Extraction and LVS for Mixed-domain Integrated MEMS Layouts

Bikram Baidya (bbaidya@ece.cmu.edu) and Tamal Mukherjee (tamal@ece.cmu.edu)

Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

Abstract

As design of integrated MicroElectroMechanical Systems (MEMS) matures, there is an increasing need for verification of MEMS layouts. This requires a mixed-domain LVS (layout-versus-schematic) methodology capable of extracting an integrated schematic from the mixed-domain layout and verifying it against the designed schematic. This paper reports on a prototype implementation of MEMS LVS and a MEMS extractor, which, in addition to reconstructing the extracted schematic also captures the domain-specific parasitics in the individual devices. This schematic is then used by a custom schematic-versus-schematic comparator to match connectivity of various elements between the designed and extracted schematics. Finally, simulation of the extracted schematic also helps in capturing the true behavior of the system.

Keywords: MEMS extraction, parasitics, MEMS LVS, verification, integrated MEMS

Introduction

Verification of MEMS layouts integrating components from various disciplines (like electronics, electrical, mechanical, etc.) needs integrated simulation of multi disciplinary components in order to capture the true behavior of the system. An emerging approach to handle this diversity of physical disciplines involves schematic-level simulators capable of handling mixed-domain schematic [1] [2] [3]. This has resulted in increasing use of schematic view as the design entry mode for such systems, followed by manual or automatic generation of layout. Verification of such layouts requires a mixed-domain LVS (layout-versus-schematic) methodology capable of handling devices across different physical disciplines.

Initial attempts towards MEMS LVS have been limited to higher level connectivity analysis [4] [5] using tagged layouts, generated automatically from schematics, to verify the initial connectivity of the design. Tagging of manually generated layout is difficult. Also such methods fail to extract layout parasitics.

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 designed schematic to verify connectivity of elements in the design. The extractor presented here uses the extraction framework introduced in [6] and can handle layout design entry. In addition it identifies domain-specific parasitics in the final layout and hence can be used to verify the true behavior of the integrated system.

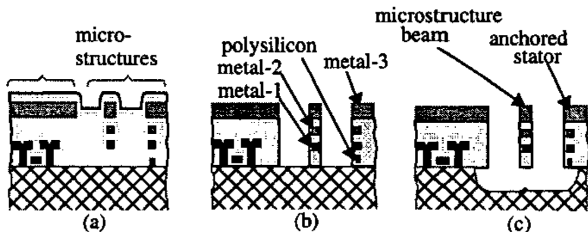

Fig. 1: Cross-section of CMOS micromachining process [7]; (a) after standard CMOS process, (b) after anisotropic etch, (c) on final release using isotropic etch

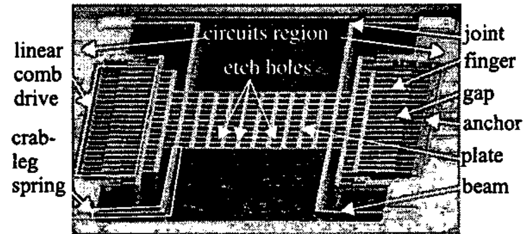

Fig. 2: SEM of a crab-leg resonator showing the atomic elements (on the right) and the functional elements (on the right) and the etch holes for post CMOS release

This paper focuses on the CMOS micromachining process [7] as a representative integrated MEMS process. Two maskless dry etches are used to release the microstructure protected by the top-most metal layer in foundry CMOS designs. Cross-sections of the microstructure during various points in the CMOS post processing are shown in Fig. 1. Fig. 2 shows a SEM of a crab-leg resonator fabricated using this process. Such MEMS components can be hierarchically decomposed into atomic (labels on right in Fig. 2) and functional elements (labels on left in Fig. 2). The atomic elements (Fig. 3) form the building blocks for MEMS components and can be either fixed elements, like the anchor (which adheres the suspended structures to the substrate) or suspended elements like plates, beams, joints (differentiated based on their compliance and connectivity) or non-structural elements like gaps. Atomic elements can be combined together to form functional elements. Fig. 3 shows common functional elements along with some typical examples. The LVS methodology described here exploits this hierarchy while extracting the schematic from the layout geometry.

The sections that follow describe 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.

Integrated MEMS Design Flow

Fig. 4 shows the design flow for integrated MEMS. It starts with the separate designs for MEMS and circuit schematics followed by the simulation of the top-level integrated design that combines these schematics. Next, layouts for MEMS and circuits components are

0-7803-7607-2/02/$17.00 © 2002 IEEE

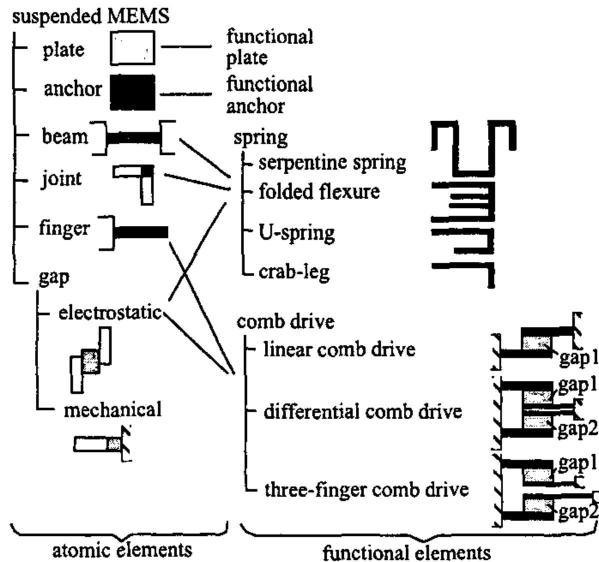

Fig. 3: Atomic and functional elements for suspended MEMS

generated, either manually or automatically, which are then integrated and wired to get the final integrated layout of the system. This layout is then verified using a two step LVS methodology. First the individual schematics along with the domain specific parasitics are extracted. These are then combined, using hints from the integrated layout, to form the integrated extracted schematic. The individual schematics are compared with the corresponding design schematic to check the connectivity of elements, followed by a comparison of the integrated schematics to verify block level connectivity across different components. If the LVS check completes successfully, the integrated extracted schematic is simulated to see if the parasitics in the layout changes system behavior. If the simulation satisfies the design specifications it can be then sent out for fabrication.

Extraction Flow

The most important step in integrated LVS flow is extraction. 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 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, a 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. 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 generate the extracted schematics together with the domain-specific parasitics. The electrical parasitics in the MEMS regions are also extracted

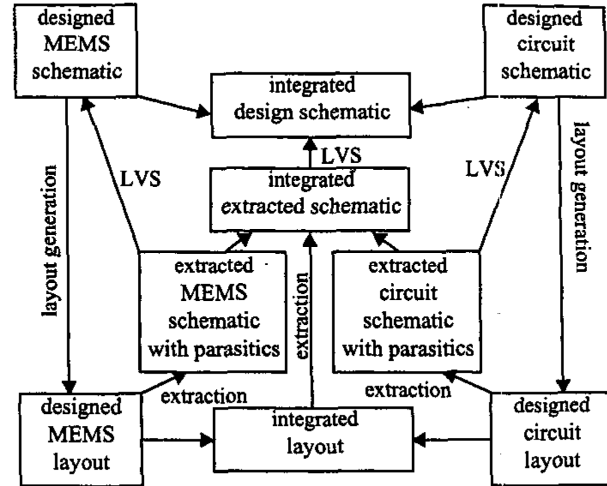

Fig. 4: Design flow for integrated MEMS

using the commercial VLSI extractor. Finally the individual schematics are stitched together by the master extractor to obtain the integrated schematic.

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

Representation

The first step in MEMS extraction is representation of the layout geometry together 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 three metal layers in the MEMS region (for the CMOS process) together with the structural holes in the layout. This 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 [8] representation in VLSI, is used to represent all the layers as shown in Fig. 5. The *MEM-layer* together with the empty

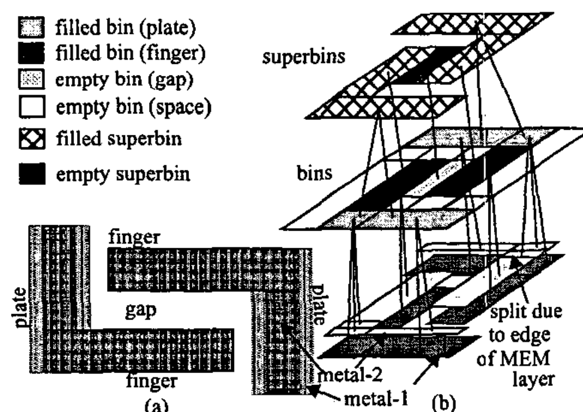

Fig. 5: Hierarchical bin data structure; (a) example layout of two cantilever fingers; (b) its bin storage

regions in the layout form the bin level of the hierarchy where the first iteration of recognition, to find atomic elements, occurs. The connectivity and structural information is also stored at this level using the information from the lowest level in the hierarchy where the structural layers (the three metal layers in the CMOS process) are stored in their canonized form. The geometry in the structural layers are further split by the edges of the *MEM-layer* so that each rectangle in every layer belongs to an unique bin. 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 recognizes the functional elements.

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 [6]. This 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 a 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.* Since the vast majority of suspended MEMS designs are Manhattan in nature, this prototype implementation of extractor, and hence the MEMS LVS, will deal with Manhattan layouts only.

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.

SORT(***edge_set E***):
  Assign direction to edges such that the layout area is to the left.
  Sort the edges in E w.r.t. their abscissa, then their ordinate and finally their angle to abscissa and return the sorted set of edges

OVERLAP (***edge a, edge b***):
  returns *TRUE* if edges a and b overlap

INTERSECTION(***edge_set A, edge_set B***):
  find the 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 and replace edge e in S with its split parts
  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** $length[V_e]$
    edge_set X = SPLIT_AND_PROPAGATE(x, $V_e[i]$, S)
  for $i = 1$ **to** $length[V_x]$
    edge_set E = SPLIT(e, $V_x[i]$, S)
  e = top most edge of E
  return INTERSECTION(X, E)

CANONIZE(***edge_set E***):
  SORT(E)
  scanline S = NULL
  for $i = 1$ **to** $length[E]$
    $push[S,E[i]]$
    edge $c = E[i]$
    edge n = successor of edge c in S
    if OVERLAP(n, c) is ***TRUE***
      edge_set I = MODIFY(n, c, S)
      $S = S\text{-}I$
      if c is a closing edge
        $S = S\text{-}I$
    edge p = predecessor of edge c in S
    if OVERLAP(p, c) is ***TRUE***
      edge_set I = MODIFY(p, c, S)
      $S = S\text{-}I$
      if c is a closing edge
        $S = S\text{-}I$
  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 $O(m + e\log k)$ where e is the initial number of vertical edges.

Fig. 6 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.

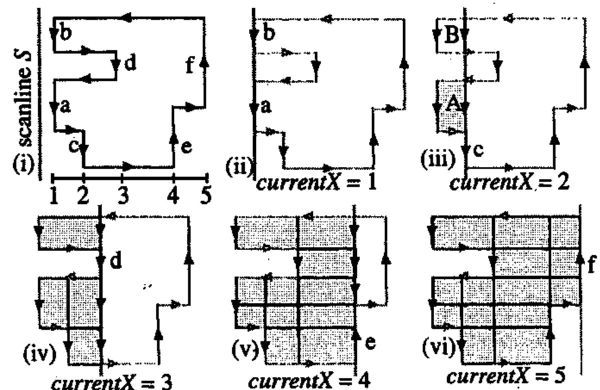

Fig. 6: Example demonstrating the canonization procedure. The contents of scanline S and the state of the canonical set of rectangles are shown in consecutive figures are shown for each value of currentX

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 neighbor information, the rectangles are sorted first with their abscissa and then with their ordinate and for the finding the horizontal neighbor information the sorting order is reversed.

The inter-layer canonization and the binning algorithms follow a similar scanline approach where the edges corresponding to different layers are color coded and the decision at each insertion and deletion uses the color of the edge together with the overlap information.

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. 3).

Recognition of atomic elements involves classifying the bins into different atomic types (plates, beams, fingers, gaps and anchors). The recognition uses 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 philosophy similar to the canonization routine and hence have $O(n\log n)$ time complexity. 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. The user also has the flexibility of choosing the subroutines to use and the order in which they can be called.

The first iteration for *hole* recognition finds *bins* having no structural layer underneath (*non-structural bins*) that are completely surrounded by *bins* having structural layers (*structural bins*). The next iteration compares the combined widths of such *bins* with the widths of the neighboring contiguous sets of *structural bins*. If the ratio is less than an user-defined ratio then they are recognized as *holes*. The rest of the *non-structural bins* are recognized as *gaps*. The next important element is the *electrostatic gap*. If a *gap bin* or a set of *gap bins* have two *structural-bin neighbors* of different electrical

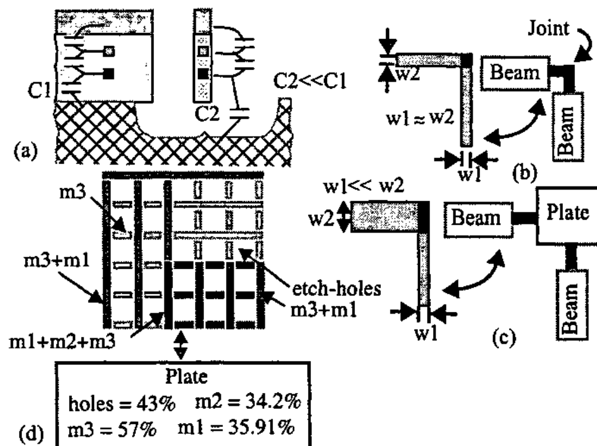

Fig. 7: Electrical and mechanical parasitics: (a) parasitic capacitances, (b) a joint between two nearly equal beams neglected in the model, (c) a joint between two beams of different aspect ratios modeled as plate to account for its parasitic effects, (d) layout of plate area in MEMS and corresponding schematic with parasitic values

connectivity then the set of *gaps* is recognized as an *electrostatic gap*. The structural compliance of the MEMS structure is determined by the *beams*. In the first iteration to recognition of *beams*, *structural bins* having an aspect ratio greater than an user defined ratio are marked as *potential beams*. In the next iteration, contiguous sets of *potential beams* having non-beam physical neighbors only on their shorter sides are marked as *beams*. Similarly, cantilever *beams* (i.e. if there is only one neighbor on one of the shorter sides) are marked as *fingers*. *Fingers* are extensively used to design electrostatic actuators and sensors. *Structural bins* overlapping with *anchor-layer* (a derived layer) are marked as *anchors*. Suspended areas of the geometry which are relatively rigid to forces along the plane of the structure are defined to be *plates* and 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 hence marked as *plates*. *Structural bins* connecting two or more *beams* are recognized to be *joints*.

Recognition of Functional Elements

The next step in element recognition is functional element recognition which occurs after the creation of *superbins* (Fig. 5). 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 an user defined library file which defines the elements for the finite state machine 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 for can be defined by

M = {Q, S, L, G, F, X}, where
Q = states = {S, {intermediate states}, F, X};
S = start state = anchor point;
L = inputs = {{joints}, {beams}, {electrostatic_gaps}, NULL};
G = transition rules;
F = set of final states; and
X = exit state.

Joints are defined by the number and the relative orientations of the beams attached to it [6]. Beams 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 and combs in the library. For example, a U-spring requires three beams (Fig. 3) 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. 3.

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.

Parasitics

Extraction of domain-specific parasitics forms a crucial part of MEMS extraction and will be described in this section. Such parasitics are finally annotated into the extracted schematic.

Electrical parasitics, like parasitic capacitances, significantly affect the operation of electromechanical elements like *comb drives*. Similarly, electrical parasitic capacitances (Fig. 7(a)) in MEMS areas tend to affect the interface between MEMS and circuits. These are extracted using commercially available VLSI extractor after incorporating the modifications due to the difference in geometry (like gap between structure and substrate) in MEMS areas.

Mechanical parasitics include parasitic mass in the *plates* and parasitic *joints* between *beams*. Parasitic *joints* (Fig. 7(b), (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 extractor identifies the parasitic *joints* by comparing adjacent *beams*. 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. 7(d)) results due to routing in lower metal layers (in a multilayer process like the CMOS-MEMS [7] process where the top layer is used to define the suspended structure and the lower layers are used for electrical connectivity) and also from the etch-holes in the *plate*. Such information are annotated as parameters to the *plate* models extracted by the extractor, as shown in the example in Fig. 7(d) which relates to the three metal CMOS-MEMS process.

Layout versus Schematic (LVS)

The final step in the LVS process is a comparison between the designed and the extracted schematic. As shown in Fig. 4, this occurs in three parallel processes. LVS for the integrated design compares the connectivity between the circuit and the MEMS components by comparing the pin to pin connectivity of the two schematics. LVS in electrical circuits relies on commercial VLSI tools while MEMS LVS is custom made.

In VLSI circuits, the nets of concern during LVS start from the power line and end with the ground. In MEMS circuits, the electrical equivalent of the paths between two anchors obey the same Kirchoffian laws as the nets in VLSI circuits. Hence in MEMS LVS the paths between anchors need to be checked and matched.

LVS for MEMS circuits starts with the enumeration of all the sets of paths, each set using a different anchor as the starting point for the paths, for both the designed and the extracted schematic. While enumerating these paths, the relative change in direction, when traversing between two circuit elements, is also taken into account. This is important because, unlike in VLSI circuits, where the relative position between circuit elements is not important, the behavior of a MEMS component can vary significantly for any change in the relative distribution of the MEMS circuit elements. 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. The final path lists for any starting anchor are then connected together to form a tree having nodes corresponding to the MEMS circuit elements and branches containing information about the relative change in direction when traversing between the elements corresponding to the nodes which the branch connects. Tree matching algorithms are then used to match the different trees and, for every such pair of trees from designed and extracted schematic, a match figure corresponding to the amount of match is assigned. Two trees are said to be completely matched if their branches and nodes match completely and hence will have a 100% match. Two branches match if the directed angles corresponding to the branches match. Two nodes are said to match if the element type and its parameters match. The parameters are checked for individual elements is user defined to modify the strictness of the LVS tool. For the results discussed later, the length and width associated with the elements are used as the matching parameters. If a complete match is found between two trees, the LVS reports a success. If such a match is not found, an unsuccessful LVS is reported and the best possible match is highlighted.

Results

This section gives some examples highlighting the usefulness of the LVS and the extraction methodology presented in this paper.

Crab-leg Accelerometer

Fig. 8 shows an example of a crab-leg accelerometer which was verified using the LVS methodology described in this paper. Fig. 8(a) shows the designed schematic for the accelerometer from which the layout was automatically generated. The layout (Fig. 8(b)) was rotated by 90°, for placement in the full system and routing for the comb drives was added using lower metal layers in the plate region. The extracted schematic is shown in Fig. 8(c). One of the beams (shown as erroneous beam in Fig. 8(b)) was accidentally changed in width. When MEMS LVS was used to check the two schematics, the LVS failed and reported the path from the anchor attached to the erroneous beam to be faulty. Correcting the width of the beam

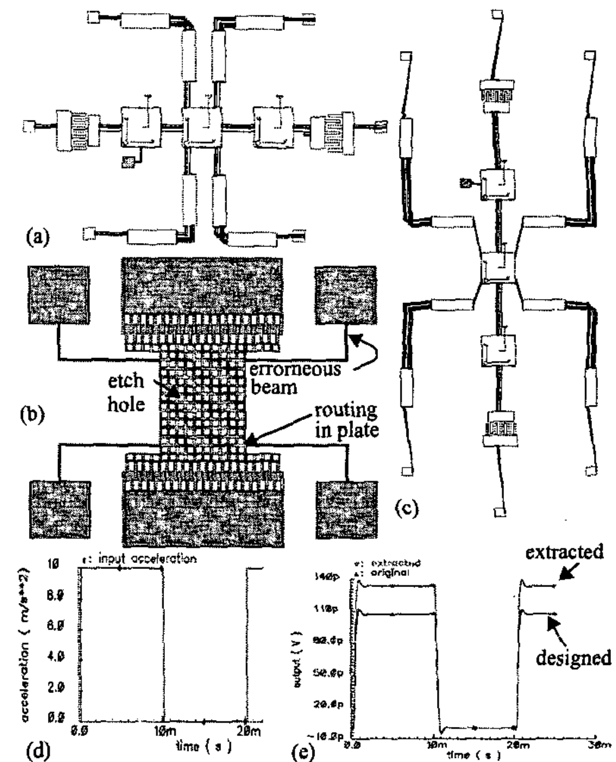

Fig. 8: Crab-leg accelerometer. (a) designed schematic, (b) layout rotated by 90°, (c) extracted schematic, (d) input to the two schematics, (e) comparison of outputs from designed and extracted schematics

resulted in a successful LVS even though the orientations of the two schematics were not the same. However, when the two schematics were simulated with an acceleration input as shown in Fig. 8(d), the sensitivity of the extracted schematic and hence the actual layout was found to be better than that predicted by the design schematic (Fig. 8(e)). This resulted from the additional mass contributed by the routing metal wires in the final layout. This example demonstrates the usefulness of the LVS methodology in catching layout errors and also proves the importance of mechanical parasitics in predicting the true behavior of the final design.

Integrated CMOS-MEMS Accelerometer

An example of an integrated MEMS device is shown in Fig. 9(a). It consists of a CMOS-micromachined accelerometer [9] with on-chip electronics. In addition to extracting the parasitic joints and the parasitic routing mass in the plates, the extracted schematic (Fig 9(b)) also captures the electrical parasitics. Comparison between the results of designed and extracted schematics show a 7.2% degradation in the output (Fig 9(c)). In addition there is a shift in the D.C. operating point. Fig 9(d) shows the cross axis sensitivity of the two schematics. The cross axis sensitivity is greater for the extracted schematic because of the effect of the sense comb bias voltage on the operation of the comb drives.

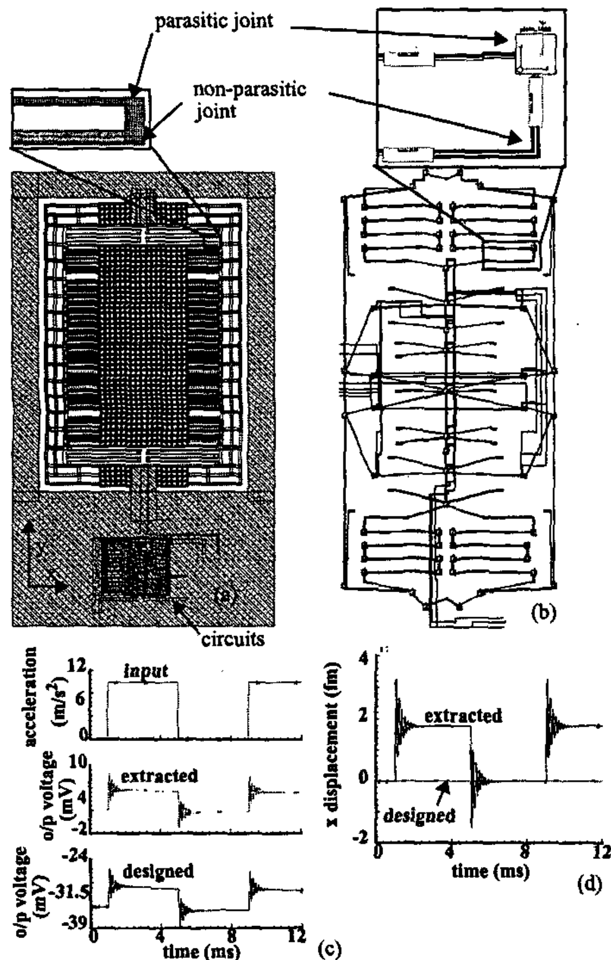

Fig. 9: CMOS accelerometer: (a) layout, (b) extracted schematic, (c) output voltage for 1g input pulse in the y direction, (d) sensitivity in x direction (Simulations done using NODAS [2])

Conclusion

A MEMS LVS methodology, using an integrated mixed-domain extractor is introduced. Computationally efficient algorithms for representation and recognition of MEMS elements from the layout geometry have been presented. Results highlighting the usefulness of the LVS methodology to correct human errors in the final layout and demonstrating the parasitic effects in mixed-domain layouts, which are captured accurately by the mixed-domain extractor, are shown. The LVS methodology presented here will help reduce the verification effort in mixed-domain MEMS designs.

Acknowledgements

This research effort is sponsored by the Defence Advanced Research Projects Agency (DARPA) and U. S. Air Force Research Laboratory, under agreement number F30602-99-2-0545 and F30602-01-2-0987 and in part by the National Science Foundation award CCR-9901171

References

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

[2] G.K. Fedder and Q. Jing, "A Hierarchical Circuit-Level Design Methodology for Microelectromechanical Systems," *IEEE Transactions on Circuits and Systems II (TCAS)*, vol.46, no.10, Oct. 1999, pp. 1309-1315.

[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, A. Agogino, "Addressing the needs of complex MEMS design," *Proceedings of MEMS '02*, Piscataway, NJ, USA, 2002, pp. 204 -9.

[4] N.R. Swart, "A Design Flow for Micromachined Electromechanical systems," *IEEE Design and Test of Computers*, vol. 16, No. 19, pp. 39-47.

[5] M.A. Maher and H.J. Lee, "MEMS Systems Design and Verification Tools," *Proceedings of SPIE Smart Structures and Materials*, San Diego, CA, March 1998, pp. 40-48.

[6] B. Baidya, S.K. Gupta, T. Mukherjee, "An Extraction based Verification Methodology for MEMS," *Journal of Microelectromechanical Systems*, vol. 11, no. 1, Feb. 2002, pp. 2-11.

[7] 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 and Actuators*, vol. A57, no. 2, pp. 103-110.

[8] J.B. Rosenberg, "Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries," *IEEE Transactions on Computer-Aided Design*, vol. cad-4, no. 1, Jan. 1985, pp. 53-67.

[9] H. Luo, G. K. Fedder, and L. R. Carley, "A 1mG Lateral CMOS-MEMS Accelerometer," *Proceedings of MEMS '00*, Miyazaki, Japan, Jan. 23-27, 2000, pp. 502-7.