Layout Verification for Mixed-Domain Integrated MEMS

Bikram Baidya and Tamal Mukherjee

***Abstract*—As design of integrated microelectromechanical systems (MEMS) mature, there is an increasing need for verification tools for such mixed-domain layouts. This requires a mixed-domain layout-versus-schematic tool capable of extracting an integrated schematic from the mixed-domain layout and verifying it against the design schematic. This paper reports on a prototype implementation of such a tool and an extraction methodology for integrated MEMS designs capable of capturing domain-specific parasitics. A custom schematic-versus-schematic tool is then used to compare the parameters and connectivity of schematic elements between the extracted and design schematic. Finally, simulation of the extracted schematic is used to analyze the behavior of the designed layout.**

***Index Terms*—Integrated MEMS, microelectromechanical systems (MEMS) extraction, MEMS layout-versus-schematic (LVS), parasitics, verification.**

I. Introduction

MICROELECTROMECHANICAL systems (MEMS) are miniaturized sensors and actuators from various disciplines of science (mechanical, optics, fluidics, etc.) manufactured using microfabrication techniques to take advantage of the benefits of integration, miniaturization, and batch fabrication, as has been demonstrated in conventional very large scale integration (VLSI). The late 1990s saw an increasing trend toward the design of complex MEMS and an increased effort toward its integration with electronics. An emerging approach to the design of integrated MEMS involves the use of schematic-level simulators capable of handling mixed-domain schematic [1]–[3], with the schematic view as the preferred design entry mode. This is followed by manual or automatic generation of layout along with layout-level modifications to meet design rule checks (DRCs) and connectivity requirements. Verification of such layouts requires a mixed-domain extraction and layout-versus-schematic (LVS) methodology capable of handling devices across different physical disciplines. This paper presents a prototype implementation of a mixed-domain extractor and LVS capable of handling integrated layouts containing miniaturized mechanical structures and electronics. While the MEMS market continues to grow at a phenomenal rate of more than 20% per year, its original focus of miniaturized mechanical structures has had the most steady growth rate. This class of MEMS components cover a wide range of applications, such as pressure sensors [4], RF switches [5], resonator filters [6], and accelerometers [7]–[9] and hence was chosen to be the focus of this work.

VLSI extraction focusses on identifying device parameters, interconnect parasitics, and device parasitics for an accurate representation of the layout at the schematic level [10]. Manual extraction of such device parameters and parasitics is prohibitively slow for large and complex VLSI layouts. As noted in [11], verification requirements for MEMS is more demanding than conventional VLSI. MEMS layout verification needs to check for layout correctness in the form of layout parameters, parasitic effects, and also geometrical issues such as symmetry and physical connectivity of MEMS elements. Hence, an accurate representation of the MEMS layout at the schematic level needs to capture the element parameters, element parasitics and interelement connectivity. As in VLSI, manual generation of such schematics is possible for small MEMS devices but is virtually impossible for large complex systems. The complexity in the MEMS case is two fold. First, the process of capturing parameter and connectivity correctness in the manually generated schematic for large layouts is a laborious process prone to human errors. Second, manual estimation of parasitics for large systems, having complex electrical properties embedded in the structure, is almost impossible. This motivates the need for an automated extractor for MEMS which can reconstruct a schematic corresponding to any given layout followed by LVS for integrated MEMS.

Initial attempts toward MEMS LVS have been limited to higher level connectivity analysis [11], [12] using tagged layouts to verify the design connectivity. Tagging of manually generated layout is tedious. Also, such methods fail to capture layout modifications following automatic layout generation, which might lead to significant parasitic effects or loss of layout symmetry. Fig. 1(a) shows a layout of a common centroid accelerometer consisting of a central floating plate suspended using four symmetrically placed crab-leg springs. The motion of the central plate due to any acceleration is sensed by four sets of differential comb drives arranged in a common centroid topology. Any motion in the horizontal direction causes a change in the capacitances on both sides of the suspended comb fingers, as they get displaced with respect to the fixed set of fingers, resulting in a differential electrical signal. Verification of

Manuscript received July 8, 2003; revised February 14, 2004. This work was supported in part by the Defence Advanced Research Projects Agency (DARPA) and U. S. Air Force Research Laboratory under Agreement F30602-99-2-0545 and Agreement F30602-01-2-0987 and in part by the National Science Foundation (NSF) under Award CCR-9901171 and Award CCR-0325344. This paper was recommended by Associate Editor M. D. F. Wong.

B. Baidya was with the MEMS Laboratory, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is now with the Intel Corporation, Hillsboro, OR 97124 USA (e-mail: bikram.baidya@intel.com).

T. Mukherjee is with the MEMS Laboratory, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: tamal@ece.cmu.edu).

Digital Object Identifier 10.1109/TCAD.2005.844100

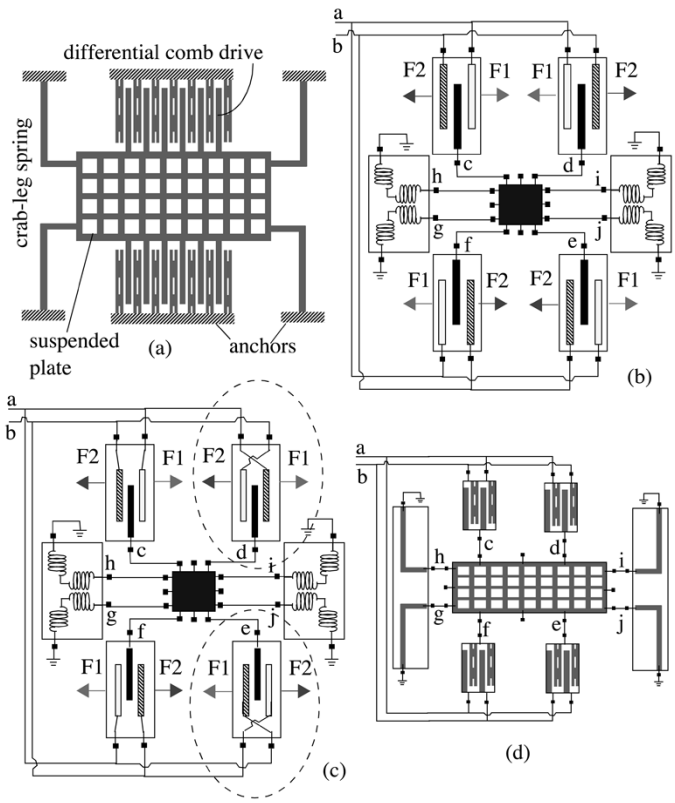

Fig. 1. Common centroid crab-leg accelerometer exemplifies MEMS verification issues. (a) Layout. (b) Tagged schematic showing the common centroid connection of the comb fingers for force and moment balance. (c) Erroneous system resulting from routing mistake which switches the voltages of the comb fingers in the two comb sets marked by circle. (d) Tagged layout corresponding to the tagged schematics in (b) and (c). The tagged layout is created by the automatic layout generator which replaces blocks in the schematic with corresponding layout standard cells. Since the block-level view for both schematics (b) and (c) are the same, the automatically generated layout is identical and hence overlooks the erroneous electrical connection in (c).

this device using schematic level simulation involves accurate representation of geometrical parameters (like length and width of beams) and connectivity of the layout at the schematic level. Both the electrical connectivity (for example, the electrical connectivity of comb fingers) and mechanical connectivity (for example the position of comb drives with respect to central plate) need to be captured. In addition, the schematic must also capture the layout symmetry. This includes mechanical symmetry (for example, the four sets of crab-leg spring must be symmetrically placed) and electrical symmetry. Fig. 1(b) shows the tagged schematic for the layout in Fig. 1(a). The schematic is able to capture the top level parameter and connectivity information but fails to capture any parasitics and layout errors that might be added after automatic layout generation. For example, an erroneous routing of the comb fingers in the layout switches the connectivity of the comb fingers on the right (top right and bottom right) resulting in the system shown in Fig. 1(c) cannot be captured using the tagged layout approach since the top level connectivity is still maintained. Such connectivity errors which change the electrical symmetry of the layout can dramatically change the behavior of the system. For example, in the original schematic, with the four comb drives electrically connected to form a common centroid topology [shown in Fig. 1(b)],

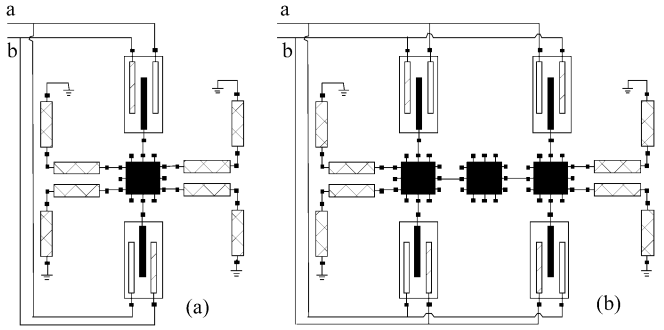

Fig. 2. Atomic level schematics. (a) Extracted schematic for the erroneous design in Fig. 1(c). (b) Atomic schematic corresponding to the original layout shown in Fig. 1(a).

both force (F1 and F2) and moment (arising from equal and opposite sets of forces F1 and F2) balance is achieved at the dc state. However, if this symmetrical connectivity is destroyed [as in Fig. 1(c)], the moments are no longer balanced, resulting in a dc twist of the device which causes unwanted second order effects during the operation of the device. A tagged layout, in which only block to block connections are tagged corresponding to both the common centroid and erroneous configurations would be identical. Any errors inside the blocks are hence neglected by the checker. This motivates the need for an extraction methodology that builds the extracted schematic directly from the layout, instead of using user-defined hints (such as connectivity tags in Fig. 1).

This paper presents a MEMS LVS methodology capable of handling both manual and automatically generated layouts. It uses an integrated MEMS extractor to convert the given integrated layout into a mixed-domain extracted schematic which can then be used to perform a schematic-versus-schematic comparison with the design schematic to verify connectivity, device parameters, and symmetry. It also points out symmetry in the design. The extractor presented here uses the extraction framework introduced in [13] which starts with the recognition of fundamental MEMS building blocks (atomic elements), which are then hierarchically combined to recognize functional components defined in the user library. Using such an hierarchy allows the extractor to capture connectivity and symmetry errors in the layout. Hence, for the example shown in Fig. 1(c), the extractor would merge the two sets of comb drives on either side of the plate, since they are composed of identical fundamental blocks, resulting in a schematic shown in Fig. 2(a). This, when compared to a correct atomic schematic [Fig. 2(b)] would highlight the connectivity error inside the comb drives. In addition, the extractor also identifies domain-specific parasitics which are annotated into the schematic elements for an accurate representation of the final layout.

The CMOS micromachining process [14] has been used as the representative integrated MEMS process in this paper. The ability to place multiple isolated metal structures along the depth of a mechanical structure and integrate electronics together with MEMS makes CMOS micromachining a complex and hence challenging representative fabrication process. It uses two maskless dry etches to release the microstructure protected by the top-most metal layer in foundry CMOS designs.

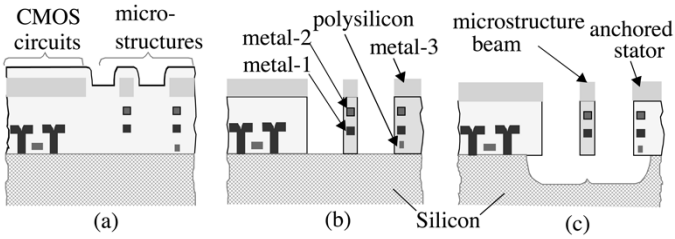

Fig. 3. Cross section of CMOS micromachining process [14]. (a) After standard CMOS process, (b) after anisotropic etch, and (c) on final release using isotropic etch.

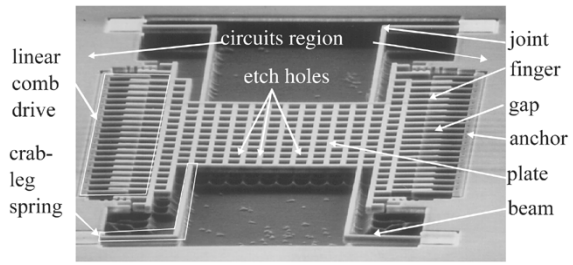

Fig. 4. SEM of a crab-leg accelerometer showing the atomic elements (on the right) and the functional elements (on the left) and the etch holes for post CMOS releaseAtomic.

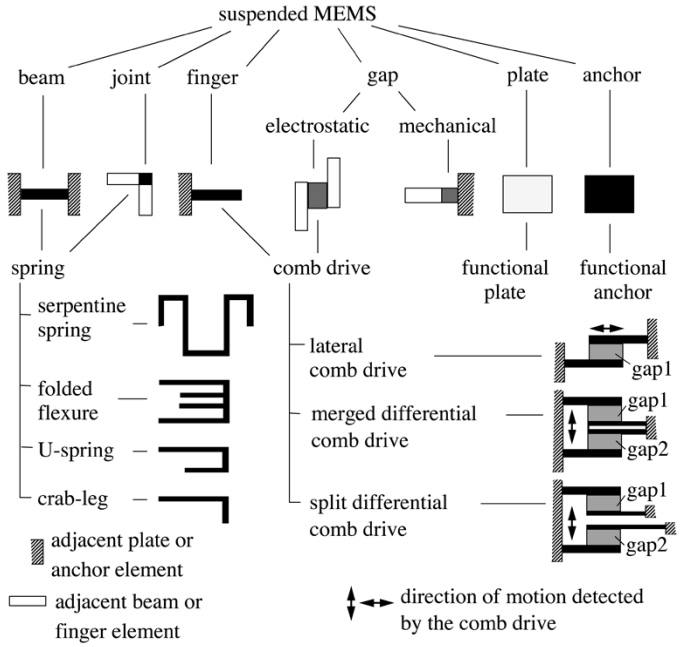

Fig. 5. Atomic (top line of figures) and functional elements (figures at the bottom) for suspended MEMS.

Cross sections of the microstructure during various points in the CMOS post processing are shown in Fig. 3.

Fig. 4 shows a SEM of a crab-leg accelerometer fabricated using this process. Such MEMS components can be hierarchically decomposed into *atomic* (labels on right in Fig. 4) and *functional elements* (labels on left in Fig. 4). The *atomic elements* (Fig. 5) form the fundamental building blocks for MEMS design and can be either fixed elements, like the *anchor* (which attaches the suspended structures to the substrate) or suspended elements like *plates, beams, joints* (differentiated based on their compliance and connectivity), or nonstructural elements like *gaps*. 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. As shown in Fig. 4, *etch holes* are needed in large *plate* areas to facilitate the release of the suspended MEMS areas. The desired structural compliance of the suspended structure is obtained by designing the topology and geometry of the beam network. 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. Two or more *beams* are connected by *joints*. *Joints* serve as logical connectivity elements. The suspended structure is connected to the base at the *anchors* which form the mechanical pillars supporting the entire suspended structure. The empty areas between suspended structures are defined to be *gaps*. *Gaps* may be either electrostatic or mechanical, depending on the electrical condition of the surrounding suspended structure. The *plate, beam, joint, finger, gap*, and the *anchor* form the *atomic element* set for suspended MEMS and are equivalent to elements like transistors, resistors, etc., in VLSI.

Atomic elements can be combined together to form *functional elements* which correspond to elements like opamps, comparators, etc., in the VLSI domain. Fig. 5 shows common *functional elements* along with some typical examples. *Springs* are composed of *beams* and *joints* and connect the suspended *plate* to the *anchors*. The decision on the type of *spring* to be used is based on the area and buckling constraints [15]. *Comb drives* are used primarily for electrostatic actuation and sensing of the MEMS component and are composed of *fingers* and *electrostatic gaps*. The *lateral comb* drive [16] allows actuation and sensing along the length of the *fingers* and uses alternating fingers having different electrical properties. The *merged* and *split differential comb drives* [17] use three sets of electrically isolated *fingers* for actuation and sensing in a direction orthogonal to the *finger* length. The electrically isolated fingers may be mechanically combined to from a single mechanical *finger* (e.g., the mechanical finger containing the middle two electrical conductors in *merged differential comb drive*) or may be mechanically separated (e.g., the fingers in *split differential comb drive*). The extraction and LVS methodology described here exploits this hierarchy while extracting the schematic from the layout geometry.

Section II describes the integrated MEMS design flow followed by the individual steps in extraction. Next, various domain-specific parasitics are discussed followed by the description of the prototype implementation of MEMS LVS. Finally, some results are presented highlighting the usefulness of the LVS and extraction methodologies in verifying integrated MEMS designs.

II. Integrated MEMS Design Flow

Fig. 6 shows the design and verification flows for integrated MEMS. The design flow starts with the separate designs for

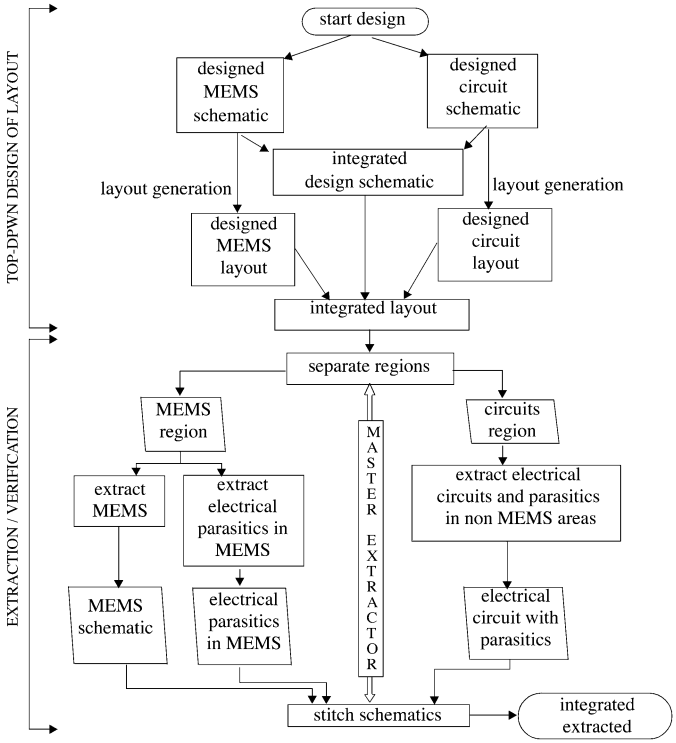

Fig. 6. Design and verification flows for integrated MEMS.

MEMS and circuit schematics followed by their integration to create the top-level integrated design. The top-level schematic is then simulated using mixed-domain simulators to verify whether the design satisfies the design specifications. Next, layouts for MEMS and circuits components are generated, either manually or automatically, which are then integrated to get the final integrated layout of the system. The electrical connectivity between different blocks is also done in the integrated layout. This step is usually done manually since the automatic layout generators for MEMS are usually not capable of performing routing of electrical connections in the MEMS areas without violating mechanical specifications of the MEMS device. The integrated layout is then verified using the integrated verification flow presented in this paper.

The core of the integrated verification flow is the extraction of integrated schematic from the integrated layout. To keep the flexibility of both layout and schematic design entry, the integrated extractor starts directly from the layout without any hints from the design schematics. While the geometric operations involved and the layout representation used is same across different domains, the recognition algorithms depend on the domain. This is needed to avoid the confusion stemming from the geometrical similarity of different structures in different disciplines. For example, an L-shaped structure might be an interconnect in the electrical domain (represented as a resistance with current crowding models for the bend) or a crab-leg spring in the mechanical domain (represented as two beams connected by a joint). Hence, integrated extraction requires partitioning of the input layout geometry into various physical domains and application of domain-specific recognition heuristics during the element recognition phase. Hence, the L structure will be recognized as a spring by the mechanical extractor and as an interconnect by the electrical extractor. This partitioning could be done internally by the extractor. However, extractors for electrical elements are already available from commercial vendors. The approach taken here is to automatically separate the integrated layout into circuits and MEMS regions and use domain-specific extractors to extract schematics along with domain-specific parasitics. Design rules for post-CMOS etching are used by the top-level master extractor to determine the areas of the layout where the underlying silicon will be etched away to result in suspended MEMS areas. The electrical parasitics in the MEMS regions are also extracted using the commercial VLSI extractor (as shown in Fig. 6). Finally, the individual schematics are stitched together by the master extractor to obtain the integrated extracted schematic. The integrated extracted schematic can then be compared with the integrated design schematic to perform a LVS for integrated MEMS.

The sections that follow describe the representation and the steps involved in MEMS domain-specific extraction.

A. Representation

MEMS extraction starts with the representation of layout geometry along with the structural and connectivity information at every region of the geometry. The extractor uses the extraction rules file to create derived layers which reduce geometric complexity of the design and also help in the recognition heuristics described later. For example, one such layer is the *MEM-layer*, which can be obtained from logical OR of the metal layers in the MEMS region (for the CMOS process [14]) together with the structural holes in the layout. The corresponding rule in the rule file would be $\mathrm{MEM} = M1 + M2 + M3 + MH1 + MH2 + MH3$, where Mi stands for the ith metal layer and MHi stands for the hole layer corresponding to the ith metal layer and '+' sign is used to symbolize logical OR between the layers. The *MEM-layer* defines the actual geometry of the structure. All the layers (derived or already existing) are represented in an unique canonical representation described later.

To maintain all the layer information while simplifying the geometric representation, a hierarchical bin representation, similar to the quad tree [18] representation in VLSI, is used to represent all the layers as shown in Fig. 7. The bottom-most layer in this hierarchy stores the information of individual structural layers (the metal layers in the CMOS process [14]) which are used to derive the *MEM-layer*. The *MEM-layer* together with the empty regions in the layout form the next level of hierarchy, the bin level. The first iteration of recognition occurs in the bin or middle level to identify the *atomic elements* in the layout. Each bin also stores the information about its parents in the lower level of hierarchy. The geometry in the structural layers (in the bottom level of hierarchy) are further split by the edges of the *MEM-layer* so that each rectangle in every layer belongs to an unique bin. For example, the plate bin rectangle A has two unique parents in bottom level, belonging to different metal layers, as shown in Fig. 7. Contiguous sets of recognized bins of the same type are merged together to form the topmost storage level, the superbins, where the final iteration of recognition identifies the *functional elements*. Having such a hierarchy allows the extractor to obtain any structural or electrical connectivity information, that may be needed during recognition, by

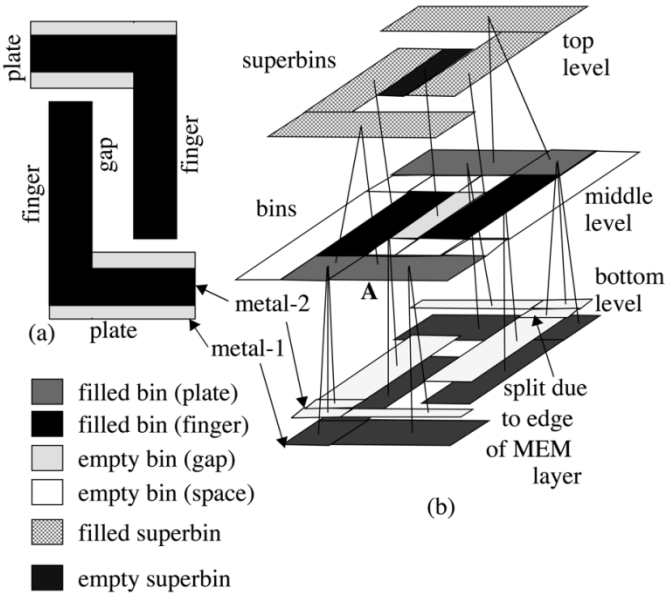

Fig. 7. Hierarchical bin data structure. (a) Example layout of two cantilever fingers and (b) its bin storage.

traversing down the hierarchical storage. The information stored in the lower levels of hierarchy is also used in the final schematic generation stage where such the structural information of each functional element is annotated for accurate schematic representation. Examples of such information include the percentage of different metal in a functional block, the center of mass for each metal in a functional block, etc.

B. Canonization

In order to simplify the recognition heuristics used during extraction, the geometrical information of the layout needs to be represented in an unique way irrespective of the designed geometry. We use a fully fractured representation of the layout geometry which we refer to as the *canonical representation* [13]. To define the *canonical representation*, we first classify the edges of the geometry into *external* and *internal edges* and can be defined to have two *faces*, one on either side of them. A *face* of an edge is said to be a *visible face* only if there exists a point, internal to the layout geometry (the inside of the geometry defined by the layout, e.g., the inside of a rectangle), from which a line, perpendicular to the edge, can be drawn without intersecting any other edge. *External edges* form the boundary of the geometry and can have only one *visible face* (the side which faces the inside of the geometry). In contrast, the *internal edges* have visibility on both their faces. Two edges are said to be *mutually visible* if they are parallel and if a line perpendicular to both the edges can be drawn, between *visible faces* of the edges, without intersecting any other edge. The area internal to the geometry that is visible to both of the *mutually visible edges* is called the *visible area* for the pair of edges. The *canonical representation* uses *the minimum number of rectangles to cover layout area between mutually visible parallel edges of the geometry such that each rectangle and polygon after partitioning has at most one neighbor per edge and each edge is either fully covered by a neighbor or not covered at all*. For Manhattan geometry, this reduces to a representation which uses *minimum number of rectangles to cover the layout area such that each rectangle has at most one unique neighbor per edge*. For simplicity and as an initial proof of the concept, the prototype implementation of extractor presented here, and hence the MEMS LVS, will deal with Manhattan layouts only. However, the definition of *canonical representation* can be used for non-Manhattan geometry and has been used to represent microfluidic layouts in microfluidic extraction [19]. A similar approach can be used to verify complex mechanical microstructures containing non-Manhattan geometries.

The canonization algorithm for Manhattan geometry uses a scanline algorithm which relies on the sorting of the vertical edges of the geometry. The sorted edges are then sequentially added into the scanline resulting in a scanline motion along increasing abscissa coordinates (i.e., the scanline moves from left to right). The sequence of events associated with each insertion and deletion is explained in the algorithm below. The output of the canonization algorithm is a set of canonical rectangles representing the layout. Each rectangle stores the coordinate information and a pointer to the neighbor adjacent to each edge of the rectangle.

```
SORT (edge_set E)
 Assign direction to edges such that the
  layout geometry (e.g., inside of a
  rectangle) is to the left when traversing
  along the edge.
 Sort the directed edges in E with respect to
  their abscissa, then their ordinate and
  finally their angle to abscissa (edge
  pointing upward makes 90° with abscissa and
  the edge pointing downwards makes 270°)
  and return the sorted set of edges
OVERLAP (edge a, edge b):
 returns TRUE if edges a and b are collinear
  and overlap in coordinates
INTERSECTION(edge_set A, Edge_set B)
 find set of edges (I) in A that overlap
  completely with edges of B
create rectangles corresponding to each edge
  in I
return I
SPLIT(edge e, point_set P, scanline S):
 split edge e by points of P
 return the set of edges created from
  splitting e
SPLIT_AND_PROPAGATE(edge e, Point_set P,
  Scanline S)
 split edge e by points of P and propagate the
  splits to the rectangles and edges to the
  left of e replace edge e in S with its
  split parts
 return the set of edges created from
  splitting e
MODIFY (edge x, edge e, Scanline S):
 point_set V_x = {vertices of edge x}
 point_set V_e = vertices of edge e}
```

```
 for i = 1 to length[Ve]/*
 split and propagate to left*/
  edge_set X = SPLIT_AND_PROPAGATE(x, V_e[i], S)
 for i = 1 to length[Vx]/* split current edge a*/
  edge_set E = SPLIT(e, V_x[i], S)
 e = topmost edge of E/* replace current edge*/
 return INTERSECTION
(X, E)/*return duplicate edges */
CANONIZE(edge_set E):
 SORT(E)
 scanline S = NULL
 for i = 1 to length[E]
  pushS, E[i]
  edge c = E[i]
  edge n = successor of edge c in S
  while OVERLAP(n, c) is TRUE
   edge_set I = MODIFY(n, c, S)
   S = S − I/* remove duplicate scanline edges */
   if c is a closing edge
    S = S − I/* remove closed scanline edges */
   n = successor of edge n in S
  edge p = predecessor of edge c in S
  if OVERLAP(p, c) is TRUE
  edge_set I = MODIFY(p, c, S)
  S = S − I/* remove duplicate scanline edges */
  if c is a closing edge
   S = S − I/* remove closed scanline edges */
return
```

A red–black tree was used for the scanline data structure for guaranteed $O(\log n)$ insertion and deletion of edges. The total time taken in the functions OVERLAP and INTERSECTION is $O(m)$ where m is the final number of rectangles in the canonized representation. Each edge insertion and deletion takes $O(\log k)$ time, where k is the current number of edges in the scanline (having an expected value of $O(\sqrt{m})$. Hence, the total time taken for the canonization is $O(m+\mathrm{e}\log\mathrm{k})$ where e is the initial number of vertical edges.

Fig. 8 shows an example of canonization using the algorithm described. Notice that when edge c is inserted (scanline position denoted by $currentX$ at 2) in the scanline (S), the rectangle A gets completed while rectangle B is not completed, as edge a on S can be reached from c while edge b cannot be reached. Also when edge e is inserted $(currentX = 4)$, the edge a, and the rectangles to its left, are split by the top of the edge e.

The canonization is followed by a neighboring phase where the neighbor information of each rectangle is assigned using two cycles of sorting and comparison of consecutive rectangles. For finding the vertical (horizontal) neighbor information, the rectangles are sorted first with their abscissa (ordinate) and then with their ordinate (abscissa).

The above algorithm is for intralayer canonization of a single layer. The interlayer canonization and the binning algorithms use similar scanline algorithms where the edges corresponding to different layers are color coded and the decision at each insertion and deletion uses the color of the edge along with the overlap information.

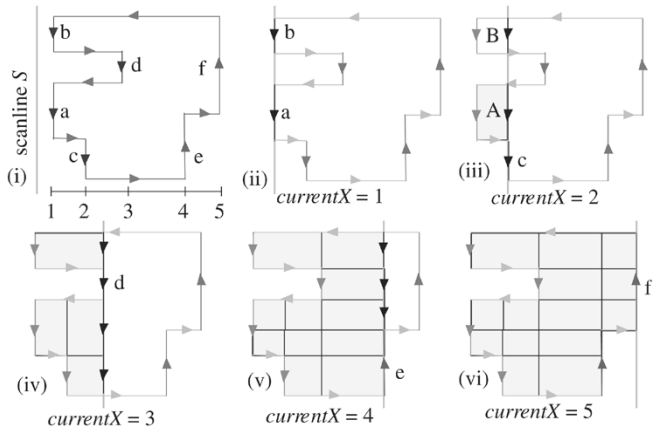

Fig. 8. Example demonstrating the canonization procedure. The contents of scanline S and the state of the canonical set of rectangles are shown in consecutive figures are shown for each value of current X.

C. Recognition of Atomic Elements

Once canonization is complete, element recognition occurs in two steps. The first step recognizes the atomic elements which are then used to recognize the functional elements (Fig. 5).

Recognition of *atomic elements* involves classifying the *bins* into different atomic types (*plates, beams, fingers, gaps, holes, and anchors*). The recognition routines use rules based on geometry and neighbor information of the *bins*. All recognition routines use two iterations of operations, each having linear time complexity. The recognition routines using overlap of sets of rectangles use the scanline algorithm described earlier and hence have $O(k + a\log b)$ time complexity, where k is the total number of overlaps, a is the number of rectangles in the source layer (the layer being used to check the overlap), and b is the number of rectangles in the target layer (the layer in which the overlap information will be stored, usually the *bins*). The first iteration marks out the *bins* that can potentially be the element being recognized and the second iteration uses stricter rules to check the potential set of *bins* to confirm the recognition. The subroutines use geometric relations like aspect-ratio, ratio of width of the *bin* to width of neighbor, etc., which can be user-specified. Given below is a short description of the algorithms used for recognizing the *atomic elements*.

The first *atomic elements* to be recognized are the *holes*. Unlike the other *atomic elements*, which are recognized after the creation of the *bin* layer, *holes* are recognized during the creation of the *bins*. The first iteration starts with the creation of a preliminary copy of the *bins* layer by canonizing the bounding box of the layout by the empty regions of the layout. Next, the rectangles in the preliminary *bin* layer are checked with rectangles in the bottom hierarchical level to obtain overlap information (this is used to create the links between the two levels of hierarchy shown in Fig. 7) between the two sets of rectangles. The Overlap information is used to mark *filled* and *empty rectangles* in *bin* layer. The Rectangles in *bin* layer having metal layers as parents are marked as *filled bins* and the rest are marked *empty*. Finally, the *empty bins* that are completely surrounded by *filled bins* are marked as *potential holes*. In the next iteration, the width of each *potential holes* is compared with the width of its *filled neighbors* (or with the combined width of a contiguous set

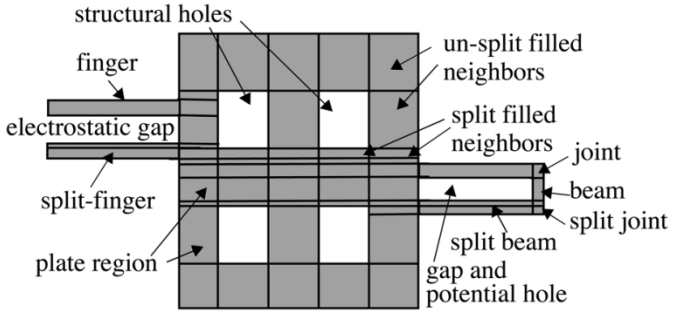

Fig. 9. Example showing split and unsplit atomic elements.

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	

of filled bins neighboring the potential *hole*). If this ratio (hole width versus neighbor width) is found to be less than an user-defined critical aspect ratio, then the *potential holes* is confirmed to be a *hole*. Fig. 9 shows examples of *holes* having split and un-split filled neighbors. The rest of the *empty bins* are marked as *gaps*.

The *filled bins* overlapping with *anchor* layer (a derived layer for CMOS micromachining which is defined as metal areas larger than the lateral etch depth) are marked as *anchors*. This is followed by the recognition of *beams*. In the first iteration to recognition of *beams, filled bins* having non-*beam filled bin* neighbors only on their shorter sides and *gaps* as neighbors on their longer sides are marked as *potential beams*. If one of the longer sides has a *gap* neighbor while the opposite side has a *filled bin* neighbor, then the *bin* is marked as a *potential split-beam*. Contiguous filled bins between two *potential split-beams* are also marked as potential split-beams. In the next iteration the aspect ratio of each *potential beam* (or the aspect ratio of a contiguous set of *potential split-beams*) is compared with an user-defined ratio to mark them as *beams* (or *split-beams*). Examples of *beams* and *split-beams* are shown in Fig. 9. Similarly, cantilever *beams* are marked as *fingers*. *Split-fingers* (Fig. 9) are also recognized using the same approach used for *split-beams*. Next, *plates* are recognized using hints from overlap with the *holes*. Such areas act as seed layers for an expansion of *plates* into neighboring unrecognized *structural bins*, which are also marked as *plates*. *Filled bins* connecting two or more *beams* are recognized to be *joints*. *Split-joints* (Fig. 9) are also recognized using the same approach used for *split-beams*. Finally, the *electrostatic gaps* are recognized using the electrical property of *filled neighbors of gaps*. The electrical properties of the *filled bins* are obtained by traversing the hierarchy to the bottom level where the electrical information is stored. If a *gap bin* or a contiguous set of *gap bins* have two *filled neighbors* of different electrical connectivity then the set of *gaps* is recognized to be an *electrostatic gap*.

D. Recognition of Functional Elements

The next step in element recognition is *functional element* recognition, which occurs after contiguous sets of similar *bins* have been merged to create the *superbins* in the *hierarchical bin representation*. The *functional plate* and *functional anchor* are automatically formed during the creation of the *superbins* from the *bins*. The most challenging aspect of *functional element* recognition is the recognition of *springs* and *comb drives*, for which a finite state machine (FSM)-based algorithm has been used. The detection routine first reads in a user-defined library file which defines the elements for the FSM and also defines the recognition transition state diagram. This is used to create the FSM through which contiguous sets of *beams, joints,* and *electrostatic gaps* are passed to either recognize or not recognize the set to be a *spring* or *comb drive*, as the case may be. The FSM can be defined by

M = {Q, S, L, G, F, X}, where
Q states $= \{S, \{\text{intermediate states}\}, F, X\}$;
S start state = *anchor* point;
L inputs or the language set $= \{\{\text{joints}\}, \{\text{beams}\}, \{electrostatic\_gaps\}, \text{NULL}\}$;
G transition rules;
F set of final states;
X exit state.

A *joint* in the FSM input element list 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 whether the twist direction is anticlockwise or clockwise. The six types of *joints* possible using such a convention are shown in Table I. *Beams* in the FSM language set are first differentiated based on the electrical properties of the embedded electrodes in the *beam*. Next, copies of the different types of *beams* are defined in the library file depending on how many unique *beams* of same electrical configuration are needed to define the *springs* in the library. For example, a *U-spring* requires three *beams* (Fig. 5) 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. 5. Similarly, for the *comb drive* library, copies of *fingers*, and *gaps* required are defined in the FSM language set. The *fingers* are also differentiated based on their electrical property. An *simple finger* is used to define a *finger* having uniform electrical property throughout the mechanical structure while a *differential finger* is used to represent *fingers* having two different electrical conductors embedded in the mechanical structure.

Fig. 10 shows a small example of an FSM that can be used to recognize any *crab-leg spring* in any given layout. Fig. 10(a) shows the two possible topologies for a *crab-leg spring* and Fig. 10(b) shows the FSM which uses two unique *beams* (a and b), two types of *joints* ($J+$ and $J-$) and the *null* element (ϕ) as its language set (L). The start state S goes to the next valid

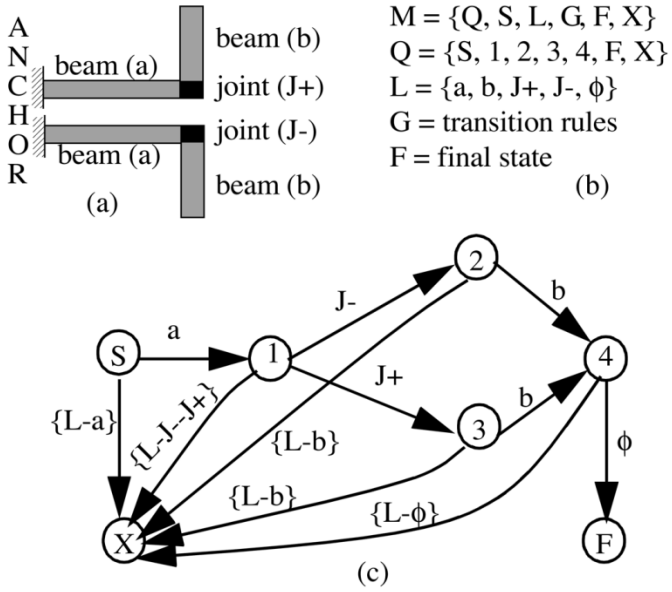

Fig. 10. Example demonstrating the FSM based recognition algorithm for springs and comb drives; (a) schematic of the two possible topology for a crab leg, (b) the FSM definition for recognition of a crab leg spring only, (c) graphical representation of the transition rules.

state (1) only if it encounters a *beam* (a). Next joints of type $J-$ or $J+$ takes the FSM to the valid states 2 and 3 respectively. A *beam* (b) at any of these states takes the FSM to state 4, at which point a *null* is needed to reach the final recognized state (F). A *null* element is encountered when the contiguous set of *beams* and *joints*, being checked for the *crab-leg spring*, is exhausted. Any deviation from the input sequence mentioned above leads the FSM to the state X which represents an *unrecognized spring*.

The overhead of setting up the library of *springs* and *comb drives* is linear with respect to the number of states defined in the library (m). The detection part of the algorithm takes $O(mn)$ time, where n is the total number of contiguous sets of *beams, joints* and *electrostatic gaps* in the given layout.

III. Parasitics

Automated recognition and extraction of domain-specific parasitics form a crucial part of MEMS extraction. Parasitics in MEMS can be classified into two types. The first type of parasitics have their origins in the assumptions of boundary condition and response function used in the derivation of lumped parameter models. Such assumptions are used to convert the partial differential equations representing the physics of the component into ordinary differential equations, resulting in simple lumped parameter models that can be solved by fast schematic simulators. However, such assumptions may not be valid for certain configurations of design. One common approach to represent such designs is to replace a single (lumped) component with several (distributed) components (updated citation number). Examples of such parasitics include the parasitic joints discussed later in this section. Such parasitics can also be detected by checking the schematic for the connectivity and element information using a schematic analysis schematic analysis tool that uses the same rules used by the extractor.

The second type of parasitics occurs from the need for accurate representation of layout information in the schematic models. Manual calculation of layout details in the design (like extra mass in the plate area of suspended MEMS due to routing, parasitic capacitance occurring due to the electrical wires in suspended MEMS) is cumbersome and impossible for large designs. This leads to approximate estimations of such layout information during manual creation of schematic. However, the simulation can be very sensitive to errors in such calculations. Such detailed layout information, which can significantly modify the response of the simulation, are classified as the representation parasitics. Such parasitics cannot be captured by schematic analysis alone since they need layout-specific information. Hence, they can only be detected by an extractor that creates the schematic directly from the layout.

The two domains investigated for the prototype implementation lead to two types of parasitics: electrical and mechanical, which will be described in paragraphs that follow.

Electrical parasitics, like parasitic capacitances significantly effect the operation of electromechanical elements like *comb drives*. Similarly, electrical parasitic capacitances [Fig. 11(a)] in MEMS areas tend to affect the interface between MEMS and circuits and hence need to be accurately captured in the integrated schematic. While estimating the parasitic capacitance in the suspended MEMS areas the gap between the substrate and the suspended structures, due to the release etch [Fig. 3(c)], needs to be taken into account. These process-related information are encoded into the extraction file used by the master extractor (Fig. 6). The extraction of electrical parasitics is done using commercially available IC extraction tool (as shown in Fig. 6) after incorporating the modifications due to the difference in geometry (like gap between structure and substrate in suspended MEMS) in the MEMS areas, shown in Fig. 11(a).

Mechanical parasitics that more accurately capture layout details for schematic simulation include the effect of parasitic mass in the *plates* and the effect of *parasitic joints* between *beams. Parasitic joints* [Fig. 11(b) and (c)] occur between *beams* which differ significantly in their widths. Such *joints* need to be modeled using *plate* elements to accurately capture the transfer of moments between the *beams*. The integrated extraction tool presented here identifies the *parasitic joints* by comparing its adjacent *beams* and decides based on an user specified ratio of the widths of *beams*. If a *joint* is determined to have parasitic effects, it is represented as a *plate* element instead of neglecting it in the schematic. A *joint* is also determined to be parasitic if its dimensions are similar to any of the lengths of its adjacent *beams*. Parasitic mass in *plates* [Fig. 11(d)] results due to routing in lower metal layers (in a multilayer process like the CMOS-MEMS [14] process where the top layer is used to define the suspended structure and the lower layers are used for connectivity) and also from the etch-holes in the *plate*. These parasitics are annotated as parameters to the *plate* models extracted by the MEMS extractor. For example, the presence of lower routing metal layers in the *plate* is accounted by the mass factor for each metal. This is demonstrated in the example in Fig. 11(d) shows a three metal CMOS-MEMS example. Note that the

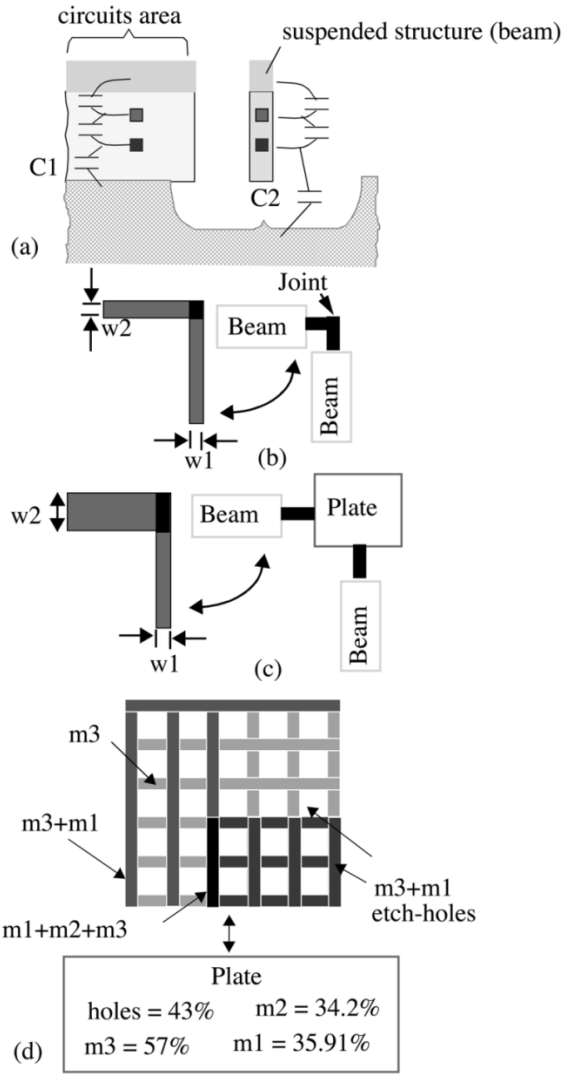

Fig. 11. Electrical and mechanical parasitics: (a) parasitic capacitances, (b) a joint between two nearly equal beams neglected in the model, (c) a joint between two beams of different aspect ratios modeled as plate to account for its parasitic effects, and (d) layout of plate area in MEMS and corresponding schematic with parasitic values.

percentage of metal3 is less than 100% because of the presence of etch holes in the structure. Other physical parameters, like center of mass, moments of inertia, etc., which capture the distribution of mass in the area, being represented by the *plate* element, are also annotated in the extracted schematic. While the *parasitic joints* can be included in the design schematic by the designer based on past experience, the *parasitic mass* in *plate* region is entirely dependent on the layout and cannot be accurately calculated before completing the layout. For large designs, manually evaluating when a *parasitic joint* is needed also becomes cumbersome and leads to the designer neglecting them. The extractor automatically inserts the *parasitic joints* and also calculates *parasitic mass* where required resulting in an accurate schematic representation of the layout.

IV. LVS

The final step in the LVS process is a comparison between the designed and the extracted schematic which is done in three parallel steps corresponding to the three schematics in Fig. 6 (top-level schematic and the two domain-specific schematics). At the top-level, the block connectivity between the circuit and MEMS subschematics are compared using a tagged methodology as in [11] and [12]. For the two subschematics (circuit and MEMS), separate LVS tools are used. A commercial circuit LVS tool is used to compare the circuit subschematics while the MEMS subschematics are compared using a custom MEMS LVS tool which is described below.

LVS for VLSI circuits compares nets between the power line and ground. In MEMS circuits, paths between two anchors or floating end points obey the same Kirchoffian laws as the VLSI nets and hence the MEMS LVS tool tries to match such paths between the extracted and design schematics.

LVS for MEMS starts with the enumeration of trees for the extracted and design schematic. Each tree contains paths starting from one anchor to all the other anchors in the schematic. All such trees are enumerated for the extracted schematic while only one randomly selected tree is enumerated for the design schematic. The nodes in the path correspond to the MEMS circuit elements and the branches contain the information about the relative change in direction when traversing between the elements. Unlike VLSI, where the relative physical position between circuit elements is not important, the behavior of a MEMS component can vary significantly for any change in the relative placement of the MEMS elements. Hence, representation of the changes in direction in the tree is important. In addition, only the relative change in direction (directed angle) is important because the component as a whole can be rotated about any axis without causing any change in the final behavior of the component. If the MEMS LVS is used for checking a group of mechanically disconnected suspended MEMS structures (e.g., a pair of accelerometers meant to detect acceleration along orthogonal axes), the direction change should be defined with respect to a global frame of reference defined for the entire system. In this work the MEMS part of the integrated system is assumed to be composed of a single mechanically connected suspended structure. Tree matching algorithms are then used to match the tree from design schematic with the forest of trees from the extracted schematic. For every pair a match-figure corresponding to the amount of match is assigned. Two elements are defined to be matched if the parameters corresponding to the elements are equal. Such parameters can be user defined to change the degree of match required for any element. For example, to define that a beam element is matched, the parameters checked may be the length and width of the beam. If the user wants the LVS to be more strict, additional parameters, such as the number of metal layers in the beam, can be added to the parameter list. Similarly, two branches are defined to be matched if they correspond to the same relative change in direction. Two trees are defined to be matched if their nodes and branches match. If a complete match is found between the tree from the design schematic and any of the trees from extracted schematic, the LVS reports a success. If such a match is not found, an unsuccessful LVS is reported with the best possible match highlighted.

The MEMS LVS tool also reports the symmetry in the layout by comparing the trees from the extracted schematic. A pair of

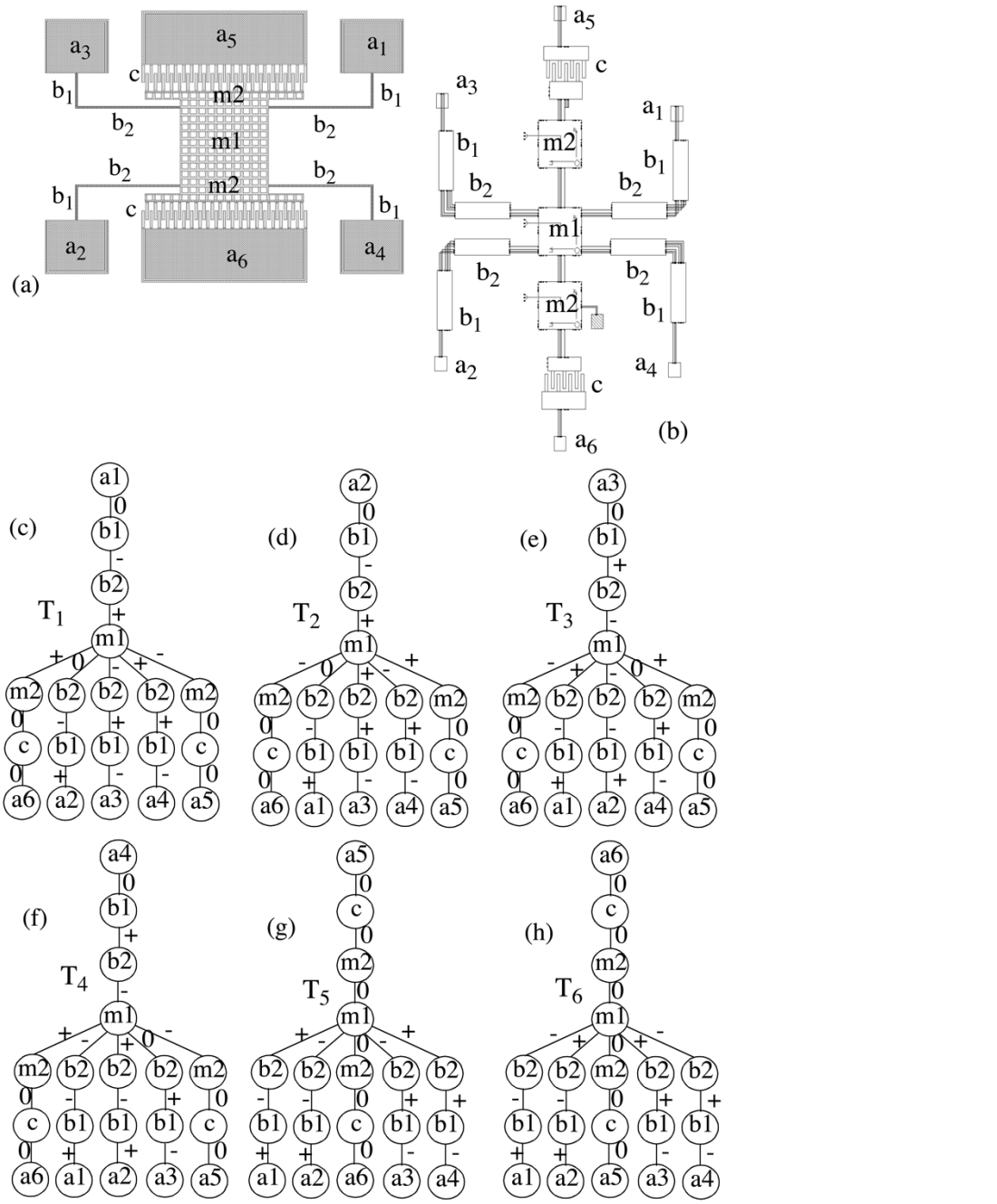

Fig. 12. Crab-leg accelerometer; (a) layout, (b) schematic, (c)–(h) forest of trees (T_1 to T_6) starting from the anchors a_1 to a_6 respectively for the extracted schematic.

trees which correspond to symmetric paths in the layout will have matching nodes and either matching or opposite branches. i.e., the trees will either match completely or will match if the relative change in direction on the branches of one of the trees is reversed. The MEMS LVS tool can highlight such symmetries by reporting such tree pairs. Since the goal of the verification flow presented in this paper is to check the correctness of the final layout, only the extracted schematic was used to check symmetry. However, the symmetry in the design schematic can also be checked, using the process mentioned here, during the design phase of the flow.

V. Results

This section presents examples of the LVS and the extraction methodology presented in this paper.

A. Crab-Leg Accelerometer

Fig. 12 shows an example of a crab-leg accelerometer which was verified using the LVS methodology described in this paper. Fig. 12(a) and (b) show the layout and corresponding schematic for the accelerometer, respectively. For a fully symmetric layout, the beams in the four sets of crab leg will match and hence

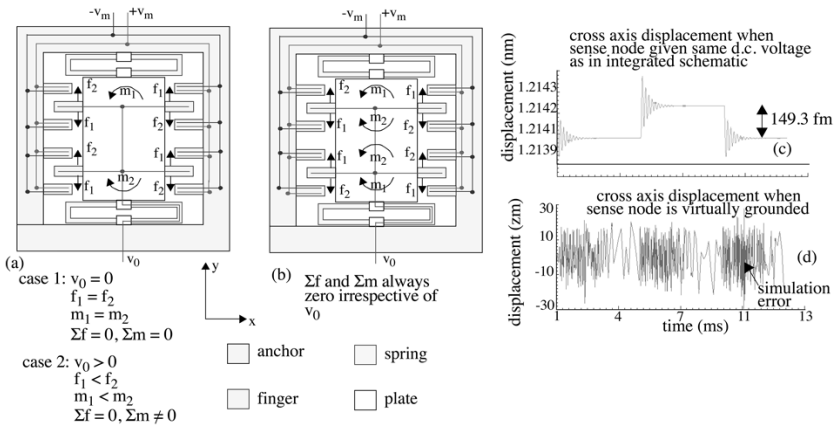

Fig. 13. (a) Symbolic picture of a rotor (center) with differential comb drive showing all the forces and moments due to a dc voltage on the sense finger. (b) Arrangement of four comb drives to remove moment due to dc voltage, cross-axis displacement for 1g input pulse (c) with and (d) without dc voltage at the sense node of the MEMS schematic in (a). Note that the signal in (d) is less than the simulation accuracy and hence suggests that there is no cross-axis signal.

can be represented by only two beam types (b_1 and b_2 as shown in Fig. 12). Similarly the two comb drives will also be similar (marked by c in Fig. 12) and the two plate elements on either side of the central plate (m_1) will also match (and hence denoted as m_2). The forest of trees corresponding to such an extracted symmetric schematic are shown in Fig. 12(c)–(h), with each tree starting from different anchors. The anchors have been represented using different symbols $(a_1$–$a_6)$ to make the trees easier to understand but have no difference from LVS perspective. The branches of the trees represent the relative change in direction when moving between the centers of two consecutive elements. A clockwise or an anticlockwise rotation is represented by a $-$ and a $+$ sign, respectively, while no change in direction is represented by 0. The MEMS LVS tool enumerates the full forest for the extracted schematic and any one random tree [for example, tree T_1 in Fig. 12(c)] for the design schematic. Next, it tries to match the schematic tree with each tree in the forest corresponding to the extracted schematic. In this example, because of symmetry, the tree T_1 for the design schematic will match completely with two trees, T_1 and T_2, from the extracted schematic forest resulting in a successful MEMS LVS. Next, the MEMS LVS tool tries to find symmetry by mutual comparison of the trees in the extracted schematic forest. For the example shown here, the tree pairs (T_1, T_2), (T_3, T_4) and (T_5, T_6) match completely. Also, the tree pairs (T_1, T_3), (T_1, T_4), (T_2, T_3), (T_2, T_4) and (T_5, T_6) match if the signs in one of the trees is reversed. This highlights the symmetry in the accelerometer in this example.

B. Integrated CMOS-MEMS Accelerometer

This example illustrates the importance of considering electrical symmetry and parasitics in design of integrated CMOS MEMS and how the integrated extraction methodology helps such analysis. Consider the CMOS-MEMS y-accelerometer [9] with an on-chip capacitive sensing interface. The initial design uses a *differential comb drive* on each side of the accelerometer as the electromechanical transducer. The symbolic representation of the design with the *comb drives* is shown in Fig. 13(a). The purpose of the accelerometer is to sense any acceleration in the y direction (sense direction). Any acceleration in the y direction results in the motion of the central *plate* causing the *comb fingers* attached to the *plate* (*rotor fingers*) to move relative to the fixed (*stator*) *fingers* [represented in Fig. 13(a) by the *fingers* connected to *anchor*]. This results in a change in capacitance on both sides of the *rotor finger* causing a differential electrical signal which can be used to electronically detect and measure the acceleration. Ideally, the motion along the two axes (x and y) should be decoupled to avoid erroneous outputs, i.e., an acceleration in x direction should not cause a movement in y direction (resulting in an erroneous electrical signal from the *differential comb drive*). This is referred to as cross-axis coupling and is measured by checking the motion along the x axis due to acceleration in the y direction. The amount of cross-axis rejection is used as a measure of the goodness of the designed accelerometer. An isolated simulation of the MEMS design schematic shows no cross-axis sensitivity. However, this information is misleading. When the integrated schematic (i.e., the MEMS schematic along with the electronics) was simulated, a cross-axis signal was observed (experiment 1 of Table II). This attributed to the dc voltage on the sense fingers (which in this case are the *rotor fingers* attached to the suspended central *plate*) of the *comb drives*, resulting from the sense circuit's operating point, which results in an unbalanced moment. This moment leads to a dc rotation of the suspended structure and in turn sensitizes the accelerometer to the cross axis acceleration. To verify this effect, the MEMS schematic (without the circuit) was simulated with different dc voltages on the *sense fingers*. The results plotted in Fig. 13(c) and (d) show the dependence of cross-axis displacement to dc voltage at the sense node of the MEMS schematic. This dependence can be eliminated if a common centroid topology [shown in Fig. 13(b)] is used. Such an arrangement leads to complete balancing of the forces and the moments thus eliminating cross coupling (experiment 2 in Table II) due to the circuit operating point on the comb drives.

Fig. 14(a) shows the layout of the common centroid accelerometer (with integrated electronics) showing the perforated central *plate* [Fig. 14(b)] and the details of the *differential comb fingers* [Fig. 14(c)]. The corresponding extracted MEMS schematic is shown in Fig. 14(d). The MEMS extractor first recognizes the *atomic elements* from the geometry of the layout followed by the detection of *functional elements* using the connectivity (both electrical and mechanical) and neighbor

TABLE II
RESULTS FOR ACCELEROMETER WHEN A 1G INPUT PULSE ACCELERATION WAS APPLIED IN y DIRECTION

Experi-ment	circuit	MEMS	x displacement (fm)	y displacement (nm)	o/p of sense circuit (mV)
1	design schematic (D_{ca})	design schematic, not common centroid (D_{ma})	149.3	3.86	5.56
2	design schematic (D_{ca})	design schematic, common centroid (D_{mb})	0	3.86	5.56
3	extracted schematic, metal2 wiring (E_{ca})	extracted schematic, common centroid (E_{ma})	28.89	3.6	5.27
4	extracted schematic, metal1 wiring (E_{cb})	extracted schematic, common centroid (E_{ma})	28.89	3.6	5.29
5	design schematic (D_{ca})	extracted schematic, common centroid (E_{ma})	28.89	3.6	5.29

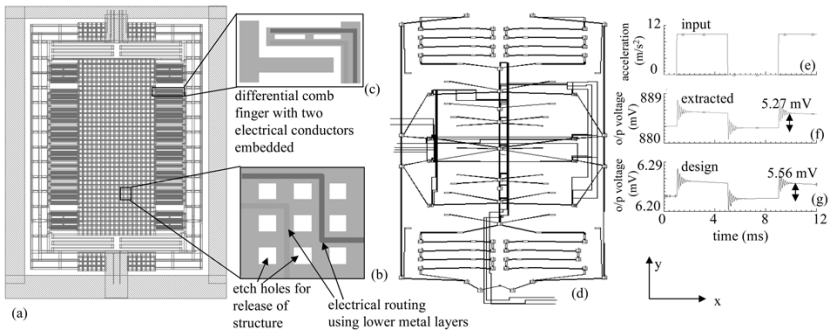

Fig. 14. Lateral CMOS-MEMS accelerometer [10]. (a) Layout. (b) Perforated plate mass. (c) Complex connectivity in differential fingers. (d) Extracted schematic; transient simulation in NODAS showing (e) input acceleration, (f) o/p voltage response of the extracted schematic, and (g) o/p voltage response of the design schematic.

information of the *atomic elements*. This hierarchical recognition helps detect any parametric or connectivity error in the layout which get accurately represented in the extracted schematic. The extracted and design schematics were simulated using an input acceleration pulse [Fig. 14(e)]. A comparison between the results of extracted [Fig. 14(f)]) design [Fig. 14(g)] schematics (experiments 2 and 3) shows a 6.7% degradation in y displacement and 5.2% degradation in the output. This is in accordance to the sensitivity equations [9] for the accelerometer, which are given by $y/a = m/k$ and $V_0/a = (2C_0mV_m)/(kg_0(2C_0 + C_P))$ where V_0 is the output voltage, a is the input acceleration, C_0 is the initial capacitance between fingers, m is the effective mass, V_m is the applied bias, C_p is the parasitic capacitance, k is the spring constant, g_0 is the initial interfinger gap, and y is the displacement in the y direction [21]. The extracted schematic captures the actual metal1 and metal2 areas used for signal routing in the *plate*, leading to as smaller mass (m) than in the designed schematic. The parasitic joints (identified by the extractor) in the springs lead to a larger spring constant (k) in the extracted schematic, as compared to the design schematic. The effect of parasitic capacitance on the transducer output can be observed by changing the routing metal layer for the sense finger of the comb drives which were first done using metal2. The layout was regenerated using metal1 as the routing layer causing a decrease in the parasitic capacitance (since metal1 is further away from the top-most metal layer which is grounded) illustrated by the results shown in experiment 4 of Table II. Comparison with the output sense voltage from experiment 5, where the extracted MEMS schematic is simulated with the circuit schematic without parasitic capacitances, proves that the parasitic capacitance for metal1 routing (experiment 4) is negligible.

This example demonstrates how the integrated verification methodology captures the mutual interaction of the mechanical, electromechanical and electronic domains in an integrated device and also shows the usefulness of parasitic extraction.

C. Gyroscope

Fig. 15(a) shows the layout of an elastically gimbaled z axis CMOS–MEMS gyroscope [22]. It consists of an outer resonator which is actuated in the horizontal direction (x direction) using a pair of linear comb drives. Any external rotation in the z direction results in a Coriolis force which couples the horizontal force (in the x direction because of the linear comb drive) to the vertical direction (along the y direction) resulting in a cross-axis motion. The cross-axis motion is proportional to the Coriolis force generated (and hence proportional to the angular velocity) and is detected by the inner accelerometer [similar to the one in Fig. 14(a)] using a pair of differential comb drives. The inner accelerometer also has four sets of differential comb drives for calibration. The corresponding extracted schematic is shown in Fig. 15(b). When the schematic was simulated in the absence of any external rotation, the accelerometer output showed a y direction motion signifying a cross axis coupling between the input motion to the output sense direction. Ideally, such a coupling should be produced only because of Coriolis force and hence in the absence of external rotation there should be no motion in the y direction. The nonideality in this case was primarily because of a dc twist in the central accelerometer because of a dc voltage on the sense fingers of the differential comb drives [as was explained in Fig. 15(a)]. When the accelerometer comb drive topology was modified to a common centroid topology, the cross axis sensitivity reduce by a factor of 10 [as shown in Fig. 15(c)]. The small cross-axis response in this case was because of the asymmetric distribution of parasitic mass in the

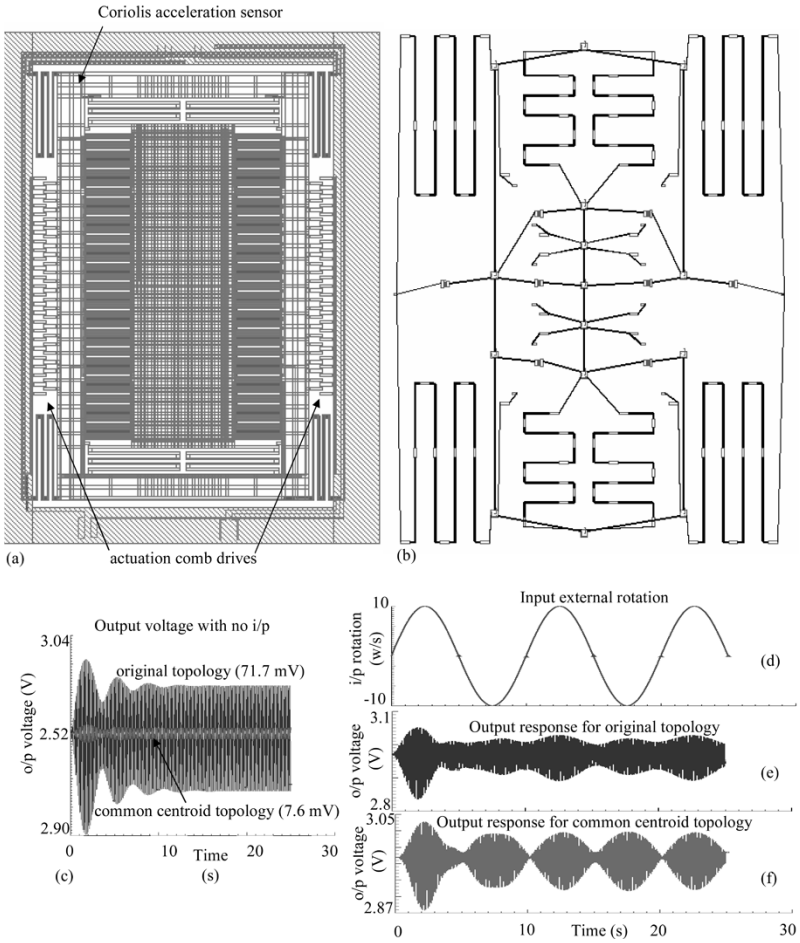

Fig. 15. CMOS–MEMS gyroscope [22] (a) layout, (b) extracted schematic in NODAS, (c) comparison of cross-axis sensitivity for the original topology and when the inner accelerometer is modified to a common centroid topology, (d) input angular rotation to the gyroscope, (e) response of the original gyroscope to the input rotation, and (f) response of the modified gyroscope with common centroid accelerometer to the input rotation.

Coriolis sense accelerometer. The response of the two topologies when a sinusoidal rotation input [Fig. 15(d)] is applied is shown in Fig. 15(e) and (f), respectively. The cross axis of the original topology causes the signal to be distorted beyond detection. This example demonstrates the usefulness of the integrated MEMS extractor in analyzing behavior of complex designs and how symmetry plays an important part in MEMS.

D. Filter

Fig. 16(a) shows an SEM of integrated CMOS–MEMS bandpass filter, with on-chip electrical interfaces [23]. It consists of three crab-leg resonators connected using O-springs. The extracted MEMS schematic is shown in Fig. 16(b). The integrated extracted schematic was simulated and compared with simulation of the design schematic along with the experimental data [23] as shown in Fig. 16(c). Table III shows the details of the simulation results for the three resonant peaks. The extracted schematic captures the effect of additional routing mass in decreasing the resonant frequencies by 2%. The small difference between the results from the extracted schematic and actual measured data demonstrates the extractors capability for accurate representation of the fabricated device at the schematic level.

E. Z Axis Accelerometer

Fig. 17(a) shows the layout of a z axis accelerometer similar to the one in [24]. The central floating plate is suspended at the four corners with the help of long serpentine springs. The serpentine spring uses a mixture of long and short beams [shown in the inset of Fig. 17(a)] to increase the stiffness in the plane of the structure while keeping it compliant in z direction (outward from the paper). Any motion of the suspended plate along the z direction is detected by the four sets of comb drives. Since the layout is quad-symmetric, only a quarter of the layout was extracted and analyzed. The extracted schematic for a quarter of the layout is shown in Fig. 17(b). The joints connected to the short beams are accurately captured as parasitic joints by the extractor. Since this device was not fabricated, there are no experimental results available for verification of the results from extraction. Hence, the simulation results from extraction is compared with results from time consuming rigorous FEM simulations. Also, the resonant frequency is the z direction that has been considered to be the critical specification being compared. The resonant frequency is important because it is used in designing the circuitry for the z axis accelerometer. Fig. 17(c) shows the first resonant mode (along the z direction) of the device obtained from FEM simulation of the quarter of the layout.

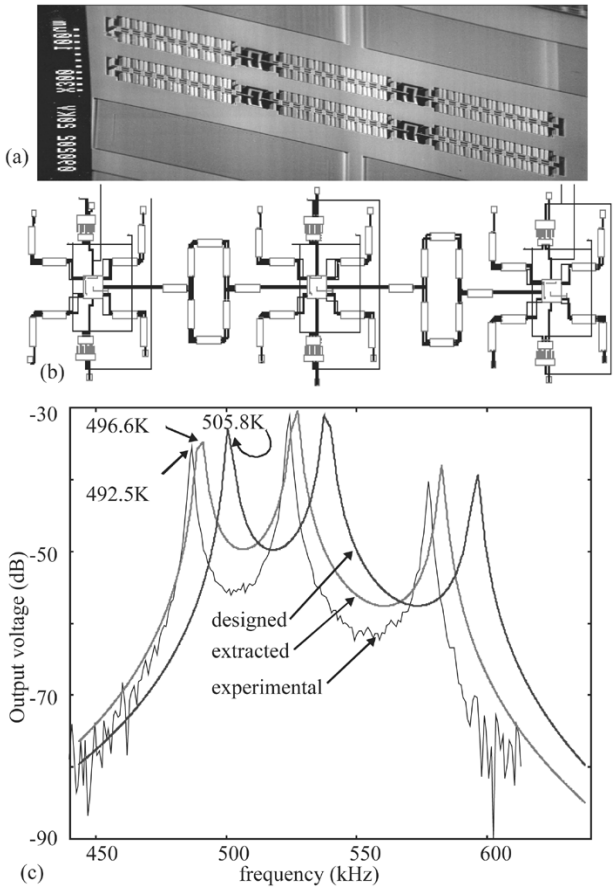

Fig. 16. (a) SEM of CMOS MEMS filter [23], (b) extracted schematic, and (c) comparison of results.

TABLE III
COMPARISON OF DATA FOR THE FIRST RESONANT PEAK IN THE BAND PASS FILTER FOR THE DESIGNED AND EXTRACTED SCHEMATICS (AC I/P 1 V, BIAS ± 20 V)

	first peak frequency (kHz)	second peak frequency (kHz)	third peak frequency (kHz)
experimental	492.5	529	581.2
designed	505.8	542	599.8
extracted	496.6	532.1	586.1

Fig. 17(d) compares the frequency sweep results for the extracted schematic and the designed schematic (without joints) and compares it to the resonant frequency obtained from FEM simulation. The result from extracted schematic comes to 9.4% of the FEM result while the designed schematic has an error of 200%. It is important to note that the schematic-level simulation is at least ten times faster than the numerical modal analysis and shows the response of the device for the whole frequency range in addition to the resonant frequency. Hence, the extractors capability of accurate parasitic recognition and representation at the schematic level allows designers to use fast schematic level simulators without significant reduction in the accuracy of results.

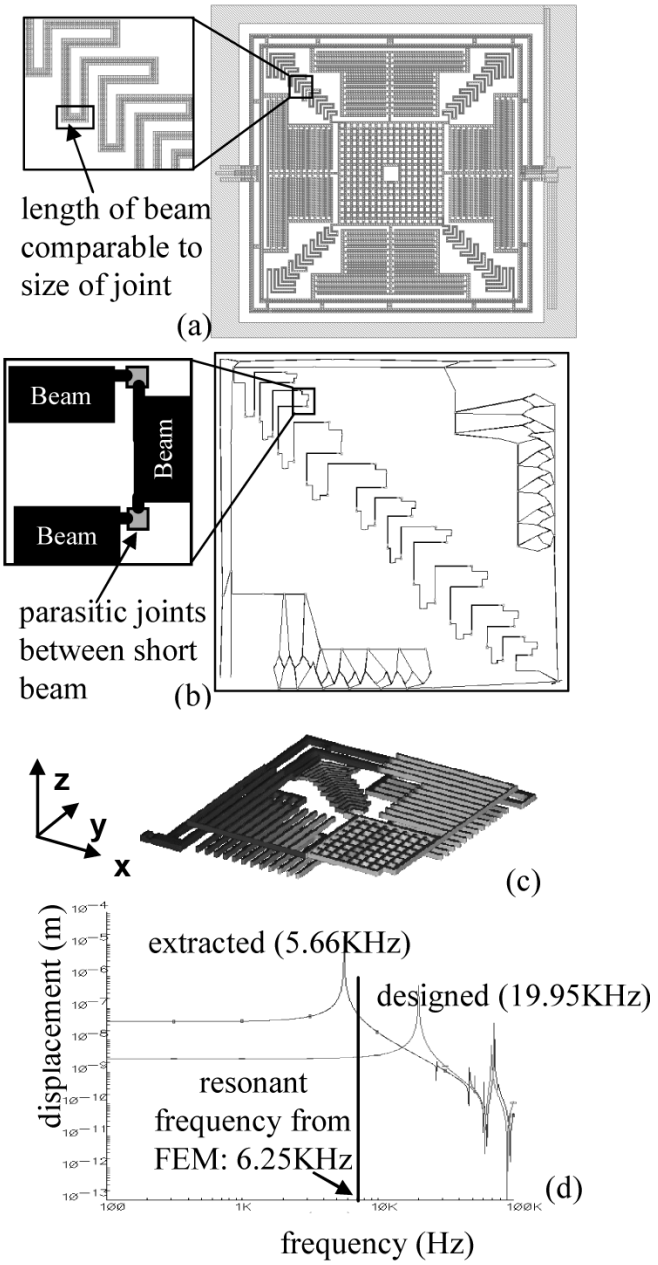

Fig. 17. Z axis accelerometer [24]. (a) Layout. (b) Extracted schematic of a quarter of the design. (c) First resonant mode in FEM simulation of the quarter design. (d) Comparison of the resonant frequencies.

VI. CONCLUSION

A MEMS LVS methodology using an integrated MEMS extractor, for verification of mixed-domain MEMS layouts has been presented in this paper. The integrated MEMS extractor facilitates the use of schematic-level simulation for verification of MEMS by automatically constructing an accurate schematic representation of an integrated MEMS layout along with all domain-specific parasitics. The use of schematic-level simulations allows designers to investigate the cause of any undesired effects in the design while avoiding time consuming numerical analysis and costly fabrication. In addition, the integrated verification methodology permits simultaneous simulation of the electronics and MEMS components of an integrated system,

thereby helping designers understand the interaction between different domains in the system. A custom MEMS LVS tool further enhances the efficiency of the verification flow by detecting common human errors, like parameter and connectivity errors, even before running schematic simulation. Layout symmetry information, which is vital for MEMS design, is also highlighted by the LVS tool. Computationally efficient algorithms for representation and recognition of MEMS *atomic* and *functional elements* from multilayered layout geometry are used to improve the efficiency of the prototype implementation of the extractor. A wide variety of examples are presented to highlight the accuracy and efficiency of the extractor in verifying complex integrated MEMS layouts. Such a verification flow is essential for faster design cycles of complex mixed-domain integrated MEMS.

REFERENCES

[1] D. Teegarden, G. Lorenz, and R. Neul, "How to model and simulate microgyroscope systems," *IEEE Spectrum*, vol. 35, no. 7, pp. 66–75, Jul. 1998.

[2] G. K. Fedder and Q. Jing, "A hierarchical circuit-level design methodology for microelectromechanical systems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 10, pp. 1309–1315, Oct. 1999.

[3] J. V. Clark, D. Bindel, W. Kao, E. Zhu, A. Kuo, N. Zhou, J. Nie, J. Demmel, Z. Bai, S. Govindjee, K. S. J. Pister, M. Gu, and A. Agogino, "Addressing the needs of complex MEMS design," in *Proc. MEMS*, Las Vegas, NV, Jan. 20–24, 2002, pp. 204–209.

[4] A. V. Chavan and K. D. Wise, "A monolithic fully-integrated vacuum-sealed CMOS pressure sensor," in *Proc. IEEE Aerospace Applicat. Conf.*, Mar. 21–28, 1998, pp. 341–346.

[5] E. R. Brown, "RF-MEMS switches for reconfigurable integrated circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 46, no. 11, pp. 1868–1880, Nov. 1998.

[6] K. Wang and C. T.-C. Nguyen, "High-order medium frequency micromechanical electronic filters," *IEEE MEMS*, vol. 8, no. 4, pp. 534–556, 1999.

[7] H. Samuels, "Single- and dual-axis micromachined accelerometers," *Analogue Dialogue*, vol. 30, no. 4, pp. 3–5.

[8] M. Lemkin and B. E. Boser, "A micromachined fully differential lateral accelerometer," in *Proc. IEEE Custom Integr. Circuits Conf.*, 1996, pp. 315–318.

[9] H. Luo, G. Zhang, L. R. Carley, and G. K. Fedder, "A Post-CMOS micromachined lateral accelerometer," *J. MEMS*, vol. 11, pp. 188–195, 2002.

[10] C. M. Baker and C. Terman, "Tools for verifying integrated circuit designs," *Lamda Mag.,* , vol. 1, no. 3, pp. 22–30, 1980.

[11] N. R. Swart, "A design flow for micromachined electromechanical systems," *IEEE Design Test Comput.*, vol. 16, no. 19, pp. 39–47, Oct./Dec. 1999.

[12] M. A. Maher and H. J. Lee, "MEMS systems design and verification tools," in *Proc./SPIE Smart Structures Mat.*, San Diego, CA, Mar. 1998, pp. 40–48.

[13] B. Baidya, S. K. Gupta, and T. Mukherjee, "An extraction based verification methodology for MEMS," *J. MEMS*, vol. 11, pp. 2–11, 2002.

[14] 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 micro-structures in a conventional CMOS process," *Sensors Actuators*, vol. A57, no. 2, pp. 103–110.

[15] G. K. Fedder, "Simulation of microelectromechanical systems," Ph.D. dissertation, Elect. Eng. Comput. Sci. Dept, Univ. Berkeley, 1994.

[16] W. C. Tang, T.-C. H. Nguyen, M. W. Judy, and R. T. Howe, "Electrostatic-comb drive of lateral polysilicon resonators," *Sensors Actuators A (Phys.)*, no. 1–3, pp. 328–331, 1990.

[17] W. C. Tang, M. G. Lim, and R. T. Howe, "Electrostatically balanced comb drive for controlled levitation," in *Tech. Dig. IEEE Solid-State Sensor Actuator Workshop*, Jun. 1990, pp. 23–27.

[18] J. B. Rosenberg, "Geographical data structures compared: A study of data structures supporting region queries," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. CAD-4, no. 1, pp. 53–67, Jan. 1985.

[19] B. Baidya and T. Mukherjee, "Layout verification by extraction for micro total analysis systems," in *Proc. Modeling Simul. Microsyst.*, vol. 1, San Francisco, CA, Feb. 23–27, 2003, pp. 262–265.

[20] Q. Jing, T. Mukherjee, and G. K. Fedder, "Schematic-Based lumped parameterized behavioral modeling for suspended MEMS," in *Tech. Dig. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 10–14, 2002, pp. 367–373.

[21] G. Zhang, "Design and simulation of a CMOS-MEMS accelerometer," M.S. thesis, Elect. Comput. Eng. Dept., Carnegie Mellon Univ., Pittsburgh, PA, 1998.

[22] H. Luo, G. K. Fedder, and L. R. Carley, "An elastically gimbaled Z-axis CMOS-MEMS Gyroscope," in *Proc. Int. Symp. Smart Structures Microsyst.*, Hong Kong, Oct. 19–21, 2000, pp. 1–6.

[23] Q. Jing, H. Luo, T. Mukherjee, L. R. Carley, and G. K. Fedder, "CMOS micromechanical bandpass filter design using a hierarchical MEMS circuit library," in *Proc. MEMS*, Miyazaki, Japan, Jan. 23–27, 2000, pp. 187–192.

[24] H. Lakdawala and G. K. Fedder, "Temperature control of CMOS micromachined sensors," in *Proc. MEMS*, Las Vegas, NV, Jan. 20–24, 2002, pp. 324–7.

Bikram Baidya received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1999 and 2003, respectively.

He is currently with the Computer-Aided Design Division, Intel Corporation, Hillsboro, OR.

Tamal Mukherjee 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.

He is currently an Associate Research Professor in the Department of Electrical and Computer Engineering, Carnegie Mellon University. His research interests include automating the design of analog circuits and microfluidic and microelectromechanical systems. His current work focuses on the developing computer-aided design methodologies and techniques for integrated microelectromechanical systems. He is also involved in modeling, simulation, extraction, and synthesis of mixed-domain microsystems. He is also active in the use of these techniques for the design of RF circuits whose performance is enhanced by micromachining.