Proceedings of DETC98
1998 ASME Design Engineering Technical Conferences
September 13-16, 1998 --- Atlanta, Georgia

DETC98/MECH-5838

FEATURE-RECOGNITION FOR MEMS EXTRACTION

Bikram Baidya
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA
bbaidya@ece.cmu.edu

Satyandra K. Gupta
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
skgupta@ri.cmu.edu

Tamal Mukherjee
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA
tamal@ece.cmu.edu

ABSTRACT

Microelectromechanical systems (MEMS) integrating multi-domain sensors and actuators with conventional microelectronic batch fabrication processes are becoming increasingly complex. In order to design systems with large numbers of multi-domain components, we need to use a hierarchical structured design approach, with design at the schematic level instead of the traditional layout representation used in MEMS design. However, since fabrication can only be done from a layout representation, an automatic or manual layout generation from schematic is necessary. It is essential to be able to translate from the layout representation back to the schematic to reason about layout correctness in meeting the schematic's function as well as to extract geometric parameters for functional simulation. An extraction module is developed which reads in the geometric description of the layout structure and reconstructs the corresponding schematic. This schematic can then be fed to an ordinary differential equation solver or can be compared with the design schematic to validate the correctness of the designed layout. The extraction module also minimizes the number of nodes required to represent the schematic as a netlist. The results presented show the success of the module for some example MEMS designs.

1. INTRODUCTION

Microelectromechanical systems (MEMS) are sensor and actuator systems made from microelectronic batch fabrication process. The advent of stable MEMS fabrication processes has recently led to the development of increasingly complex MEMS designs. The acceptance of MEMS in industry has been quite smooth and fast and today we already have commercial products like accelerometers to trigger airbags from Analog Devices [1] and Motorola [22], and Texas Instruments' micro mirror display devices [12].

The MEMS designer has traditionally completed his task manually. He has had access to only a few tools to help control the design and fabrication process complexity. The understanding of the fabrication process and the capability to visualize the final product was a matter of designer's experience. With no tools present to test the final layouts, the design of MEMS devices has been primarily a trial and error cycle requiring numerous design and fabrication iterations. Thus the design cycle for a MEMS device is considerably longer than its VLSI electronics counterpart. Initial CAD tools for MEMS primarily focussed on accurate 3-D simulations of simple micromechanical devices. Later self consistent analysis tools [7][11], which integrated both electrostatic and mechanical aspects of MEMS, were also developed. These tools focused on device behavior, given a 3D model of the device after fabrication. Since the generation of 3D device models was tedious, process simulation tools [15] were also devel-

oped. Though these helped the designer to get a preview of the device being designed, they were restricted to simple MEMS devices only. This was mainly due to the time and memory requirements for these simulations. This led to the development of structured design methodologies for MEMS [4][23][24] for managing the increasingly complex MEMS devices being imagined. During the last decade there has been a considerable development in this field, resulting in numerous CAD tools for mixed technology lumped parameter simulation [34][35] and for layout synthesis [9][16]. While the schematic to layout flow was being enriched by such tools, the reverse flow, i.e., from layout back to schematic, was still untouched. The need for a CAD tool to do this conversion arises from the fact that this would allow the layout designer to check the design faster. The present options available to the designer are to either perform a finite element analysis, which is very slow, or to go into actual fabrication and then check the designed structure, which is an expensive waste if the design fails. Thus came the need for a MEMS device extractor which would give a reconstructed schematic, corresponding to the designed layout, which can then be simulated and checked easily. The present work attempts to fill this gap in the structured MEMS design methodology.

Schematic-level extraction in MEMS expands the geometric specification of the device, in accordance to the selected process, into a layer-by-layer description of the geometrical shapes and their topological placements, just as it does in VLSI [20][31]. The finer objective for such an extractor is to have a schematic representation of the layout. This objective is achieved in a two step process. First we process the machine readable layout specification to extract atomic mechanical features such as beams, plates, joints and mechanical gaps, from which the MEMS device is made. The next step is to extract commonly-used MEMS components by geometrically matching the topology and connectivity of these atomic elements with those in a component library. The resulting schematic representation can now be compared with the original designer's schematic to determine layout errors using a tool for Layout-versus-Schematic. In addition, the extraction generates lumped-parameter values for the atomic elements it extracts to enable lumped-parameter functional simulation [34][35] of the MEMS layout.

The motivation behind this exercise is to have a rapid, efficient and accurate analysis of the layout in order to develop high quality robust devices and systems. The ability to ensure layout correctness via such an extraction methodology will significantly shorten the time, and reduce the cost, of MEMS design.

The next section provides a background of existing MEMS technology, devices, and design process. Section 3 overviews our extraction methodology. Section 4 describes the algorithms used for feature recognition and extraction. Section 5 presents results from a prototype implementation of the extraction algorithms, and is followed by a concluding section.

2. BACKGROUND

2.1. MEMS Technology

Microelectromechanical systems are integrated systems combining electrical and mechanical components. They vary in size from microns to a few millimeters. The whole aim of research in this field is directed towards using fabrication processes for the parallel manufacture of many MEMS devices. This is motivated by the cost benefits that such integration would bring. There are three major technologies [6][13][14][30] used in MEMS fabrication: bulk micromachining, LIGA and surface micromachining. As in the VLSI world, silicon technologies tend to be the most widely used technology in MEMS fabrication. This is not only due to the fact that silicon technology is already very advanced, but also due to remarkable mechanical properties of silicon and its variants (like polycrystalline silicon), like lack of mechanical hysteresis, high modulus of elasticity, high strength to weight ratio, high thermal conductivity and low thermal expansion coefficient.

In the bulk micromachining technique, mechanical structures are etched out of the bulk of the silicon wafer. It is the most mature micromachining technique. Though its initial investment is quite low, it suffers from the inherent disadvantage that the planar geometry of the structures must always be rectangular. Integration with electronics is also difficult, though not impossible. The LIGA (German acronym for LIthographie, Galvanoformung, Abformung) technique involves X-ray lithography, micro-electroplating and micro-molding processes. It begins with X-ray lithography on a conductive substrate followed by preferential electroplating of the gaps between the resist patterns. Though LIGA is suitable for mechanical structures like microrelays, micromotors, etc., its integration with electronics is difficult. It also involves a high capital investment and initial cost. Within the last decade, surface micromachining techniques have had a phenomenal growth. We will focus our attention to this technique. Unlike its bulk cousin, surface micromachining does not penetrate the bulk silicon wafer. Instead thin films are selectively deposited on and/or removed from its surface. Traditional surface micromachined processes do not allow for integration of MEMS with electronics (e.g., MCNC's Multi-User MEMS Process, MUMPS [18]). The need to electronically process the signals generated by MEMS sensors, and/or electronically control MEMS actuators has led to the development of integrated surface micromachining processes. Currently, the most viable integrated MEMS fabrication alternatives in the U.S. are polysilicon processes from MCNC (Multi-User MEMS Process service with flip-chip CMOS, SmartMUMPs) [18], Sandia National Laboratories (Sandia Agile MEMS Prototyping, Layout tools, and Education program, SAMPLE) [27][28], and Analog Devices' iMEMS^{TM} process [2], and the CMU standard CMOS process [8].

Figure 1: A folded flexure comb-drive microresonator fabricated in the MUMPS process.

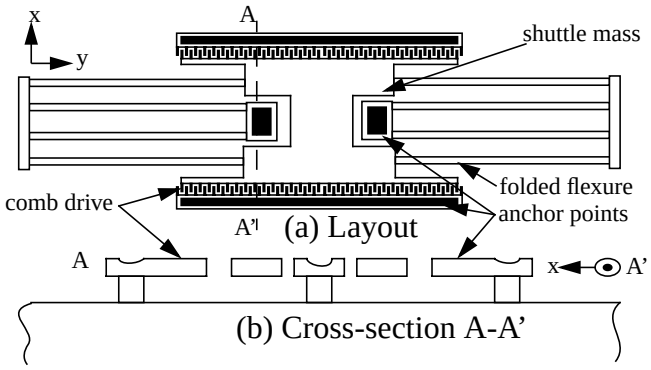
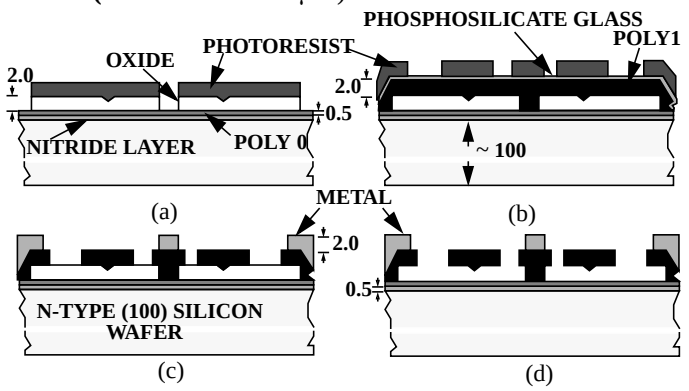


Figure 2: MUMPS process steps highlighting (a) first sacrificial oxide layer, (b) first structural polysilicon layer, (c) metal layer on the polysilicon surface and (d) the final release. (all dimensions in μm)



We will focus on the MUMPS [18] process due to its simplicity, popularity, and maturity as a surface micromachining process. Extensions to the newer integrated processes are easily accomplished. An example MEMS device layout and a view of its cross section on the A-A' axis is shown in Figure 1. As can be seen in the figure, the device consists of a floating structural layer that is attached to the substrate by anchors. The device itself will be described in Section 2.2. Figure 2 details the process steps that leads to the fabrication of such devices. We focus on how the cross-section A-A' of Figure 1 would look at different points of the fabrication process. First a layer of low-stress silicon nitride is deposited on the substrate to form an electrical insulation. This is followed by a layer of low stress polysilicon which is patterned and etched to form electrical interconnects. A sacrificial layer of oxide is then deposited and patterned to get the dimples and the first anchor holes. This stage is shown in Figure 2(a). This is coated with a layer of phosphosilicate glass followed by another layer of polysilicon which is then etched to form the first structural layer. The photoresist pattern needed for this etch step is shown in Figure 2(b). Finally the metal layer is deposited and patterned to form the interconnects and pads (Figure 2(c)). The sacrificial oxide layers are etched out and the resulting structure contains the free mechanical device (Figure 2(d)).

The rest of this paper makes numerous references to various mask layers. We will use the mask conventions laid down in MCNC's SmartMUMPS Design Handbook. Table I lists all the layers that we will refer to.

TABLE I Mask conventions used in the MUMPS process

Pnemonic Level Name	Purpose
POLY1	pattern for first structural polycrystalline silicon layer (poly1)
ANCHOR1	open holes for poly1 to nitride or base polycrystalline silicon layer (poly0) connection. The elements fabricated after this stage are connected to the nitride/poly0 layer and thus are not floating.
HOLE1	provide release holes for poly1structure. The etchant flows through these holes and thus better and uniform etching is possible
DIMPLE	create dimples/brushings for poly1structure

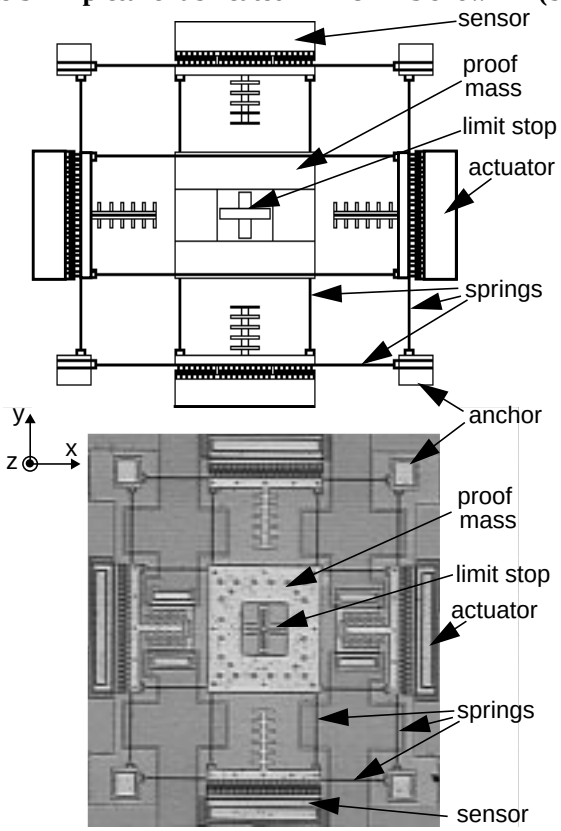
2.2. MEMS Devices

MEMS devices find numerous applications in different areas, generally functioning as miniature sensors or actuators. MEMS technology allows us to integrate many devices into a single system comprising electrical, mechanical, thermal, optical, chemical and biochemical components. This results in their low per unit cost and small size which makes them suitable for portable and remote applications. We focus on inertial MEMS sensors because of their numerous applications, and their relative maturity. In particular, we describe the MEMS microresonator, since it forms the basis of most inertial sensors. We also describe the gyroscope because of the keen research interest in that device.

The microresonator [29] shown in Figure 1 is a typical example of a folded-beam flexure resonator. The folded flexure is a popular design choice for the suspension because it is insensitive to buckling (arising from residual stress in the polysilicon film). The resonator is driven in the preferred (x) direction by electrostatic actuators that are symmetrically placed on the sides of the shuttle. Each actuator, commonly called a 'comb drive', is made from a set of interdigited comb fingers. When a voltage is applied across the comb fingers, the shuttle mass moves due to electrostatic forces. Thus by applying a sinusoidal voltage we may have a periodic motion of the resonator. If the frequency of this motion resonates with the natural frequency of the resona-

tor, the device will operate at resonance. This natural frequency is a function of the stiffness of the folded-beam flexure and the shuttle mass. A resonator structure can also be excited by the application of an inertial force, with the comb-drive now acting as a capacitive sensor, thus resulting in an accelerometer.

Figure 3: Three-fold symmetric gyroscope design with the electrostatic drive along the y-axis, and a capacitive displacement sensor along the x-axis. Layout shown in (a), and its SEM picture fabricated in MUMPS shown in (b).



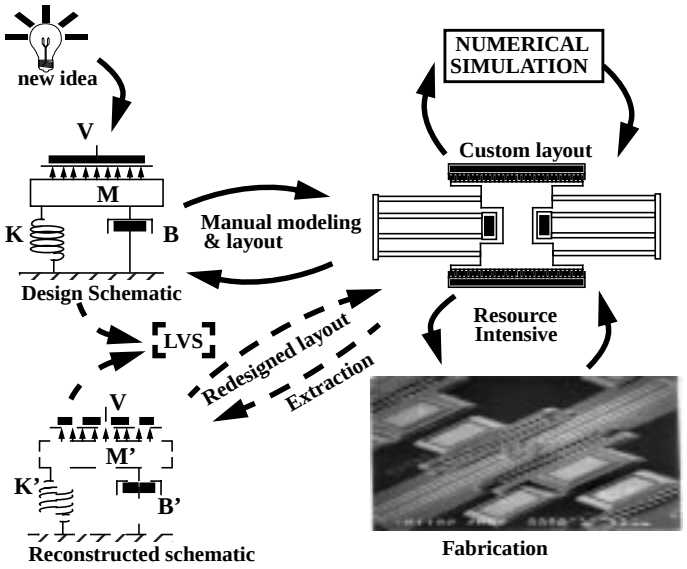
Vibratory-rate surface-micromachined gyroscopes [19] use an oscillating proof mass suspended by a set of springs. When the mass moves with a constant velocity in a fixed frame of reference, it experiences an external rotation which appears as an acceleration (Coriolis acceleration) in the local frame. This acceleration results in an inertial force, the Coriolis force, which is, in most cases, sensed capacitively. The three fold symmetric vibratory-rate gyroscope (Figure 3(a)) consists of a set of identical set of springs that are placed symmetrically about a central mass. These springs are anchored at one end and connected to the mass at the other end in a rolling pin condition. When the mass is forced to oscillate in the driven mode, say along y-axis, the Coriolis force induces oscillations in the sensed mode, along x-axis. The micromechanical implementation is shown Figure 3(b).

Other surface micromachined devices [6] include micromirror arrays [21][26] from Texas Instruments Inc. [12]; microaccelerometers from Analog Devices [1], Motorola [22] and photo detector arrays from U. C. Berkeley [17]. The list is never complete and new devices are continuously being added to the list.

2.3. Overview of MEMS Design Process

The current MEMS design practice (Figure 4) is very time consuming. It starts off with the designer making a rough sketch of the schematic of the design, shown in the top left of the figure, and very basic equations to ensure feasibility of the design. After being satisfied with the schematic, the designer proceeds to physical layout. At this step the only tool available to the designer to check the layout is numerical simulation (top right in the figure). The problem with numerical simulation is that it is prohibitively slow and interpretation of the results is tedious and requires a lot of expertise. Thus in many cases the layout is sent to fabrication without proper checking, resulting in non functional devices. Small errors detected in the fabricated device are then used to redesign the layout. Thus a fabrication-design loop (bottom right in the figure) is set up which is very expensive.

Figure 4: Present and proposed (in dotted line) design flow



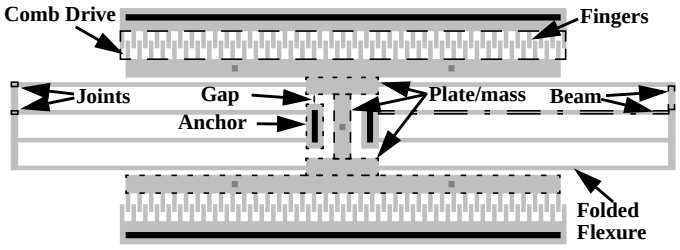
A need for structured MEMS design process [4][23][24], akin to that in VLSI, was felt. This meant that CAD tools were needed to perform the task at each level of design. Synthesis tools with powerful optimization techniques [9] were introduced for MEMS cell-level design and layout optimization. Optimized synthesis and etch simulation tools were also developed for design of compliant mechanisms [3], which are flexible structures that generate a wide variety of mechanical motions through elastic deformations. In addition to these synthesis

tools, lumped parameter simulation models [34][35] are also being explored. While the top to bottom flow, i.e., from design schematic to layout was seeing these changes, nothing much was being done on the reverse flow. This is necessary to verify the designed layout. Our work addresses this problem by developing a device extractor for greatly simplifying the task of design checking. By reconstructing the design schematic from the layout the designer will be able to perform faster simulations on the reconstructed schematic and also compare it with the design schematic, thus replacing the fabrication iteration loop with a faster and less expensive schematic-layout loop (shown with broken lines in Figure 4). Detection of minor errors in the layout, like missing connections can also be done very easily during the comparison between the reconstructed and designed schematic. In summary, the final aim is to make the task of design verification as smooth and easy as possible.

3. OVERVIEW OF THE EXTRACTION PROCESS

We first focussed on the problem of extracting polysilicon surface micromachined MEMS built in MCNC's MUMPS process. Once the methodology is established for this extraction we intend to extend our scope to other surface micromachined MEMS processes, like the laminated oxide/aluminum MEMS built using MOSIS [8]. The extraction process begins with the device layout, which is often described by the polygonal geometry that makes up the device.

Figure 5: Extractable elements in a polysilicon comb-drive resonator



While the key idea has been borrowed from the extraction process used in VLSI world [5][20][31][36], there are a number of differences. Unlike VLSI, in MEMS the shape, size and position of an object is of utmost importance and plays a crucial role in deciding what element it is. While in VLSI, recognition is done mostly by detecting the overlaps between different layers; in MEMS overlap of layers plays only a secondary role. Here the main task is to recognize the elements based on its features (shape, size and placement). The data structure that is most opted for in VLSI extractor designs [32][33] is a list of all non vertical edges, sorted first according to their abscissa followed by their ordinates and lastly by their slopes. These edge-based data structures make overlap detection simple but are not computationally friendly for shape detection. Therefore we use a list of polygons, since it eases the task of shape recognition. Since our detection loops are to be run for all the polygons it was found unnecessary to use other complex data structures like quad trees. The only positional information ever needed is about neighboring polygons and this is easily solved by maintaining pointers to the neighbors. In effect our representation is a hybrid of linked lists and corner stitching.

The extraction process can be broken down to two stages; first being detection of atomic elements and second being detection of commonly used device components. Figure 5 highlights both the atomic and component-level features that can be extracted from a MEMS layout (a folded-beam microresonator, like the one we saw in Figure 1). In the process of recognition of the atomic elements it was found necessary to represent the given layout in a unique manner. Thus a canonical form of representation is used for any given layout. This is discussed in more detail in Section 4.1. Feature based recognition is then used to detect the various atomic elements. Information from other layers, like location of first anchor holes from the ANCHOR1 layer, have been used to detect possible locations for holes and anchor. This information is technology specific and must be read in from the accompanying process description file. The final recognized set is then optimized to reduce the total number of nodes required to represent it as a netlist. Using the information contained in the recognized set we generate a netlist which can then be compared with the original design netlist. Thus our primary objective of having a check on the designed layout can be met. Device function can also be confirmed by running an ordinary differential equation solver on this netlist to simulate device performance.

4. DESCRIPTION

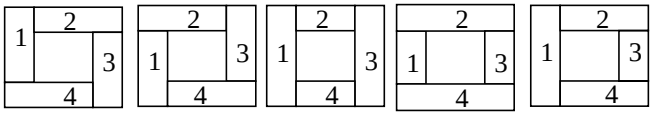
This section describes the algorithms used in the extraction module. First we describe the representation that we use to achieve uniqueness. This is followed by the description of the algorithms used to detect the atomic elements. Section 4.3 and Section 4.4 describes the optimizations done to improve the efficiency of the global design flow (both extraction and subsequent simulation). Section 4.5 speaks of the final aim of the routine, i.e., to generate a netlist which can be used to run simulations. We limit our discussion to Manhattan layouts in the following sections for simplicity of description.

4.1. Canonical Representation

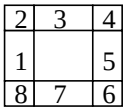
The input to the extractor is the designer's layout. The layout design is a reflection of the designer's style. Thus even if two designers do the same design, final designs might differ. This poses a serious bottleneck for recognition. To overcome this problem, we utilize a representation which will be unique for a given design. We define this representation to be the canonical representation for the layout. It is a representation

which uses minimum number of rectangles to cover the given layout area, such that infinitesimal outward extensions of an edge of any rectangle never intersects with the interior of the layout area. We use the term *layout area* to define the area which represents the actual device in the layout, i.e., it is the interior area(s) defined by the boundary/boundaries of the geometrical representation of the device in the layout. Thus in the canonical representation, the layout is made up of small rectangles such that each rectangle has *only one neighbor per edge*. This can be achieved very easily by extending the boundary edges into the interior of the layout area till it meets another boundary edge.

Figure 6: Canonical representation



VARIOUS REPRESENTATIONS FOR SAME STRUCTURE



CANONICAL REPRESENTATION

Since we are dealing with Manhattan geometry only, our task is to canonize polygons whose edges lie along one of two orthogonal coordinates. Figure 6 explains our idea of canonical representation. We further assume that the input is in the form of rectangles, i.e, a rectangle cover for the layout is supplied to us. Given this as input we proceed to canonize the structure using the algorithm described below. Before taking in the input file we need to flatten the chip's hierarchical description (written, perhaps, in CIF). This serves as the input to the main function which reads out only the rectangles that are mentioned in the POLY1 layer, i.e, reads out the first polysilicon structural pattern. This serves as the input to the subroutine to create the canonical representation. The final canonical representation is built up sequentially. The primary interaction takes place between two sets; the input set and the output set. The output set will eventually contain the canonical version of the input set. The output set is always kept in canonical state with respect to its contents. Elements from the input set are selected sequentially and added to the output set. Whenever there is an addition to the output set, its equilibrium might be destroyed (i.e, the output set might no longer be a canonical set). If this occurs then it sparks off a series of operations which ultimately brings the output set back to its equilibrium or canonical state. This is repeated till the input set is exhausted and at this point the output set will contain the canonical representation of the input layout. The process which drives the output set to equilibrium, after it is disturbed by an insertion of a new element, is described in Figure 7.

Figure 7: Algorithm to canonize a layout

1. Let **R** be the input set.
2. Let **G** be the output set ***initialized*** to a **NULL** set.
3. Let us randomly ***pick*** an element **r** from **R** and ***initialize*** a working set **P** by adding **r** to **P.**
4. **Q = {x| ADJ(x, r); x** is an element of **G}**
   ADJ(x, r) is an operator which returns those elements **x** (**x** is an element of **G**), which are adjacent to **r**
5. ***for all*** **q** ***in*** set **Q**
   for all **p** ***in*** set **P**
   $\mathbf{V_q}$ = set of vertices of **Q**
   $\mathbf{E_p}$ = set of edges of **P**
   if **CON($\mathbf{V_q}$,$\mathbf{E_p}$)** ***then*** **SPLIT(p,q)**
   Function **CON($\mathbf{V_q}$, $\mathbf{E_p}$)** ***returns*** **TRUE** if there exists a pair **v** in $\mathbf{V_q}$ and **e** in $\mathbf{E_p}$ such that **v** lies on or is contained by **e**. Function **SPLIT(p, q)** splits **p** by the edges of **q**.
6. ***for all*** **p** ***in*** **P**
   for all **q** ***in*** **Q**
   if **CON($\mathbf{V_p}$, $\mathbf{E_q}$)** ***then*** **SPLIT(q,p)**
7. ***while*****(Q!= NULL)**
   Q' = {x|ADJ(x, q), x is an element of **G}**
   for all **q** ***in*** **Q**
   for all **q' *in* Q'**
   if **CON($\mathbf{V_q}$, $\mathbf{E_q}$)** ***then***
   SPLIT(q', q) ***and***
   Q =Q'
8. **G = G $\cup$ P**
9. **R = R - {r}**
10. ***if*** **R!= NULL** ***then go to*** **step 3**
    else end

The procedure used to obtain a canonical representation for Manhattan based layouts can easily be extended to non Manhattan designs which use polygons. The key idea of developing the canonical representation sequentially, by extending boundary edges of the representative blocks, can be used for polygons also. The final set will then consist of polygons which have *only one neighbor per edge*. This technique fails in case of layouts utilizing arcs. However, if the arcs are reduced to best-fit polygons, they can again be canonized using the same principle. While finding the polygon representation for an arc the user may use error in area covered as an optimization criterion. The maximum allowable error can be fixed according to the desired accu-

racy of the extractor, at the cost of increase in the running time. The current implementation is based solely on Manhattan designs and some of the recognition criterion might have to be modified when applied to non Manhattan designs.

4.2. Detecting Atomic Elements

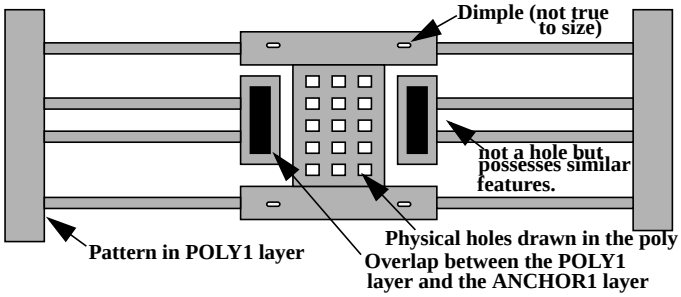
After we have an unique representation for our layout we can now recognize the atomic elements. Our approach is to use feature based recognition techniques. The type of a particular rectangular cell is determined by its shape, size and information of its neighbors. Since it is very difficult to derive all data from this information alone, we make use of some information that is supplied by other layers. This information is technology specific and provides us with hints of locations of some critical elements. This calls for boolean operations of different layers. The main operations performed are the OR and the AND which are very easily achieved using box matching and box overlap techniques. The result of this exercise is a collection of recognized elements. The information embedded in the recognized set can be used to derive a netlist which can then be analyzed using a lumped-parameter simulator.

The first step in the recognition process is to use some information that the other layers already give us. The layer with mnemonic name ANCHOR1 (opens holes for poly1 to nitride or poly0 connection), for example gives us the positions of anchor cells. Similarly the DIMPLE layer gives the position of dimples and HOLE1 layer gives the position of etch holes for plate/mass. Since we know that dimples and holes are placed on mass we use them to find some probable sites for mass. Box overlap checking is done between the rectangles read in from these layers and the cells in the canonical representation of the structural (POLY1) layer. Any cell that overlaps is marked as mass or anchor as the case may be. These pre-recognized cells help us later in detecting other mass and anchor cells.

The next step is to recognize the fingers and the beams. We define a beam to be a long and slender rectangle with neighbors on its opposite edges. Another restriction on beams is that its longer sides should not have any neighbors. We define fingers to be short cantilever beams. Thus they will have only one neighbor that is attached to one of its shorter sides. To take care of fingers that have been split during canonizing we run a proximity test. If an element appears to be a finger, because it possesses some finger properties, and lies in the proximity of a number of other fingers of similar dimension and if these fingers have the same bound (along abscissa if the fingers lie along the ordinate or vice versa) as that of the concerned element, then it is also marked as a finger.

Since many designers tend to put physical holes in their POLY1 layer itself (as opposed to using the HOLE layer), we put in an additional subroutine which detects and eliminates such holes. The key idea in this subroutine is to detect an empty area which is surrounded by cells from all sides. Once such an area has been detected a number of tests are then performed to ascertain whether it is actually a hole. These tests are mainly based on the size, shape and location of the empty space. The type of neighbors surrounding the empty space is also considered. These detected holes also give us cells for mass. Since eliminating these physical holes will in turn increase the mass of the plate which they belong to, we keep a track of total increase in area and, when we calculate the area of the plate, we delete this excess virtual area. Gaps between some beams might also appear as holes, as illustrated in Figure 8. Thus we first use conservative heuristics to recognize holes. This step ensures that empty areas which are not holes are not accidently recognized as holes. After detecting the beams and fingers, we run the hole detection subroutine without conservative heuristics to detect all the remaining holes, if any.

Figure 8: Simple folded flexure element showing holes, dimples and anchor



After having recognized the beams, fingers and some mass and anchor cells, we proceed to recognize the rest of the mass and anchor cells. The key idea is to use the mass and anchor cells that have been recognized and then recursively expand in all directions. To make the subroutine efficient, we do not check any rectangle more than once.

4.3. Improving Computational Efficiency

In the process of canonizing the layout it can be shown that the presence of fingers (as in a comb drive) attached to plate or anchor cells tend to divide the layout unnecessarily. The same happens when there are small holes present in the POLY1 layer (i.e., in the plate). To prevent these unnecessary partitions in the canonization process from adversely affecting the feature-based recognition, we separate out the fingers after they are detected. The removed fingers are stored as a separate group. Next the cells are merged to get a simplified cover of the remaining layout (with the small holes removed). The subroutine to make a canonical representation is run on this modified cover to get a new canonical layout representation. This small modification brings a great improvement in the speed of operation due to the a heavy reduction in the number of cells.

Figure 9: Optimization Algorithm

1. ***initialize*** **A = NULL** and **B = NULL**
   where, **A** and **B** are sets which will contain the final merged structure.
2. ***take*** an element **g** ***from*** the set **G** and ***initialize*** set **P** with it. Where **G** is the set containing the elements that make up the canonical structure and **P** is the working set.
3. ***delete*** the element **g** from set **G**
4. ***for all*** **p** ***in*** **P**
   Q = {x| x = NBR(G,p)
   where **NBR(G, p)** ***returns*** the neighbors of **p** in **G**
5. ***for all*** **q** ***in*** **Q**
   if **TYPE(p, q)** ***then***
   P =P ∪{q} and
   G = G - {q}
   Function **TYPE(p,q)** ***returns*** a value **TRUE** iff **p** and **q** are of the same type.
6. **a = HOR(P)**
   where **HOR(x)** ***returns*** a set which is the horizontal merged version of set **x**.
7. **a = VERT(a)**
   where the **VERT(x)** function ***returns*** the vertical merged version of set **x**.
8. **b = VERT(P);**
9. **b = HOR(b);**
10. **A = A ∪a**
11. **B = B ∪b**
12. ***if*** **G!= NULL** ***go to*** **2**
13. ***if*** **N(A) > N(B)** ***then***
    output **A**
    else
    output **B**
    where N(C) stands for the cardinal number of a set C

4.4. Optimization of Number of Nodes

The canonical representation results in a great number of mass and anchor cells. It breaks the plate and the anchor blocks into many small cells. This is mainly due to the fingers and beams which are connected to them. This would pose a problem for the simulator as the number of nodes unnecessarily increases. We combine the mass cells which are actually a part of a single plate to minimize the number of nodes necessary for the simulation of the extracted MEMS device.

We use an approach similar to that used in corner stitching [25]. In corner stitching the cells are first expanded horizontally, i.e., adjacent cells having the same vertical coordinates are combined, followed by vertical expansion. Thus the representation is maximally horizontal. In our case we still stick to these discrete combining steps (viz., vertical and horizontal expansion) but try and take the best sequence. This is done by comparing the horizontal-then-vertical expansion with the vertical-then-horizontal expansion. The sequence of steps used to achieve this is described in Figure 9.

4.5. Netlist Generation

Now that all the canonized cells have been recognized, and integrated for optimum representation at the schematic level, our next step is to generate the netlist. The information needed for this is already embedded in the recognized set. The main conversion needed is to name the different nodes and express the connectivity between them. The templates for different nodes can be easily done using a counter for each type of atomic elements. Other information like length, width and global position of any node can be easily derived from the coordinates of the rectangles. The positions where the neighboring nodes are connected can be derived from the information about the neighbors. The fine details of the netlist representation will depend on the simulator used. This netlist can then be simulated using a multi domain lumped parameter ordinary differential equation solver like *SABER* [10]. Thus the recognized representation of the layout can be truly used as a reconstructed schematic to verify the original design schematic.

5. RESULTS AND DISCUSSION

We have developed a prototype extraction tool that implements the above algorithms, and are able to successfully identify the atomic elements in many common devices. We now present a select few results to demonstrate these algorithms. To simplify the explanation we have color coded the elements. All mass cells have been marked by a red cross and the anchor cells by a black cross. The fingers have been marked by filled blue rectangles and the beams by filled green rectangles. Finally the joints are shown in brown filled rectangles. The purpose of using crossed rectangles for mass and anchor cells is to show the actual number of rectangles in each type.

5.1. Example 1: Folded Flexure Resonator

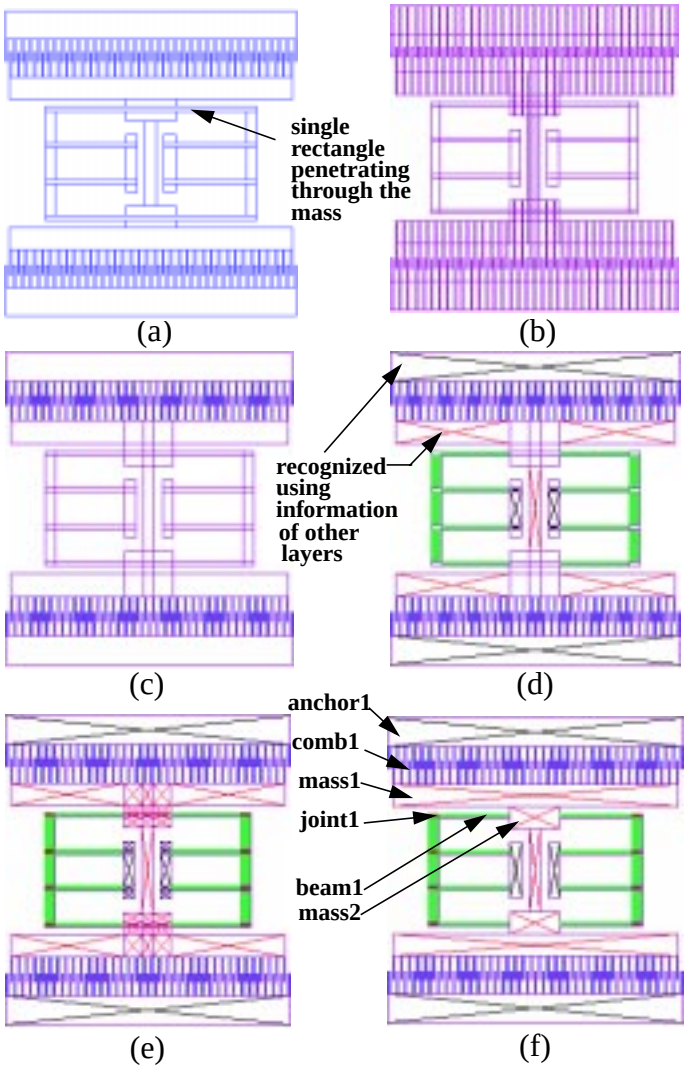
Figure 10 shows the result for a folded flexure resonator. The input is shown in Figure 10(a). It can be clearly seen that in case of the beam marked in the figure, the rectangle representing it penetrates into the mass area. This shows the need for a canonical representation. Figure 10(b) shows the rectangles in the canonical representation. It is important to note that each constituent cell has only one neighbor on each side. This helps us in deriving the neighbor information very easily. As can be easily seen, the number of rectangles or cells in this representa-

tion has increased tremendously. This occurs due to extension of all edges and the situation is made severe due to the presence of fingers. Thus in the next step we separate out the fingers, after having recognized them, and then recanonize the remaining layout. The result is shown in Figure 10(c). This step brings a huge reduction in the number of cells. We then proceed to apply our feature recognition algorithms to recognize beams. Also, interlayer interaction information is used to recognize some of the mass and anchor cells. The result of these feature recognition algorithms is shown in Figure 10(d). This step is followed by the expansion of the mass and anchor cells to detect the remaining cells resulting in the complete recognition of all the cells as shown in Figure 10(e). The next step is to reduce the number of rectangles needed to represent the mass and anchor area. Figure 10(f) shows a minimal representation where the cells have been first merged horizontally and then merged vertically. It can be seen that the number of rectangles needed to represent the central I shaped mass here is three. In case we had opted for vertical merge first followed by horizontal merge, then the number would have been five. Thus the algorithm has taken the correct decision.

The resulting recognized set can now be used to create the netlist. The netlist for the example considered will look similar to the one shown below.

anchor1: xleft: 0; yleft:140; xright: 300; yright: 150; len: 300; wid: 10; neigh1: comb1; neigh_x1:0; neigh_y1: 140; neigh_x2: 300; neigh_y2: 140;
anchor2: xleft: 0; yleft:0; xright: 300; yright: 10; len: 300; wid: 10; neigh1: comb4; neigh_x1:0; neigh_y1: 10; neigh_x2: 300; neigh_y2: 10;
mass1: xleft: 0; yleft:95; xright: 300; yright: 105; len: 300; wid: 10; neigh1: comb2; neigh_x1:0; neigh_y1: 105; neigh_x2: 300; neigh_y2: 105; neigh2: mass2; neigh_x1:135; neigh_y1: 95; neigh_x2: 165; neigh_y2: 95;
....
comb1: xleft 0; yleft: 120; xright: 300; yright: 140; len: 300; wid: 20; overlap: 5; num: 36; neigh1: anchor1; neigh_x1: 0; neigh_y1: 140; neigh_x2: 300; neigh_y2: 140;
....
beam1: xleft: 35; yleft: 87; xright: 135; yright: 89; len: 100; wid: 2; neigh1: mass2; neigh_x1: 135; neigh_y1: 87; neigh_x2: 135; neigh_y2: 89; neigh2: joint1; neigh_x1: 35; neigh_y1: 87; neigh_x2: 35; neigh_y2: 89;
...

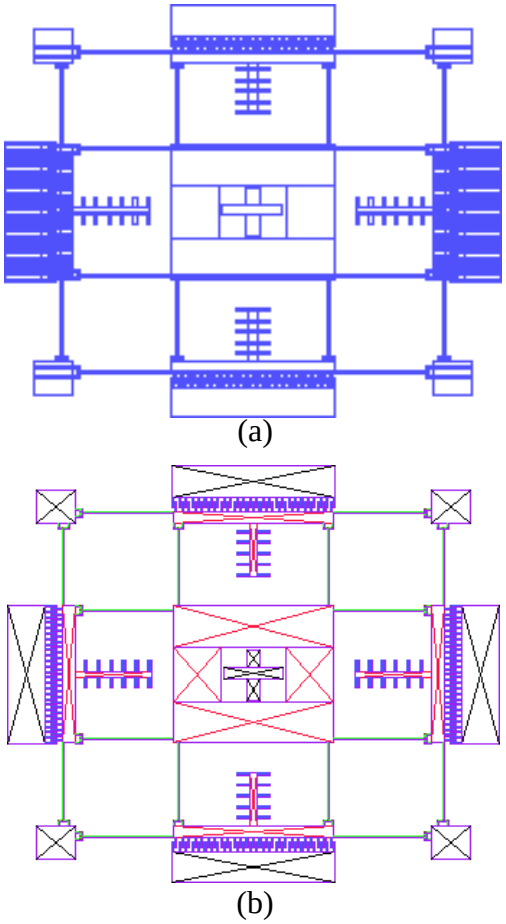
Figure 10: Folded flexure resonator. (a) layout, (b) canonical representation, (c) canonical representation after separating the fingers, (d) intermediate state, (e) detected state, (f) optimized result. (Please refer to text for explanation of color coding.)



5.2. Example 2: Three Fold Symmetric Gyroscope

Figure 11(a) shows the layout for a three fold symmetric gyroscope with the electrostatic drive along the y-axis and a capacitive sensor along the x-axis. The presence of a cross in the center of the mass is to measure the displacement and also to restrict the motion before it buckles the beams. The recognized pattern is shown in Figure 11(b).

Figure 11: Three fold symmetric gyroscope; (a) layout and (b) the reconstructed schematic



5.3. Example 3: Z-direction Accelerometer

Figure 12(a) shows the layout of a z-direction accelerometer and Figure 12(b) shows its corresponding reconstructed schematic. It is important to note here that this design contains no fingers. The capacitive arrangement is actually between the central plate (mass) and a similar ground plate below it. The springs used here are different from the ones (folded flexure) used in a resonator and are actually U-springs.

5.4. Example 4: Pressure Sensor

Figure 13(a) shows a layout of a pressure sensor and Figure 13(b) shows the corresponding reconstructed schematic. Here a crab-leg suspensor been used. It is important to note that the small protrusions on its mass have been detected as a mass and not as a finger. This is in accordance with our heuristic where we had put a condition that the finger's suspended length must be more than the length of the edge which is connected to the mass or anchor.

Figure 12: Z-direction accelerometer; (a) layout and (b) reconstructed schematic

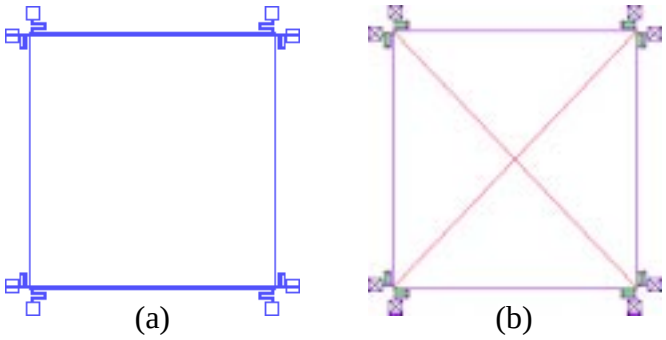
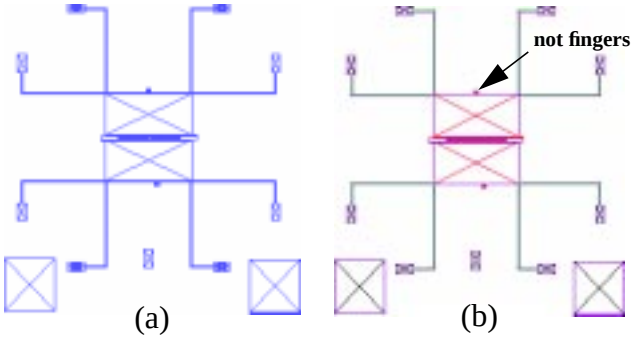


Figure 13: Pressure sensor; (a) layout and (b) reconstructed schematic



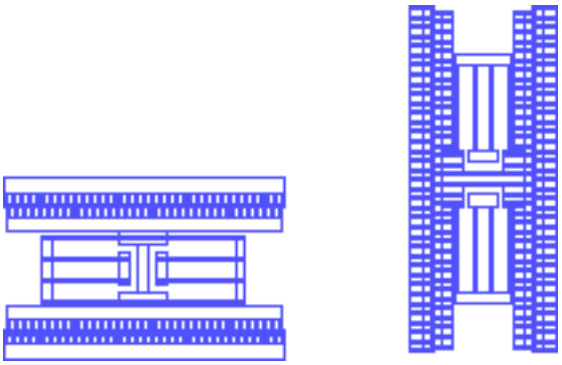
5.5. Example 5: Two Orthogonal Folded Flexure Resonators

Figure 14(a) shows the layout of two folded flexure resonators, like the one used in the first example, placed orthogonal to each other. As can be seen from the reconstructed schematic in Figure 14(b), the merging routine has successfully selected the merging sequence that is best for each resonator. Thus for the left resonator it uses the sequence of horizontal merge followed by vertical merge; while for the resonator to the right it uses the reverse sequence. Thus the net number of rectangles and hence nodes in the netlist is minimized.

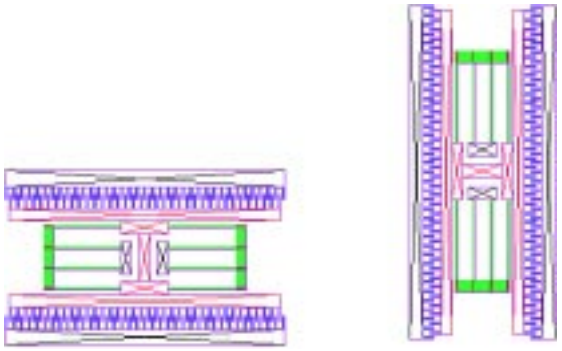
6. CONCLUSION

In order to verify that the detailed design of MEMS device is a correct spatial realization of the desired schematic representation, we need capabilities to reconstruct schematic representations from the spatial representation of the device. Reconstructed schematics can be used to identify design problems without performing expansive physical prototyping.

Figure 14: Two orthogonally placed folded flexure resonators; (a) layout and (b) reconstructed schematic



(a)



(b)

In this paper, we present a feature-based methodology for reconstructing schematics from the layout information. Reconstructed schematics provide information regarding types and parameters of various constitutive elements, and interconnectivity of various elements. Our current implementation is limited to designs based on MUMPS process utilizing Manhattan geometry. It performs quite satisfactorily for a wide variety of devices and can be used extracts springs, comb drives, and masses. We are planning to extend our system to handle designs described by generalized polygons and a wider variety of fabrication processes.

We expect that our tool will help the CAD designers to shrink their design time and also help them in building much more complex MEMS designs.

ACKNOWLEDGEMENT

This research effort is sponsored by the Defence Advanced Research Projects Agency (DARPA) and U. S. Air Force Research Laboratory, under agreement number F30602-97-2-0323. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, the U. S. Air Force Laboratory, or the U. S. Government.

REFERENCES

[1] ADXL50 *Accelerometer Data Sheet*, Analog Devices, Inc., One Technology Way, P.O.Box 9106, Norwood, MA 02062-9106, 1996 (http://www/analog.com).

[2] Introduction to iMEMS^{TM}, Analog Devices, Inc., (http://imems.mcnc.org/imems/imems.html

[3] G.K. Ananthasuresh, S. Kota, and N. Kikuchi, “Strategies for Systematic Synthesis of Compliant MEMS,” DSC-Vol. 55-2, *Dynamics Systems and Control*, Presented at the 1994 ASME Winter Annual Meeting at Chicago, IL, pp. 677-686.

[4] E.K. Antonsson, “Structured Design Methods for MEMS,” *NSF sponsored workshop on Structured Design Methods for MEMS*, November 12-15, 1995.

[5] C.M. Baker and C. Terman, “Tools for Verifying Integrated Circuit Designs,”*Lamda Magazine*, 4th Quarter, 1980, pp 22-30.

[6] J. Bryzek, K. Petersen and W. McCulley, “Micromachines on the march,”*IEEE Spectrum*, IEEE Inc. May 1994, pp. 20-31.

[7] X. Cai, H. Yie, P. Osterberg, J. Gilbert, S. Senturia and J. White, “A Relaxation/Multipole-Accelerated Scheme for Self-Consistent Electromechanical Analysis of Complex 3-D Microelectromechanical Structures,” *Proc. Int. Conf. on Computer-Aided Design*, Santa Clara, California, November 1993, pp. 270-274.

[8] G. K. Fedder, S. Santhanam, M. L. Reed, S. C. Eagle, D. F. Guillou, M. S.-C. Lu, and L. R.Carley, “Laminated High-Aspect-Ratio Microstructures in a Conventional CMO Process,” *Proceedings of the IEEE Micro Electro Mechanical Systems Workshop*, SanDiego, CA, Feb. 11-15 1996, pp.13-18.

[9] G.K. Fedder, S. Iyer and T. Mukherjee, “Automated Optimal Synthesis of Microresonators,” *Proc. 9th Int’l. Conf. on Solid-State Sensors and Actuators (Transducers ‘97)*, Chicago, IL, June 16-19, 1997.

[10] I. Getreu, “Behavioral Modeling of Analog Blocks using the SABER Simulator,”, Proc. Microwave Circuits and Systems, pp 977-980, August 1989.

[11] J.R. Gilbert, G.K. Ananthasuresh, and S.D. Senturia, “3D Modeling of Contact Problems and Hysteresis in Coupled Electro-Mechanics,” *Proc. IEEE MEMS Workshop*, San Diego, CA, Feb. 1996, pp. 127-132.

[12] L.J. Hornbeck, invited paper “Current status of the digital micromirror device (DMD) for projection television applications,” *International Electron Devices Technical Digest*, p.15.1.1 (1993)

[13] R.T. Howe and R.S. Muller, “Resonant Microbridge Vapor Sensor,” *IEEE Trans. Electron Devices*, Vol. ED-33, pp. 499-506, 1986.

[14] R.T. Howe, *et. al.*, “Silicon Micromechanics,” *IEEE Spectrum*, July 1990, pp. 29-35.

[15] T.J. Hubbard, E.K. Antonsson, “Design of MEMS via Efficient Simulation of Fabrication,” *Proceedings of The 1996 Design Engineering Technical Conference and Computers in Engineering Conference*, August 18-22, 1996.

[16] S. Iyer, T. Mukherjee and G.K. Fedder, “Multi-Mode Sensitive Layout Synthesis of Microresonators,” *1998 International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators (MSM 98)*, Santa Clara CA, April 6-8, 1998.

[17] M.-H. Kiang, *et. al*,“Surface-Micromachined Electrostatic-comb Driven Scanning Micromirrors for Barcode Scanners,” *Proceedings IEEE Micro Electro Mechanical Systems Workshop 1996 (MEMS`96)*, Feb. 11-15, 1996, San Diego CA, pp. 192-197.

[18] D.A. Koester, R. Mahadevan, K.W. Markus, *Multi-User MEMS Processes (MUMPs) Introduction and Design Rules*, available from MCNC MEMS Technology Applications Center, 3021 Cornwallis Road, Research Triangle Park, NC 27709, rev. 3, Oct. 1994, 39 pages.

[19] M.S. Kranz and G.K. Fedder, “Micromechanical Vibratory Rate Gyroscopes Fabricated in Conventional CMOS,” *Symposium Gyro Technology*, Stuttgart, Germany, Sept. 1997.

[20] S.P. McCormick, “EXCL: A Circuit Extractor for Integrated Circuit Designs,” *Proceedings of the 21st DAC*, June 1984, pp. 616-23.

[21] M.A. Mignardi, “Digital Micromirror Array for Projection TV,” *Solid State Technology*, v.37, no.7, pp. 63-4, July 1994.

[22] MMAS40G10D *Accelerometer Data Sheet*, Motorola Sensor Products, 1996 (http://design-net.com/senseon).

[23] T. Mukherjee and G.K. Fedder, “Structured Design Of Microelectromechanical Systems,” *Proceedings of the 34th Design Automation Conference (DAC ‘97)*, Anaheim, CA, June 9-13, 1997.

[24] T. Mukherjee and G.K. Fedder, “Design Methodology for Mixed Domain Systems on a chip,” *Proc. IEEE Computer Society Workshop pn VLSI 98*, Orlando FL, April 1998, pp 96-101.

[25] J.K. Ousterhout, “Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools,” *IEEE Transactions on Computer-Aided Design*, Vol. CAD-3, No. 1, January 1984, pp. 87-100.

[26] J.B. Sampsell, “An overview of the digital micromirror device (DMD) and its application to projection displays,” *1993 SID International Symposium Digest of Technical Papers*, 1993, Vol. 24,p.1012.

[27] Intelligent Micro Machine Initiative, Sandia National Laboratories, http://www.mdl.sandia.gov/micromachine/.

[28] J. H. Smith, S. Montague, J. J. Sniegowski, J. R. Murray, and P. J. McWhorter, “Embedded micromechanical devices for the monolithic integration of MEMS with CMOS,” in *Proc. IEDM*, 1995, pp. 609-612.

[29] W. C. Tang, T.-C. H. Nguyen, and R. T. Howe, “Laterally driven polysilicon resonant microstructures,” *Sensors and Actuators*, vol. 20, pp. 25-32, 1989.

[30] W.C. Tang, “Overview of Microelectromechanical Systems and Design Processes,” *34th DAC Proceedings*, 1997, pp. 670-3.

[31] G.M. Tarolli and W.J. Herman, “Hierarchical Circuit Extraction with detailed Parasitic Capacitance,” *ACM IEEE 20th DAC Proceedings*, 1983, pp. 734-38.

[32] S.M. Trimberger, *An Introduction to CAD for VLSI*, Kluwer Academic Publisher, 1987.

[33] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1987.

[34] J.E. Vandameer, M.S. Kranz and G.K. Fedder, “Nodal Simulation of Suspended MEMS with Multiple Degrees of Freedom,” *1997 International Mechanical Engineering Congress and Exposition: The Winter Annual Meeting of ASME in the 8th Symposium on Microelectromechanical Systems*, Dallas, TX, Nov. 16-21, 1997.

[35] J.E. Vandemeer,M.S. Kranz, G.K. Fedder, “Hierarchical Representation and Simulation of Micromachined Inertial Sensors,” *1998 International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators (MSM 98)*, Santa Clara, CA, April 6-8, 1998.

[36] T.J. Wagner, “Hierarchical Layout Verification,” *IEEE Design and Test*, February 1985, pp. 31-37.

[37] H. Yie, S. F. Bart, J. White, and S. D. Senturia, “A Computationally Practical Approach to Simulating Complex Surface-Micromachined Structures with Fabrication Non-idealities." *Proc. IEEE MEMS Workshop*, pp. 128-132, Jan.-Feb. 1995.