. An estimate for Eq. (4) can be rapidly calculated using Monte-Carlo techniques [19, 20].

**SYNTHESIS ALGORITHM**

The synthesis algorithm consists of three different parts: (a) a sequencing routine which generates all possible linear extensions of the adjacency matrix representing the device cross-section, (b) a layer extracting routine which groups polygons into layers, and (c) a flow extraction routine that adds the necessary patterning and definition steps to the layer sequence.

*Sequence Generation:* The generation of all the linear extensions of the poset is described in this section. For finite posets, the problem of finding all the linear extensions is known as topological sorting. The topological sorting problem is to imbed a given partial order into a linear (total) order. For the DAG representation, this is equivalent to arranging the vertices into a straight line such that all arcs go from left to right as shown in Fig. 4(b). In terms of the adjacency matrix \( A \) describing the partial ordering \( R \) on a finite, non empty set \( S \), this is equivalent to finding the permutations of the rows and columns of \( A \) so that it becomes an upper triangular matrix [21]. Using the notation of Knuth [22], the sequencing problem is formally described with its equivalent representations. Given a number \( N \), and a set of integer pairs \((i, j)\) where \( 1 \leq i, j \leq N \), the problem of topological sorting is to find all permutations \((x_1, x_2, \ldots x_N)\) of \((1, 2, \ldots N)\) such that \( i \) appears to the left of \( j \) for all pairs \((i, j)\) that has been input.

A number of topological sorting algorithms based on the sequential computation approach are described in the literature [21, 22, 23, 24, 25, 26] as well as algorithms which consider the problem in terms of an acyclic graph [9, 27, 28] and triangulation of the adjacency matrix [29]. Parallel computation algorithms [30, 31, 32, 33] have also been developed.

Since the number of topological sortings is a decreasing function of the number of relations in the par-
tial order, it is practical to generate all the linear extensions and search for a particular one. The sequential algorithm due to Steiner [20] which generates all possible linear orders was implemented. If \( n(P) \) denotes the cardinality of the poset \( P \) of \( N \) elements, the algorithm has a complexity of \( O(n(P).N) \). Steiner’s algorithm takes direct advantage of the fact that only the ideals (feasible subsets) are needed to generate all linear extensions since they exploit the structure of the poset. This method is very efficient compared to other sequential algorithms. Fig. 4(b) shows the some of the linear extensions of the poset of Fig. 4(a).

Layer Extraction: Using the \( N \) nodes (polygons) generated by the polygon processor as the input, the linear extension generation program generates all possible sequences in which the polygons can be deposited. The layer extraction routine attempts to group some of the polygons into layers. Polygons of the same material and thickness which are adjacent to each other in the sequence can be grouped into one layer. The mask information for each layer is obtained from the projection of all its polygons onto a plane. The output of the layer extraction routine is a sequence of layers, each one containing its corresponding polygons. The dotted lines in Fig. 4(b) show adjacent polygons that can be grouped.

Flow Extraction: The layer sequence only specifies the order of the structure and does not specify how to reconstruct the original geometry. The flow extraction routine scans the layer sequence and inserts appropriate deposition, definition (lithography), and patterning (etching) steps, thus creating a process flow for the fabrication of the device. The layer sequence is expanded according to the following rules. (a) In the process flow, there is one deposition step for every layer. (b) The patterning step for every layer must occur after its deposition. (c) The definition step for a layer must precede the patterning. (d) Every layer must be visible before its definition. The last condition can be determined by examining the blocking graph of the layer sequence [34, 35]. As a first step in the extraction, all layers are converted to depositions. The above rules define partial order relations between the order of the depositions and the patterning and definition steps. For each deposition sequence, all linear extensions which include deposition, patterning, and definition are generated using the same sequencing algorithm used for the polygons. Since the deposition orders is fixed by the layer order, this procedure is equivalent to merging of two ordered sets [36].

Because of visibility constraints, some of the flows generated by this procedure are not feasible. In the present implementation, invalid flows are discarded by a stack filter. The stack filter uses the fact that it is not possible to etch an underlying layer unless it is visible. The stack algorithm scans the flows as follows. Deposition steps are pushed onto the stack as they appear on the flow. Etch steps pop the corresponding depositions when they are on the top of the stack. Valid flows yield empty stacks.

The output of the flow extractor is a complete set of feasible fabrication flows described in terms of fundamental processing steps. The set of all feasible fabrication processes is the design space for the device. The structure of the flow extractor output is shown in Fig. 4(c).

Examples: The cardinality of the design space can be large even for a few polygons. The synthesis algorithm was tested on a number of structures using a SUN-Sparc 10 workstation. For the micromotor of Fig. 5 consisting of 9 polygons, the synthesizer produced 77 feasible process flows in 18 seconds with one process flow with the minimum number of 4 layers and 7 process flows with 5 layers. For the 18 polygon microbridge of Fig. 6, more than 300 feasible process flows were generated. Twenty-five of these flows had minimum number of 14 layers.

FLOW OPTIMIZATION

The output of the synthesis algorithm is the design space of feasible solutions from which an optimal process flow can be selected. The optimization algorithm selects one of the flows that minimizes a defined cost function. This problem is similar to precedence constrained scheduling [37, 38, 39, 40].

The nature of the optimal solution is strongly dependent on the cost function. Currently, the flow optimization algorithm uses a three-step cost function determination that maximizes the process yield. In the
first step, the algorithm discards flows which have incompatible materials. For example, it eliminates flows that require deposition of high temperature materials on top of low temperature materials. The cost function is defined as the inverse of the minimum selectivity against all the exposed materials of all the etchants in the flow. In the second step, process flows with cost that are below a minimum acceptable value are discarded. In the third step, a smooth cost function which includes weighted selectivities of all etching steps against all exposed materials is calculated, and the optimal flow from this set is selected. Deposition parameters, etch rates and selectivities are obtained from an internal database of etchants and materials. Further optimization is done by using the remaining flows that have the least complexity and number of layers.

PARAMETER ESTIMATOR AND OUTPUT INTERFACE

The output of the flow optimization algorithm is a process flow which qualitatively describes how the device is constructed. The parameter estimator calculates specific values for each step of the flow. The estimator calculates deposition, exposure, and etching times and specific conditions required for each step. The estimator uses information from the internal database to estimate the parameters.

The output interface translates this information into various formats. One of the tasks of the interface is to generate a process flow that is human-readable. The format of the output is generated automatically and each step is numbered appropriately as shown in the inset of Fig. 7. The output interface also generates SUPREM files for simulation of the process flow and is also easily adapted to Computer Integrated Manufacturing languages for downloading to semiconductor equipment. A yield estimator is currently under development which provides yield and cost information as well as expected statistical variations to the user. Currently, only the printed output is fully implemented.

The output interface also displays the cross-section of the device after each process step. This enables the user to verify each step of the fabrication sequence. The output of the graphical interface is shown in Fig. 7. Clicking on any of the process steps menu entries displays the corresponding cross-section at that step. A movie button dynamically displays all the process steps in sequence.

CONCLUSIONS

A synthesis method for the automatic generation of fabrication process sequences of surface micromachined structures from a two-dimensional description has been developed. The method uses topological sorting techniques to find all possible flows for the device. It is shown that the set of feasible flows is large. Hence, an optimal flow is selected which maximizes the device yield. The method was implemented in a program which uses a graphical interface to input the device cross-section and output the process flow data. The fabrication process flow is described by fundamental processing steps as deposition, lithography, and etching. Future work will include the incorporation of reactive growth, diffusion, and ion-implantation steps.

ACKNOWLEDGEMENTS

Part of this research was sponsored by NSF contract ECS-9309229. We thank Mr. K. Kalita of the EECS Dept. for helpful discussions about the geometric algorithms.

References


