An Extraction-Based Verification Methodology for MEMS

Bikram Baidya, Satyandra K. Gupta, and Tamal Mukherjee, *Member, IEEE*

***Abstract*—Micromachining techniques are being increasingly used to develop miniaturized sensor and actuator systems. These system designs tend to be captured as layout, requiring *extraction* of the equivalent microelectromechanical circuit as a necessary step for design verification. This paper presents an extraction methodology to (re-)construct a circuit schematic representation from the layout, enabling the designer to use microelectromechanical circuit simulators to verify the functional behavior of the layout. This methodology uses a *canonical representation* of the given layout on which *feature-based* and *graph-based* recognition algorithms are applied to generate the equivalent extracted schematic. Extraction can be performed to either the *atomic level* or the *functional level* representation of the reconstructed circuit. The choice of level in *hierarchy* is governed by the trade off between simulation time and simulation accuracy of the extracted circuit. The combination of the MEMS layout extraction and lumped-parameter circuit simulation provides MEMS designers with VLSI-like tools enabling faster design cycles, and improved design productivity. [682]**

***Index Terms*—Atomic elements, canonical representation, extraction, functional elements, microelectromechanical systems (MEMS).**

I. Introduction

MICROELECTROMECHANICAL systems (MEMS) integrating multidomain sensors and actuators using conventional batch microfabrication processes are becoming increasingly complex. Manual verification of layouts of such complex systems is virtually impossible. This has led to an increasing need for MEMS layout verification tools. Conventional verification tools using finite element (FEM) analysis or boundary element (BEM) analysis tend to be quite cumbersome and time consuming for such large designs. This paper proposes an alternative approach of using an extractor [1] which reads in the geometric description of the layout and reconstructs the corresponding MEMS circuit. This enables the designer to use the circuit-level lumped parameter simulators [3]–[6] for faster and more convenient layout verification. In addition, minor errors in the layout, like missing connections, can be easily detected in the reconstructed circuit.

Extraction is common in the VLSI world where the main challenge lies in extracting parasitic capacitances and resistances of long interconnect wires [7]–[9]. In the MEMS world we have similar challenges along with added consideration to geometrical features like orientation, relative location, etc., from a mechanical perspective. For example, a L-shaped structure may be recognized as an interconnect with lumped or distributed parasitic capacitance and resistance by a VLSI extractor; but, depending on the context, a MEMS extractor might recognize it as a crab leg spring. Thus, MEMS extraction needs to inherit the principles of VLSI extraction [10], [11] and also expand on them by incorporating geometrical heuristics from the mechanical world. Traditional VLSI extraction tools are designed to recognize only electrical elements like transistors, resistances, capacitances. They do not currently support element recognition for mechanical elements like beams, plates, joints which are essential in MEMS. However, the basic geometry processing functions needed for the extraction of mechanical and electrical elements, like proximity analysis, shape analysis, area calculation, etc., are the same. Access to such geometry processing functions would help us modify existing VLSI extraction tools to include the element recognition modules needed for extracting mechanical elements in addition to the electrical elements. Since none of the existing commercial VLSI extraction tools are transparent enough to allow access to their core geometry processing routines, custom tools are necessary for MEMS extraction. This paper explains the element recognition and geometry processing algorithms written to implement a custom prototype extractor capable of extracting MEMS elements.

In VLSI, designers start by capturing the connectivity and individual element parameters in the form of a schematic. Layout in VLSI designs merely adds parasitic resistances and capacitances due to routing and placement of the circuit elements. These effects are captured in the extracted view of the circuit which is then used to do a Layout versus Schematic (LVS) check [12] for layout verification. Unlike the VLSI world, layout forms the most important representation for MEMS designs [13]. Here the placement of the elements is as crucial as the connectivity and parameters. For example, a symmetrical design in VLSI world would simply mean a symmetrical netlist from power supply to ground, while in MEMS a symmetrical design would additionally mean that the structures (like springs around a plate) are symmetric geometrically. This also increases the importance of having a LVS tool for MEMS. Initial attempts to achieve such capability is limited to higher

Manuscript received April 20, 2001; revised July 30, 2001. This work was supported by the Defence Advanced Research Projects Agency (DARPA) and U.S. Air Force Research Laboratory, under agreement F30602-97-2-0323 and F30602-99-2-0545 and in part by the National Science Foundation Award CCR-9901171. 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. Subject Editor S. D. Senturia.

B. Baidya and T. Mukherjee are with the ECE Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: bbaidya@ece.cmu.edu).

S. K. Gupta is with the University of Maryland, College Park, MD 20742 USA.

Publisher Item Identifier S 1057-7157(02)00075-6.

level connectivity analysis as in [14]. Here, the design entry is at the schematic level and the layout is generated automatically. Each of the elements in the layout are tagged with the corresponding schematic element and the LVS tool verifies the pin to pin connectivity of this tagged layout. Such a methodology does not allow manual layout generation and also fails to capture the placement related errors in the layout. In contrast, the extractor presented here can be used as an initial framework for a MEMS LVS tool having the flexibility of both schematic and layout design entry. The extractor results in a netlist which captures the connectivity and the placement of the actual layout along with the various properties of the recognized elements. Such a netlist can be easily used to perform a LVS check as in VLSI. However improvements need to be made to create a complete LVS tool capable of addressing MEMS specific errors like placement errors, geometrical symmetry errors, etc. As with VLSI, schematic design capture and automated layout generators will eventually replace manual layouts for MEMS. At that time, extraction will still be needed to capture the mechanical parasitics in the final layout. Examples of mechanical parasitics include joints between two beams of different aspect ratios [15].

This paper focuses on single layer *suspended* MEMS components because of their wide applicability. This class of MEMS components cover a wide range of applications, such as pressure sensors [16], micromirrors [17], RF switches [18], accelerometers [19] and resonator filters [20]; and can be fabricated using MEMS fabrication processes like bulk micromachining [21], LIGA [22], surface micromachining [23] or DRIE [21]. As a representative process we will use the Multi-User MEMS Process (MUMPS) [24] which is similar to many other surface micromachining processes like Sandia's SUMMiT process [25], Analog Devices's iMEMS process [26] and Case Western Reserve University's Polycrystalline Silicon Carbide surface micromachining process [27]. An example of a MEMS component built using this process is shown in Fig. 1. It consists of an I-shaped plate suspended by a pair of folded flexure springs. Two sets of electrostatic comb drives on either side act as electromechanical transducers. The process uses two layers of polycrystaline silicon separated by a sacrificial oxide layer. The oxide is etched away in the final step of the process releasing the suspended mechanical structures (defined by the *structural polysilicon mask*) which are connected to the base *conducting polysilicon layer* at the anchors (defined by the *anchor-cut mask*). Additional holes (defined by the *hole mask*) are used on the large areas of the structural layer to aid the release step. These areas also require bushings (defined by the *dimple mask*) to prevent sticking to the bottom surface during the release step. Though the current extraction tool uses the mask conventions of MUMPS for the recognition process, it can be very easily extended to other processes.

A MEMS component can be hierarchically decomposed into *functional elements* like springs and comb drives. Each of these *functional elements* can in turn be decomposed into more fundamental or *atomic elements* like beams, joints, fingers, anchors, plates and gaps. The various *atomic* and *functional elements* for the example in Fig. 1 are marked on the figure. A MEMS component can be represented using schematic symbols and associated behavioral models at each hierarchical level resulting in either an atomic level or a functional level circuit schematic. The functional level schematic simulates faster than the atomic level schematic. The extraction tool gives the user the freedom to choose the hierarchy level depending on the specific needs of accuracy and simulation time.

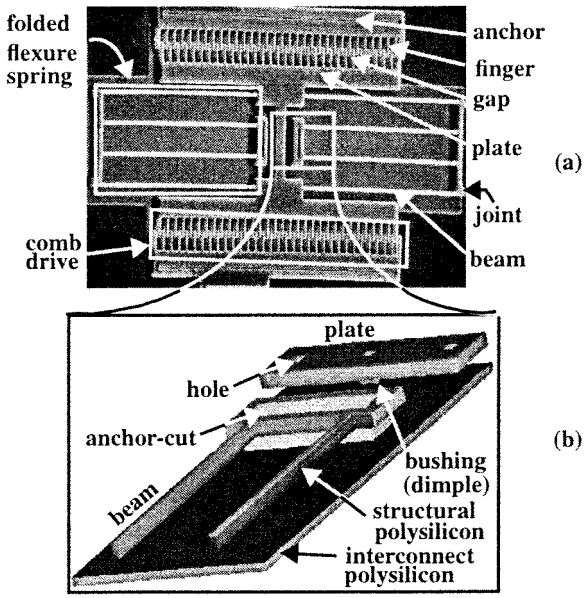


Fig. 1. (a) A scanning electron micrograph (SEM) picture of a folded flexure comb drive micro resonator fabricated in the MUMPS process with atomic elements (listed on the right of the SEM) and functional elements (listed on the left of the SEM), (b) 3-D view of the area highlighted in (a).

The sections that follow give a detailed description of the algorithms and representations used in the prototype extractor. The extraction process is divided into three steps. First, an unique representation for the given layout geometry, referred to as the *canonical representation*, is generated. Section II motivates the need for such a representation and also describes the algorithm to create a canonical representation. The next step is extraction of atomic elements which is described in Section III. The user may generate an atomic level schematic at this stage or proceed toward extraction to the functional level. Section IV describes algorithms used to detect functional elements like springs and comb drives. Section V presents results demonstrating the usefulness of the extractor followed by the conclusion in Section VI.

II. Canonical Representation

Designers tend to design layouts in their own specific way resulting in nonunique representations of the same layout. The extraction tool converts such representations to an unique representation in order to simplify the recognition algorithms used later for extraction. There are various types of unique representations that are followed in VLSI CAD, out of which the tile plane representation [28] is well known. The *canonical representation* is a derivative of the tile plane representation. Unlike the tile plane representation, where each tile can have multiple neighbors on each of its sides, the polygons in canonical representation can either have one or no neighbors on each

edge. Having such an unique neighbor information makes the neighbor based recognition algorithms simpler to implement.

As a vast majority of MEMS layouts are Manhattan, the prototype implementation is limited to Manhattan designs only. Fig. 2 explains canonical representation in the context of Manhattan designs. We define the *canonical representation* of the layout to be the *one 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 component in the layout, i.e., it is the interior area(s) defined by the boundary/boundaries of the geometrical representation of the component in the layout. Thus, in the canonical representation, the Manhattan layout is made up of rectangles such that each rectangle has *at most one neighbor per edge and each edge is either fully covered by a neighbor or not covered at all*. This can be easily achieved by extending the boundary edges into the interior of the layout area till it meets another boundary edge. The resulting representation uniquely partitions the layout area. Extension to non-Manhattan geometries can be accomplished by following the same principle after replacing rectangles by polygons.

The process of canonization starts from the input geometric description of the chip (written, perhaps, in CIF, i.e., Caltech Intermediate Form). The hierarchical layout description of the chip is first flattened followed by the removal of all overlaps between polygons. An initial rectangular cover of the layout is then obtained for the suspended structural pattern in the structural polysilicon mask. This serves as input to the actual canonization routine.

The primary interaction in the canonization process takes place between two sets; the input set and the output set. The output set is always kept in canonical state with respect to its contents and will eventually contain the canonical version of the input set. 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, a series of operations is initiated which ultimately brings the output set back to its equilibrium or canonical state. This is repeated till the input set is empty at which 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 the insertion of a new element, is described below:

```
ADJACENT(rectangle a, rectangle b): returns TRUE if a and b have a common edge
SPLIT(rectangle a, rectangle b): splits a by edges of b, if any of the vertices of the edges of b lies on any edge of a, and also updates the neighbor information
CANONIZE(rectangle_set R)
  G = NULL
  while R! = NULL
    r = pop[R]
    P = NULL
    push[P, r]
    Q = {x|ADJACENT(x, r); x is an element of G}
    G = G - Q
    for i = 0 to length[Q]
      for j = 0 to length[P]
        SPLIT(P[j], Q[i])
    for i = 0 to length[P]
      for j = 0 to length[Q]
        SPLIT(Q[j], P[i])
    M = Q
    until M = NULL
      N = NULL
      for i = 0 to length[M]
        S = {x|x is neighbor of M[i]; x is element of G}
        G = G − S
        for j = 0 to length[S]
          SPLIT(S[j], M[i])
        N = N + S
      P = P + M
      M = N
    G = G + P
  return
```

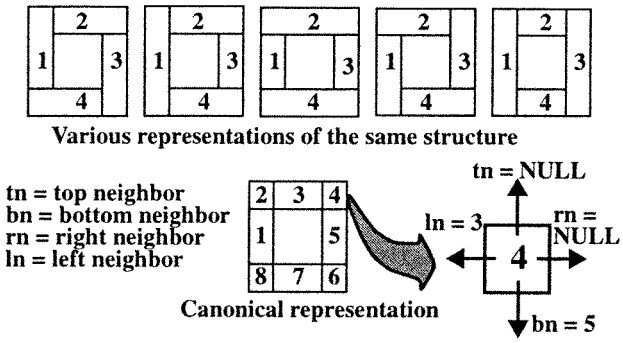


Fig. 2. Fully partitioned canonical representation.

The neighbor information of each rectangle in the resulting canonical representation comes as a by-product of the algorithm. Each rectangle in this representation has four pointers which point to the neighbor, if it exists, on each of the sides (as shown in Fig. 2). The algorithm has a worst case asymptotic upper bound of $O(n^{1.5})$ (See the "Appendix"), where n is the number of rectangles in the final canonized representation. Though this is much greater than the usual complexity of VLSI CAD tools, which normally tend to be $O(nlgn)$, it does not pose a serious time restriction since the problem size is much smaller ($\sim$1000) than that encountered in VLSI world. Nevertheless, the algorithm does have ample scope of improvement in terms of time and storage requirements if the problem size increases.

III. Extraction of Atomic Elements

Atomic elements [as listed on the right side of Fig. 1(a)] form the fundamental building blocks of MEMS components [29]. This section describes some common atomic elements that are used to build suspended MEMS components followed by the description of their recognition heuristics and algorithms.

A. Atomic Elements

Suspended components essentially consists of structural areas suspended over the substrate. Such suspended areas can be partitioned into two groups based on their relative rigidity: *plates* and *beams*. The suspended areas of the structure, which are relatively rigid to forces along the plane of the structure, are defined to be *plates*. They are the major contributors to the mass of the suspended structure. The structural compliance of the suspended structure is decided by the *beams*. Geometrically these are rectangular areas having neighbors only on their shorter sides. Cantilever beams are often classified separately as *fingers* and are extensively used to design electrostatic actuators and sensors. Sometimes fingers are provided with pedestals to reduce the inter-finger gap below the lithography limit for higher sensitivity of the mechanical to electrical transfer function for the electrostatic transducer [30]. Two or more beams are connected by *joints* which can be modeled in the adjacent beams for simplicity. Hence joints serve as logical connectivity elements. The suspended structure is connected to the base (interconnect polysilicon) at the *anchors* which are defined by the anchor-cut mask. These areas provide electrical connection to the suspended structure and also act as mechanical pillars supporting the suspended areas.

B. Extraction Flow for Atomic Elements

At this stage of extraction, a canonical representation of the layout geometry is available and the objective is to tag each rectangle in the canonical representation by the correct element type (anchor, plate, finger, beam or joint). The recognition of each type of elements is a two step process as shown in the flow diagram in Fig. 3. The first step marks probable elements which are confirmed in the second step using stricter rules.

The first step in the recognition process is to mark potential plate and anchor areas using information from the nonstructural masks. For example, overlap of the structural mask with the *anchor-cut mask* is used to tag potential anchors. Similarly overlap with *hole mask* and *bushing mask* are used to tag potential plate rectangles.

The next step is to mark potential fingers from the neighbor information of the rectangles. These are rectangles having a neighbor at one of its shorter sides. The subroutine to confirm fingers selects those rectangles or connected sets of rectangles (fingers split into multiple rectangles during the canonization process), out of the set of potential fingers, which satisfy the criteria that a finger can have only one neighbor and it should lie only on one of its shorter side. If fingers are detected they are removed from the total set of rectangles and the remaining structure is recanonized. This reduces the number of rectangles being checked by the subroutines and also helps in detecting pedestals which come out as fingers in the remaining geometry. The worst case complexity of the finger detection is $O(n+mp)$ where, n is the initial number of rectangles, m number of rectangles tagged in the first loop and p is the remaining rectangles $(n-m)$. Though theoretically this implies quadratic time complexity, in reality it does not create any serious time limitation because the problem size is small ($m \sim 100$).

In the next step, any rectangular empty space surrounded by filled rectangles on each side is tagged as a potential hole. This is achieved by sieving out rectangles in the canonized representation of the geometric *not* of the structural geometry. This is followed by the detection of potential beams which are rectangles having neighbors on each of their shorter sides. This is used by the hole confirmation routine which removes potential holes that are actually gaps between beams. The holes detected are then replaced by filled rectangles followed by a recanonization of the resulting geometry. The change in physical parameters (like mass-factor, center of mass, etc.) due to the addition of these filled rectangles is annotated to the corresponding rectangles. In addition to reducing the number of rectangles in the representation of the layout, the recanonization also reduces the chances of split beams.

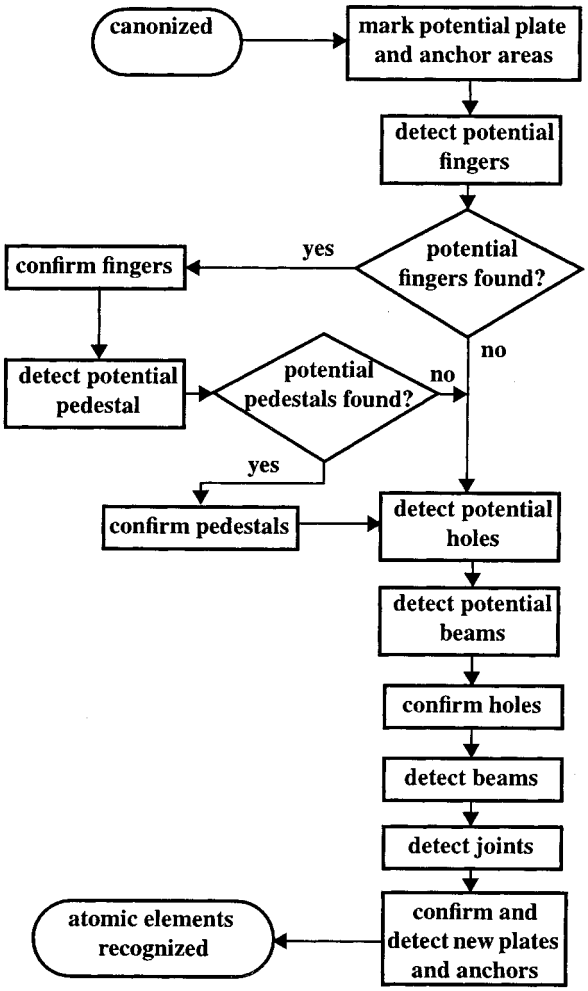


Fig. 3. Flow chart for extraction of atomic elements.

The next step detects beams and joints using neighbor-based heuristics followed by the final detection of plates and anchors. Previously marked potential plate and anchors are used as seeds to recursively expand into unrecognized areas and mark them as plates or anchors. The expansion process checks each rectangle only once and hence runs in linear time. The final recognized set of rectangles can be used to generate an atomic level schematic or can be used to proceed to functional level extraction.

IV. Extraction of Functional Elements

Complex MEMS components are best modeled in the functional level schematic. Such schematics have fewer elements

than atomic level schematics and hence allow faster simulations without any significant change in accuracy of result [31]. This motivates the extraction to the functional level instead of stopping at the atomic level. This section describes the various types of functional elements [as listed on left side of Fig. 1(a)] and the algorithms that are used for extracting them.

A. Plate and Anchor

The canonization process results in a large number of interconnected plates and anchors in the atomic level schematic. Combining such interconnected rigid plates and anchors to get a minimal representation would result in a significant decrease in the number of elements in the schematic. The plates and anchors in such a minimal representation are referred to as functional plates and anchors. The algorithm described below is used to achieve a maximal horizontal or maximal vertical representation of interconnected sets of similar atomic elements depending on which representation results in smaller number of elements.

```
HORIZONTAL(rectangle_set A): returns the
  maximal horizontal representation of A
VERTICAL(rectangle_set A): returns the
  maximal vertical representation of A
PATH(rectangle a, rectangle b): returns
  TRUE if a path exists from a to b such
  that every element of the path including
  a and b are of the same type
OPTIMIZE_ELEM(rectangle_set G)
  H = NULL
  V = NULL
  while G! = NULL
    g = pop[G]
    P = NULL
    push[P, g]
    Q = {x|PATH(g, x); x is an element of G}
    P = P + Q
    G = G − Q
    A' = HORIZONTAL(P)
    A = VERTICAL(A)
    B' = VERTICAL(P)
    B = HORZONTAL(B')
    H = H + A
    V = V + B
if length[H] < length[V] then return(H)
else return(V)
```

Since the neighbor information for each rectangle is already available from the canonization algorithm, the sets Q can be easily obtained by visiting each rectangle only once. The merging algorithms also make use of the neighbor information and hence check each rectangle in the set only once. Hence the entire algorithm runs in linear time.

B. Electromechanical Comb Actuators

Silicon microstructures have long been actuated and sensed electrostatically. A widely used electrostatic actuator is the linear comb drive [32] made up of interdigited fingers which may or may not have pedestals [30] [see Fig. 4(a) and (b)]. One side of the comb is fixed (stator) while the other side is allowed to move (rotor). Another popular electrostatic actuator is the differential comb drive [33] [see Fig. 4(c)] using three sets of electrically isolated comb fingers arranged such that the rotor set of comb finger sees different sets of capacitances on its two sides. Such a structure is used to sense transverse motion via differential sensing of the two sets of capacitances and has an added advantage of reduced levitation problems [34].

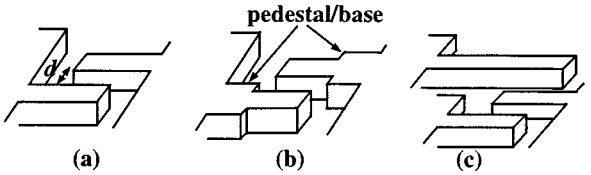


Fig. 4. Different types of finger arrangement (a) pair of cantilever beams forming the building block of electrostatic comb drive, (b) fingers with pedestals, (c) differential comb finger arrangement.

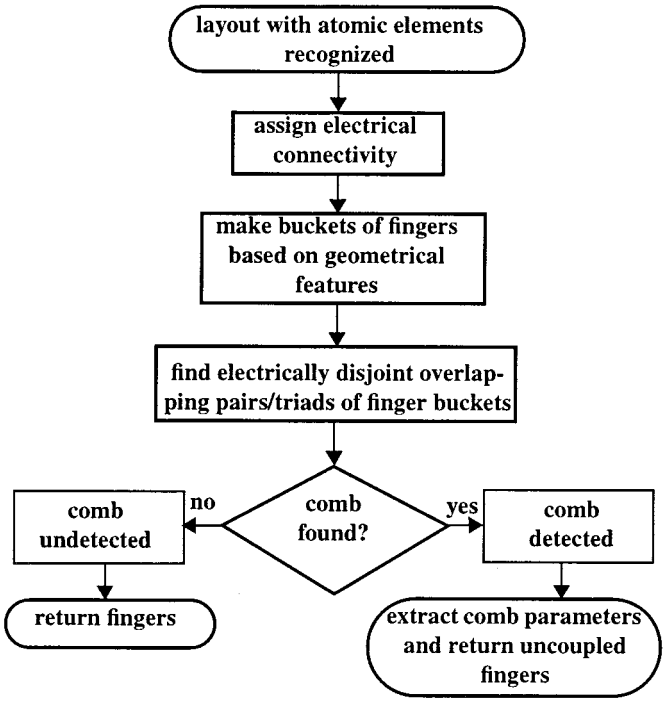


Fig. 5. Comb drive extraction flow chart.

The comb drive extraction flow is shown in Fig. 5. It starts with a connectivity analysis of the set of recognized fingers. Electrically connected fingers are given the same connectivity number and then sorted into buckets based on their orientation and connectivity. Each such finger bucket is then checked for uniformity of the fingers with respect to their geometrical parameters like, region of occurrence, length of fingers, width of fingers and interfinger gap. If the fingers have pedestals, then the geometrical parameters of the pedestal (like region of occurrence, length and width of the pedestal, interpedestal gap and the relative position of the pedestal with respect to the thin cantilever finger) are also checked. The buckets are partitioned whenever any nonuniformity is found in any of these parameters. A set of overlapping pair or triad of such buckets with different electrical connectivity numbers will result in a linear

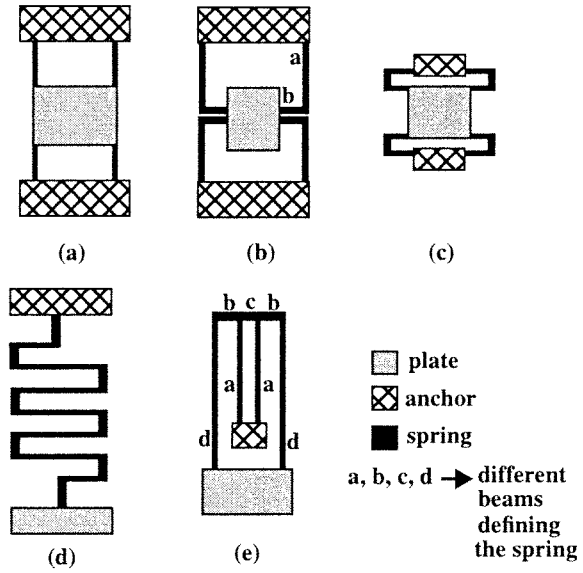


Fig. 6. Springs in the spring library: (a) fixed–fixed, (b) crab leg, (c) U-spring, (d) meander spring, and (e) folded flexure.

or differential comb drive respectively. The box covers of the buckets are checked using box overlap rules to find such pairs or triads. Any such set is then checked to avoid uncoupled fingers and finally grouped together to form a linear or differential comb drive.

C. Mechanical Springs

Springs are composed of beams and joints and connect the suspended plate to the anchors. Few commonly used springs are shown in Fig. 6. The fixed–fixed flexure [see Fig. 6(a)] consists of a simple straight beam connecting the suspended plate to the anchor and has a high spring constant because of extensional axial stress in the beams. Crab-leg springs and U-springs [see Fig. 6(b) and (c)] are modifications of the fixed–fixed beam to reduce peak stress in the flexure at the cost of reduced stiffness in undesired directions. A meander spring [see Fig. 6(d)] is also a modified version of fixed–fixed flexure which helps achieve more compliance using less space. A folded flexure [see Fig. 6(e)] design reduces axial stress and also has the advantages of providing more compliance while occupying less area. The springs types (like crab leg, serpentine spring, folded flexure, etc.) to be recognized are stored in the form of graphs in a library file. The spring detection routine reads in this library file to create an internal copy of the graphs. Any spring consisting of a contiguous set of beams and joints that can be represented in the manner described below can be recognized by the spring detection routine. However, the methodology described can be generalized to handle springs made of other atomic elements by incorporating user definable atomic elements.

The spring detection routine stores the spring library by creating a finite state machine (FSM) for each of the springs defined in the library. Each of the FSMs can be defined by $\mathrm{M} = \{\mathrm{Q}, \mathrm{S}, \mathrm{L}, \mathrm{G}, \mathrm{F}, \mathrm{X}\}$, where

$\mathrm{Q}:$ states = {S, {intermediate states}, F, X};
$\mathrm{S}:$ start state = anchor point;
$\mathrm{L}:$ inputs = {{joints}, {beams}, NULL};
$\mathrm{G}:$ transition rules;

TABLE I
DICTIONARY OF JOINTS

Joint name	m - param	t- param	ports	example
J_+	+1	0	2	
J_-	-1	0	2	
J_{T0}	0	0	3	
J_{T+}	+1	+1	3	
J_{T-}	-1	+1	3	
J_0	0	+1	4	

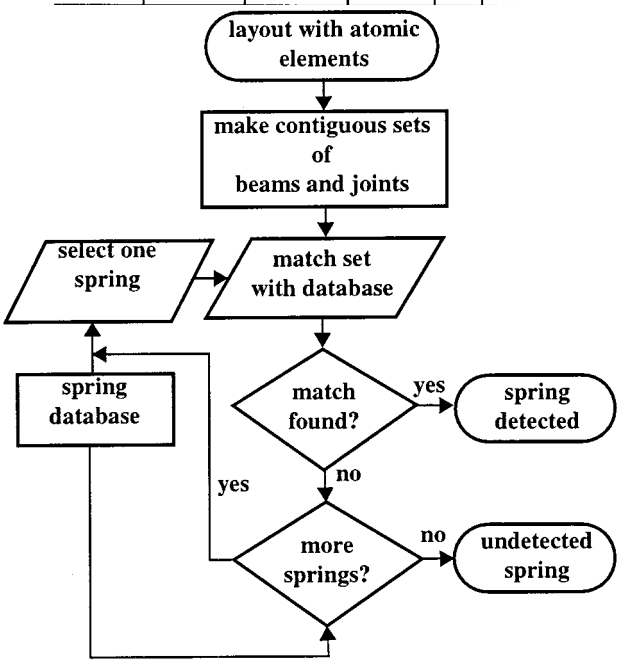


Fig. 7. Spring extraction flow chart.

$\mathrm{F}:$ set of final states; and
$\mathrm{X}:$ exit state.

A joint, in such a language, is defined to be a node having one input port and at most three output ports and is labeled using the "m" (from moment) and "t" (from transition) parameters. The t-parameter is 1 only if there is an output port along the direction of the input port. An output port at right angles to the input port contributes a $+1$ or -1 to m-parameter depending on the direction of (anticlockwise or clockwise respectively). The six types of joints possible using such a convention are shown in Table I. The set of beams for the language depend on the spring to be detected. For example, a crab leg spring requires two beams [see Fig. 6(b)] which may or may not be equal in dimension, while a folded flexure requires four type of beams which must be arranged as shown in Fig. 6(d).

Connected sets of beams and joints obtained after the atomic recognition are passed through each of these FSM's to recognize their type. For each such set, the input is started from a beam which is connected to an anchor rectangle. The flow of the spring detection algorithm is shown in Fig. 7. Any undetected spring is replaced by its atomic level representation, i.e., in the form of beams and joints.

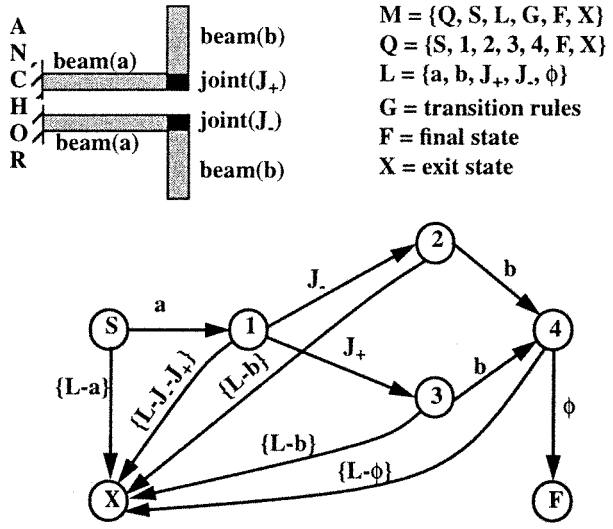


Fig. 8. FSM to recognize a crab leg.

Fig. 8 shows the FSM that defines a crab leg. For a connected set of beams and joints, the algorithm first detects the beam that is connected to an anchor rectangle and initializes the *current state* to be the *start state* (S in Fig. 8) of the FSM. At every state the algorithm uses the next element in the connected set of beams and joints as the input to decide the next state. For example, for the *start state*, if the next input is a beam (which, being the first instance of beam, will be registered as *beam a*) then the *current state* advances to *state 1*. If the input is not a beam (i.e., if it is a joint or if there are no other elements in the set) then the *current state* advances to *state x*. Reaching *state x* implies that the given set is not a crab leg and the algorithm exits from the current spring type (crab leg). It then tries to match the set with some other spring type in the library. If there are no more spring types remaining in the library, the current set is marked as an unrecognized spring. For a set that forms a crab leg, the first input will be a beam (*beam a*) followed by either a J_+ or J_- joint and finally another beam (which will be marked as *beam b*). At this point the *current state* will reach *state 4*. For a crab leg, there should be no more elements (beams or joints) left as input. If this is the case (i.e., the input is NULL, symbolized by in Fig. 8), then the current state reaches *final state* (F) and the set of beams and joints is recognized as a crab leg.

V. Results

This section presents some results to demonstrate the capability of the prototype extractor which implements the algorithms discussed in the previous sections.

A. Accelerometer

Fig. 9 shows the results for an accelerometer design [33] using U spring as the flexure and a pair of differential comb drives for sensing the output. Any inertial force acting on the central plate due to an external acceleration causes the suspended structure to move in the horizontal direction resulting in a differential change of the capacitances in the comb drive which can then be sensed using sensing electronics. Fig. 9(a) shows the initial layout. Notice the structural holes that are present in the plate area of the layout. Fig. 9(b) shows the fully partitioned canonical representation of the component. The information from the anchor-cut mask is used to mark the anchor areas of the layout [see Fig. 9(c)]. This is followed by the recognition of atomic elements [see Fig. 9(d)]. The structural holes in the plate have been replaced by actual plate areas. The resulting decrease in mass is annotated in the schematic resulting from this representation. The next step is to recognize the functional elements, i.e., U spring and differential comb drive. Fig. 9(e) shows the final recognized state of the component. The corresponding functional level schematic is shown in Fig. 9(f). The transient response of the accelerometer due to a pulse acceleration input of 1g is shown in Fig. 9(g).

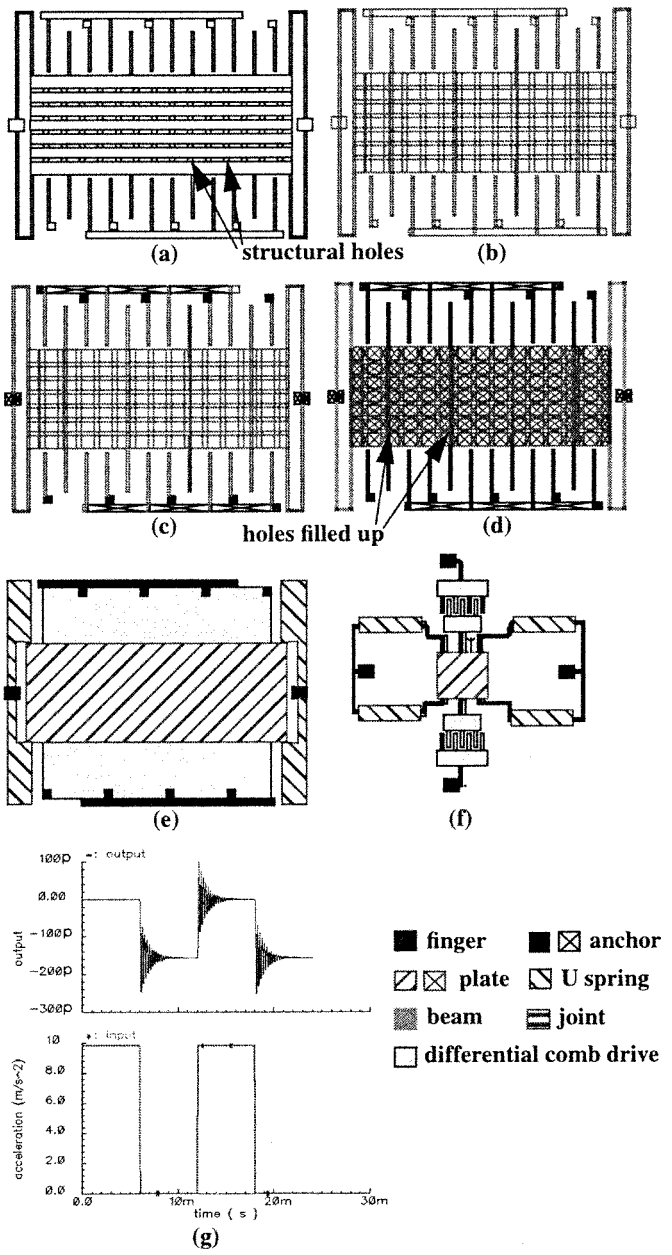


Fig. 9. Accelerometer: (a) input layout, (b) canonical representation, (c) intermediate state, (d) atomic elements recognized, (e) functional elements recognized, (f) reconstructed functional level schematic, and (g) transient response to a 1-g pulse acceleration input.

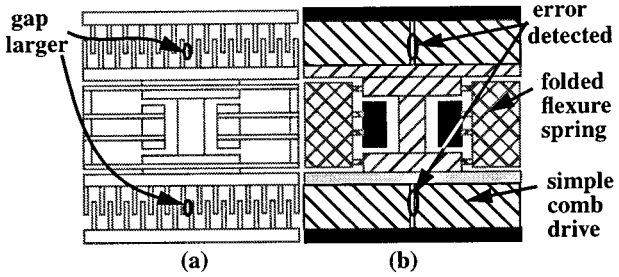


Fig. 10. Erroneous layout of a resonator: (a) input layout and (b) recognized state.

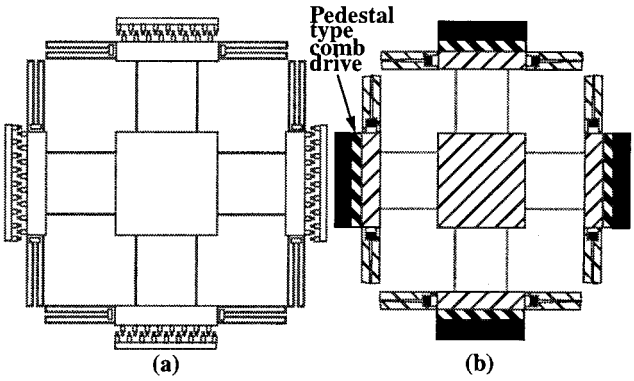


Fig. 11. Three-fold symmetric gyroscope: (a) input layout and (b) recognized state.

The simulation using functional level schematic was found to be approximately 10 times faster, than the same simulation using atomic level schematic, with no appreciable change in accuracy of the result.

B. Erroneous Resonator Layout

The importance of the extractor is demonstrated in this example (see Fig. 10) where the input layout of a folded-flexure resonator [35] was found to have a very small error which was not detected by the human eye. When extracted, the recognized representation contained two sets of comb actuators instead of just one pair. On inspecting the original layout, it was found that there was a difference in the gaps between the two halves of each comb drive. This was because when the half was being replicated and placed to double the size of the comb actuator, a small human error resulted in a gap which was more than the gaps between other fingers. This was detected by the extractor and was interpreted as two sets of comb actuators. The layout was then corrected to remove the error. This example demonstrates how the extractor can be combined with visual inspection to act as a LVS tool for MEMS designs.

C. Gyroscope

The issues like symmetry and cross axis coupling become extremely important in complex devices like the three-fold symmetric gyroscope [36] shown in Fig. 11(a). The gyroscope shown uses U-springs and beams for its suspension mechanism and pedestal type fingers in its comb drive for increased actuation. The extractor is able to correctly recognize the various mechanical and electromechanical parts of the device resulting in an extracted schematic representation [see Fig. 11(b)] which

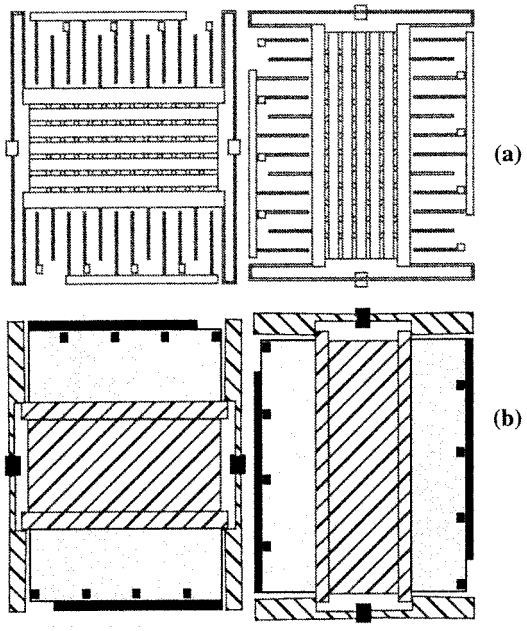


Fig. 12. Orthogonal accelerometers: (a) input layout and (b) recognized state.

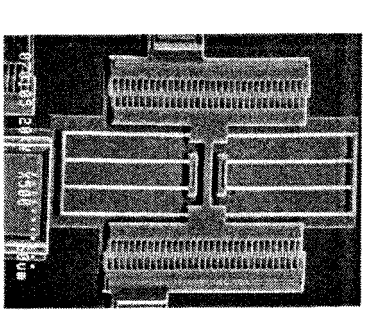
can be simulated to verify the functional behavior of the device. This example demonstrates the capability of the extractor to act as a verification tool for complex MEMS devices.

D. Orthogonal Accelerometers

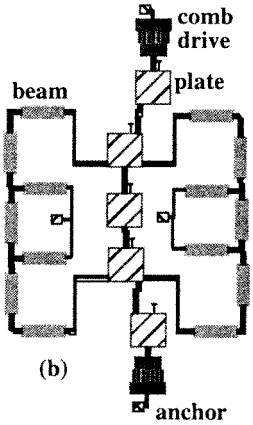
Fig. 12(a) shows a layout consisting of two orthogonal accelerometers. Such a combination is sometimes used to sense acceleration along the orthogonal axes [19]. Fig. 12(b) shows the recognized representation. As can be seen, the optimization algorithm for generating minimal number of plates and anchors in the functional level representation selected the correct representation for each. Thus for the accelerometer on the left it selected the maximal horizontal representation while for the accelerometer on the right it selected a maximal vertical representation.

E. Experimental Verification

An array of folded flexure resonators [35] were fabricated in MUMPS [24] and the experimental results [37] were compared with the results from simulation of extracted schematic (using NODAS [6]). Fig. 13(a) shows an SEM picture of a 30 kHz resonator fabricated in the array. The corresponding extracted schematic is shown in Fig. 13(b). The extracted schematic included a mean overetch value of 0.135 μm which was obtained from actual measurements of the fabricated structures. For this particular example the spring recognition was disabled, resulting in a schematic which contained atomic as well as functional elements. This was done to capture the effect of overetch in the beams which was not available in the functional level model of the folded flexure spring [31]. Accuracy of simulation result was considered more important than the gain in simulation time that could have been achieved if a schematic with only functional elements was used. The table in Fig. 13(c) compares the resonant frequency and Q-factor obtained from simulation and experimental measurements. The deviation in



(a)



(b)

parameter	measured	extracted	extracted over measured
resonant frequency	22.7 kHz	22.9 kHz	0.9%
quality factor	25.6	27.1	5.9%

(c)

Fig. 13. A 30-kHz folded flexure resonator: (a) SEM picture, (b) extracted schematic, and (c) comparison of results.

Q-factor is due to the inaccuracies in the damping models used in simulation.

VI. Conclusion

Verification of complex MEMS designs need capabilities to reconstruct circuit schematic representation from the layout representation of the component. Such reconstructed schematics can be used to identify design problems without performing time consuming numerical simulation. An extractor based on geometrical heuristics has been proposed and its capabilities demonstrated. The extractor also gives the user the flexibility to extract to two different levels of design hierarchy depending on the user's requirements of simulation time and accuracy. The prototype implementation of the extractor is limited to Manhattan designs using the MUMPS process, but can be extended to handle non Manhattan designs and other processes.

Appendix

Proposition 1: The maximum size of the number of rectangles (n) in the canonical representation of a layout is $O(r^2)$ where r is the number of nontrivial rectangles used to represent the layout.

Proof: Let r be the initial number of nontrivial rectangles. Since the rectangles are nontrivial, there are no overlapping rectangles and also no two neighboring rectangles have totally overlapping edges. Then, number of vertical outer edges of the layout (v) is $O(r)$ and similarly, number of horizontal outer edges of the layout (h) is also $O(r)$. Let n_v be the points on the abscissa obtained by projecting the vertical outer edges on the abscissa. Since some of the vertical outer edges will be colinear, $n_v \leq v$. Hence, n_v is $O(r)$. Similarly, number of points on the ordinate axis obtained by projecting the edges in h on the ordinate (n_h) will also be $O(r)$. If a grid is formed using these points (n_v and n_h) then total number of grid rectangles (n_g) will be $O(n_v * n_h)$ which is $O(r^2)$. Since canonical representation is obtained by extending the outer edges of the polygon, the number of rectangles in the canonical representation (n) will be less than or equal to n_g. Hence n has an upper bound of $O(r^2)$.

Proposition 2: The running time for the canonization algorithm is $O(n^{1.5})$ where n is the number of rectangles in the final canonical representation.

Proof: Total time in canonization is the sum of the time required to create the rectangles of the canonical representation and the time required to find the ADJACENT set (Q) in each loop. The time required to create n rectangles is $O(n)$. To find out the time required to find the ADJACENT set let us consider the ith rectangle of the starting set R. The number of rectangles already in the final set G at this stage will be of $O((i-1)^2)$ [see Proposition 1]. Hence, number of comparisons needed to get Q for the ith rectangle will be $O((i-1)^2)$. Summing up for all i from 1 to r (where r is the number of rectangles in the starting set R), we get, total time in comparisons to be $0(\Sigma(i-1)^2)$ which is $O(r^3)$ or $O(n^{1.5})$ [since $n = O(r^2)$ from Proposition 1]. Hence total time in canonization is $O(n + n^{1.5})$ which is $O(n^{1.5})$.

References

[1] B. Baidya, S. K. Gupta, and T. Mukherjee, "Feature-recognition for MEMS extraction," in *Proc. CDROM 1998 ASME Design Engineering Technical Conf.*, Atlanta, GA, Sept. 13–16, 1998.

[2] B. Baidya, S. K. Gupta, and T. Mukherjee, "MEMS component extraction," in *Proc. Modeling and Simulation of Microsystems '99*, San Juan, Puerto Rico, Apr. 19–21, 1999, pp. 143–146.

[3] G. Lorenz and R. Neul, "Network-type modeling of micromachined sensor systems," in *Proc. Modeling and Simulation of Microsystems '98*, Santa Clara, CA, Apr. 6–8, 1998, pp. 233–238.

[4] J. V. Clark, N. Zhou, and K. S. J. Pister, "Modified nodal analysis for MEMS with multi-energy domains," in *Proc. Modeling and Simulation of Microsystems 2000*, San Diego, CA, Mar. 27–29, 2000, pp. 723–726.

[5] J. E. Vandameer, M. S. Kranz, and G. K. Fedder, "Nodal simulation of suspended MEMS with multiple degrees of freedom," in *Proc. 1997 Int. Mechanical Engineering Congress and Exposition: The Winter Annu. Meeting of ASME in the 8th Symp. on Microelectromechanical Systems*, Dallas, TX, Nov. 16–21, 1997.

[6] G. K. Fedder and Q. Jing, "A hierarchical circuit-level design methodology for microelectromechanical systems," *IEEE Trans. Circuits Syst. II* , vol. 46, pp. 1309–1315, Oct. 1999.

[7] C. M. Baker and C. Terman, "Tools for verifying integrated circuit designs," *Lamda Mag.*, pp. 22–30, 4th Quarter 1980.

[8] S. P. McCormick, "EXCL: A circuit extractor for integrated circuit designs," *Proc. 21st DAC*, pp. 616–623, June 1984.

[9] G. M. Tarolli and W. J. Herman, "Hierarchical circuit extraction with detailed parasitic capacitance," in *Proc. ACM IEEE 20th DAC*, 1983, pp. 734–738.

[10] S. M. Trimberger, *An Introduction to CAD for VLSI*. New York: Kluwer Academic, 1987.

[11] J. D. Ullman, *Computational Aspects of VLSI*. Rockville, MD: Computer Science, 1987.

[12] C. Ebeling, "GeminiII: A second generation layout validation program," in *Proc. IEEE Int. Conf. Computer-Aided Design, ICCAD-88. Digest of Technical Papers*, Santa Clara, CA, Nov. 7–10, 1988, pp. 322–325.

[13] N. R. Swart, "A design flow for micromachined electromechanical systems," *IEEE Design Test Comput.*, vol. 16, no. 19, pp. 39–47.

[14] S. F. Bart, N. R. Swart, M. Mariappan, M. H. Zaman, and J. R. Gilbert, "An environment for MEMS design and verification," in *Proc. Modeling and Simulation of Microsystems 1998*, Santa Clara, CA, Apr. 6–8, 1998, pp. 386–391.

[15] B. Baidy and T. Mukherjee, "Layout extraction for integrated electronics and MEMS devices," in *Proc. Transducers '01*, Munich, Germany, June 10–14, 2001, pp. 280–283.

[16] A. V. Chavan and K. D. Wise, "A monolithic fully-integrated vacuum-sealed CMOS pressure sensor," in *Proc. IEEE Aerospace Applications Conf.*, Mar. 21–28, 1998, pp. 341–346.
[17] P. F. Van Kessel, L. J. Hornbeck, R. E. Meier, and M. R. Douglass, "A MEMS-based projection display," in *Proc. IEEE Integrated Sensors, Microactuators, and Microsystems (MEMS)*, vol. 86, Aug. 1998, pp. 1687–1704.
[18] E. R. Brown, "RF-MEMS switches for reconfigurable integrated circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 46, no. 11, pp. 1868–1880, Nov. 1998.
[19] H. Samuels, "Single- and dual-axis micromachined accelerometers," *Analogue Dialogue*, vol. 30, no. 4, pp. 3–5, 1996.
[20] K. Wang and C. T.-C. Nguyen, "High-order micromechanical electronic filters," in *Proc. IEEE MEMS Workshop*, Japan, Jan. 26–30, 1997, pp. 25–30.
[21] G. T. A. Kovacs, N. I. Maluf, and K. E. Petersen, "Bulk micromachining of silicon," in *Proc. IEEE Integrated Sensors, Microactuators, and Microsystems (MEMS)*, vol. 86, Aug. 1998, pp. 1536–1551.
[22] H. Guckel, "High-aspect-ratio micromachining via deep x-ray lithography," in *Proc. IEEE Integrated Sensors, Microactuators, and Microsystems (MEMS)*, vol. 86, Aug. 1998, pp. 1586–1593.
[23] J. M. Bustillo, R. T. Howe, and R. S. Muller, "Surface micromachining for microelectromechanical systems," in *Proc. IEEE Integrated Sensors, Microactuators, and Microsystems (MEMS)*, vol. 86, Aug. 1998, pp. 1552–1574.
[24] Cronos web page [Online]. Available: http://www.memsrus.com
[25] [Online]. Available: http://www.mdl.sandia.gov/micromachine/trilevel.html
[26] [Online]. Available: http://www.analog.com/industry/iMEMS/
[27] A. A. Yasseen, C. A. Zorman, and M. Mehregany, "Surface micromachining of polycrystalline SiC films using microfabricated molds of SiO_2 and polysilicon," *J. Microelectromech. Syst.*, vol. 8, no. 3, pp. 237–242.
[28] J. K. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 1, pp. 87–100, Jan. 1984.
[29] T. Mukherjee, G. K. Fedder, and R. D. (Shawn) Blanton, "Hierarchical design and test of integrated microsystems," *IEEE Design Test Comput.*, vol. 16, no. 4, pp. 18–27, Oct.-Dec. 1999.
[30] T. Hirano, T. Furuhatha, K. T. Gabriel, and H. Fujita, "Design, fabrication and operation of submicron gap comb-drive microactuator," *J. Microelectromech. Syst.*, vol. 1, pp. 52–59, Mar. 1992.
[31] S. V. Iyer and T. Mukherjee, "Numerical spring models for behavioral simulation of MEMS inertial sensors," in *Proc. Design, Test, Integration and Packaging of MEMS/MOEMS*, Paris, France, May 9–11, 2000, pp. 55–62.
[32] W. C. Tang, T.-C. H. Nguyen, M. W. Judy, and R. T. Howe, "Electrostatic-comb drive of lateral polysilicon resonators," in *Proc. Transducers '89*, vol. 2, June 1990, pp. 328–331.
[33] M. Lemkin and B. E. Boser, "A micromachined fully differential lateral accelerometer," in *Proc. IEEE Custom Integrated Circuits Conference*, 1996, pp. 315–318.
[34] W. C. Tang, M. G. Lim, and R. T. Howe, "Electrostatically balanced comb drive for controlled levitation," in *Proc. Technical Digest IEEE Solid-State Sensor and Actuator Workshop*, June 1990, pp. 23–27.
[35] W. C. Tang, T.-C. H. Nguyen, and R. T. Howe, "Laterally driven polysilicon resonant microstructures," *Sens. Actuators*, vol. 20, pp. 25–32, 1989.
[36] J. E. Vandemeer, M. S. Kranz, and G. K. Fedder, "Hierarchical representation and simulation of micromachined inertial sensors," in *Proc. Modeling and Simulation of Microsystems '98*, Santa Clara, CA, Apr. 6–8, 1998, pp. 540–545.
[37] S. Iyer, "Layout synthesis of microresonators," M.S. thesis, Carnegie Mellon University, Pittsburgh, PA, 1998.

Bikram Baidya received the B.Tech. degree in electrical and electrical communication engineering from Indian Institute of Technology, Kharagpur, India, in 1997 and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1999. He is currently pursuing the Ph.D. degree in the MEMS laboratory at Carnegie Mellon University.

His research interests include CAD for MEMS, modeling of fluidic systems and extraction of integrated MEMS.

Satyandra K. Gupta received the Ph.D. degree in mechanical engineering from the University of Maryland at College Park in 1994.

In fall 1998, he joined University of Maryland, College Park, as an Assistant Professor in Mechanical Engineering and the Institute for Systems Research. Prior to joining the University of Maryland, he was a Research Scientist in the Robotics Institute and an Adjunct Assistant Professor of Manufacturing in the Graduate School of Industrial Administration at Carnegie Mellon University, Pittsburgh, PA. His research is focused on developing new algorithms for creating next-generation computer-aided design and manufacturing (CAD/CAM) systems that can reduce time-to-market, enable cost-effective small batch manufacturing, and facilitate manufacturing of geometrically complex heterogeneous objects. Over the last 10 years, he has participated in a number of research projects in the area of computer-aided design and manufacturing. Representative projects include generative process planning for machining, automated manufacturability analysis, automated generation of redesign suggestions, generative process planning for sheet metal bending, automated tool design for sheet metal bending, assembly planning and simulation, extraction of lumped parameter simulation models for microelectromechanical systems, distributed design and manufacturing for solid freeform fabrication, integration of market research with design option evaluation, and automated design of multistage, multipiece molds. He has authored or coauthored more than 75 articles in journals, conference proceedings, and book chapters. He has organized several conference sessions in the area of computer-aided design and manufacturing.

Dr. Gupta is a member of American Society of Mechanical Engineers (ASME) and Society of Manufacturing Engineers (SME). He has served as Program Co-Chair in 1998 ASME's Design for Manufacturing Conference, Papers Chair in 1999 ASME's Design for Manufacturing Conference, and Exhibit Chair in 2000 ASME's Design Engineering Technical Conferences. He has won many honors and awards for his academic excellence and his research contribution to computer-aided design and manufacturing area. In particular, he has been awarded a Graduate School Fellowship and an Institute for Systems Research Graduate Fellowship during his Ph.D. study. Other awards include Gold Medal for first rank in B.E. (1988), Gold Medal for the Best Engineering Design Project (1988), Institute for Systems Research Outstanding Systems Engineering Graduate Student award (1993–1994), Best Paper Award in ASME's International Conference on Computers in Engineering in 1994, Best Paper Award in ASME's Design for Manufacturing Conference in 1999, ONR's Young Investigator Award in 2000, SME's Robert W. Galvin Outstanding Young Manufacturing Engineer Award in 2001, NSF's CAREER Award in 2001, and Institute for Systems Research Outstanding Systems Engineering Faculty Award (2000–2001).

Tamal Mukherjee (S'89–M'95) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1987, 1990, and 1995, respectively. Currently, he is a Research Engineer in Electrical and Computer Engineering Department at Carnegie Mellon University.

His research interests include automating the design of analog circuits and microelectromechanical systems. His current work focuses on the developing computer-aided design methodologies and techniques for integrated microelectromechanical systems, and is involved in modeling, simulation, extraction, and synthesis of microelectromechanical systems.