# Challenges in CMOS-MEMS Extraction

Bikram Baidya (bbaidya@ece.cmu.edu) and Tamal Mukherjee (tamal@ece.cmu.edu)
Carnegie Mellon University, Pittsburgh, PA 15213, USA

## ABSTRACT

CMOS micromachining processes are being increasingly used to fabricate integrated MEMS devices. Verification of such designs requires extraction from layout to mixed domain circuits and MEMS schematics for lumped parameter simulation. The higher etch hole density and multilayer interconnect in CMOS-MEMS designs results in a larger and more complex problem than in polysilicon MEMS. In addition, integrated MEMS verification needs accurate extraction of mechanical and electrical parasitics. This paper reports an extractor with improved *scanline-based* algorithms to meet the larger problem size associated with integrated CMOS-MEMS layouts. In addition, a *hierarchical bin* representation is used to store the multilayer electrical connectivity information. A new *graph-based* algorithm, which reads in an user customizable library file, is used to recognize the wide range of comb drive and spring designs resulting from the flexibility in connectivity. The utility of the extractor is demonstrated using selected designs.

**Keywords:** CMOS micromachining, integrated MEMS, extraction, parasitics, canonical representation, layout verification

## INTRODUCTION

The ability of CMOS-MEMS [1] processes to integrate MEMS sensing and actuation with electronics is drawing increasing interest in the development of custom physical interfaces to the digital IC. A representative CMOS micromachining process is shown in Figure 1. The standard CMOS process is followed by two maskless dry etches to release the microstructures protected by the top-most metal layer.

Figure 2 shows a SEM of a crab-leg resonator fabricated using this process. Such integrated MEMS designs can be divided into MEMS and circuits components. The MEMS components can be hierarchically decomposed into *atomic* and *functional elements* [2]. The functional elements of the resonator (labels on left) can be decomposed into their atomic elements (labels on right). The extraction process exploits this hierarchy by first detecting the atomic elements [3] and then recognizing functional elements [4]. The ability to embed different electrical connections within the structural geometry and the higher etch hole density required for post-CMOS release results in a larger problem size which makes the $O(n^2)$ algorithms in [3][4] prohibitively slow. This paper reports on faster algorithms to handle such designs.

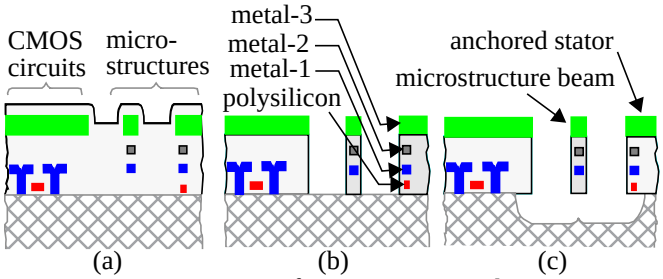


Figure 1: Cross-section of CMOS micromachining process [1]; (a) after standard CMOS process, (b) after anisotropic etch, (c) on final release using isotropic etch

## EXTRACTION FLOW

The CMOS-MEMS extractor consists of a top-level master-extractor which uses two separate extractor modules to extract the MEMS and the electronics. Design rules for post-CMOS etching are used to determine the areas of the layout where the underlying silicon will be etched away to result in suspended MEMS areas. The on chip circuitry and the parasitics of the entire layout is extracted using commercial VLSI extraction tools. The information of etched silicon substrate is accounted accurately while extracting parasitic capacitances in the MEMS areas. The MEMS schematic is extracted from the layout using the MEMS-extractor. Finally, all the sub-schematics are 'stitched' together by the master-extractor to create the integrated schematic which can be simulated to verify the performance of the integrated microsystem. The overall extraction flow is shown in Figure 3.

MEMS extraction starts by the creation of the derived layers (*anchor* layer, *hole* layer, *gap* layer and the *structural* layer), using the layer definitions from the user. The *structural* layer is obtained by a geometric *or* of the topmost metal layer which defines the MEMS structure. The holes in the *structural* layer increases its complexity. A secondary layer (referred to as *MEM* layer) is derived from the *or* of the *structural* layer

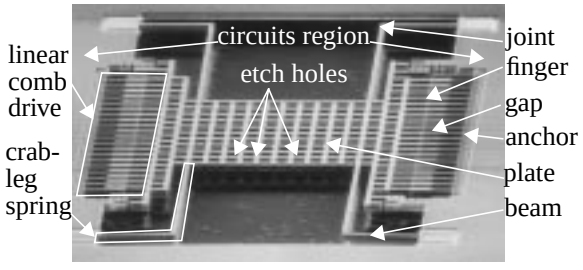


Figure 2: SEM of a crab-leg resonator showing the atomic elements (on the right) and the functional elements (on the right) and the etch holes for post CMOS release

and the *hole* layer, thereby reducing the number of elements needed to represent the layer. The extra area added thereby is annotated to the corresponding rectangle and is accurately reported in the final schematic. The *MEM* layer acts as the top level layer on which all the heuristics are applied to recognize the MEMS components. All the layers are stored in the canonical format explained in [3]. The canonical representations of the *MEM* layer and the gap layer are used to store the connectivity and structural information and are referred as bins in the hierarchical data structure used by the extractor (Figure 4). The structural layers are further split by the edges of the *MEM* layer so that each rectangle in the layer has an unique bin. This is followed by the atomic level recognition in which the filled bins (obtained from the *MEM* layer) are tagged with the atomic element type. The valid gaps are also recognized amongst the empty bins (obtained from the *gap* layer). Recognized bins of the same type are merged together to form the topmost storage level (referred to as superbins). Such a representation simplifies the heuristics and algorithms used for functional level recognition.

## 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, the *canonical representation*, which uses *minimum number of rectangles to cover the layout area such that each rectangle has at most one neighbor per edge and each edge is either fully covered by a neighbor or not covered at all*. The prototype implementation described here deal with Manhattan layouts only and can be extended to non-Manhattan geometry by replacing the rectangles by polygons.

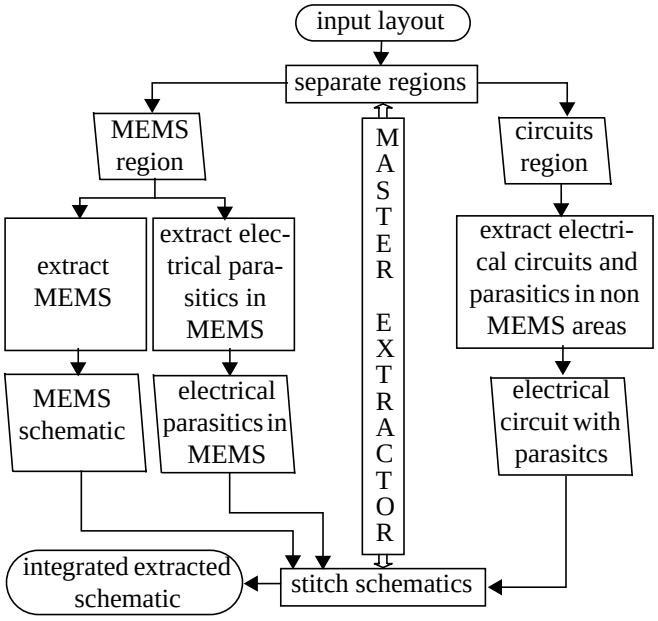


Figure 3: Overall extraction flow

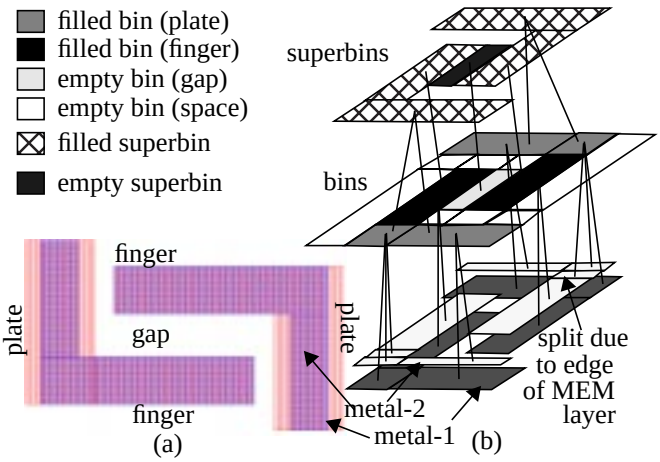


Figure 4: Hierarchical bin data structure; (a) example layout of two cantilever fingers made of metal-1 and metal-2; (b) its bin storage

Unlike the incremental algorithm used in [3], the new canonization routine uses scanline algorithm which relies on the sorting of outer edges of the input geometry. Our algorithm works in two phases. The initial *scan phase* creates the canonical set of rectangles by sweeping a vertical scanline in the horizontal direction across the layout geometry. Only vertical edges of the layout geometry are considered, thereby reducing the problem size by half. The horizontal sweep produces the canonized representation and also the horizontal neighbor information of its rectangles. The final *neighboring phase* is then used to create the vertical neighbor information.

The data structures used by the *scan phase* of the algorithm are a variable *currentX*, which stores the current position of the scanline, and the list of edges *S* in the scanline. The flow of the algorithm is shown in Figure 5. The direction of the edges is chosen such that while traversing along the edge in the given direction, the filled area of the geometry always lies to the left. The vertical edges are sorted first by their abscissa (x-coordinate), then by their ordinate (y-coordinate) and finally by their angle with the abscissa ($90^o$ or $270^o$ from x-axis). Each edge is then sequentially inserted into the scan-

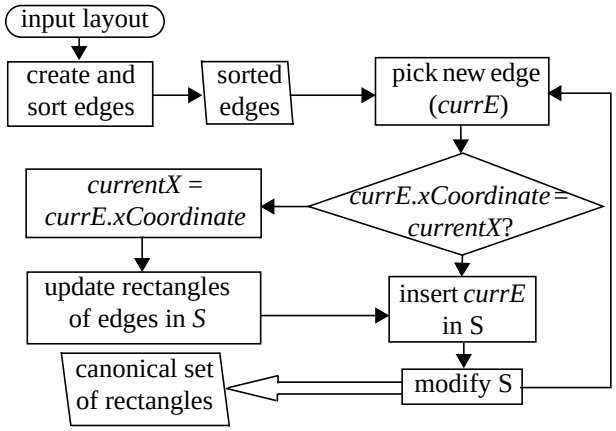


Figure 5: Flow diagram for the scanline-based canonization algorithm

line and begins a new rectangle to its left. As the scanline moves, it drags the vertical edges in it, thereby extending the rectangles associated with those edges. When a new edge is inserted, it completes the rectangles associated with all the contiguous edges it can reach on the scanline, and starts new rectangles for those edges. If the end points of the new edge lie between any of these connected edges, the connected edge is split at that point. This split is propagated to the neighboring boxes on the left. If the scanline is maintained as a balanced binary tree, the insertion of each edge takes $O(\log m)$ time, where $m$ is the current number of elements in $S$; and the connected edges can be found from the *predecessors* and *successors* of the inserted edge. The total number of such comparisons is equal to the number of rectangles in the final canonized representation ($n$), and the total time in insertion is $O(e \log m)$, where $e$ is the initial number of vertical edges and $m$ has an expected value of $O(\sqrt{n})$. If the edge inserted has a positive direction (angle = $90^o$), then the edges of $S$ overlapping with it are deleted from the scanline.

The algorithm is explained with the help of an example in Figure 6. Notice that when edge $c$ is inserted (at $currentX = 2$), 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 edge $e$. Note that the edge $e$ is not split because

The *neighboring phase* sorts the canonized rectangles, first with their x-coordinate and then with their y-coordinate. It then checks whether the top edge of each rectangle coincides with the bottom edge of the next rectangle. If they coincide then they form a vertical neighbor couple and the information is entered.

The inter-layer canonization and the binning algorithms follow a similar scanline approach where the edges of different layers are colored differently and the decision at every step uses the color of the edge along with its direction.

## ELEMENT RECOGNITION

Element recognition occurs in two steps. The first step recognizes the atomic elements which are then used to recognize the functional elements.

Atomic recognition involves classifying the bins into different atomic types (plates, beams, fingers, gaps and anchors as shown in Figure 2) and starts with the recognition of holes. Rectangular empty spaces in the geometry having a width less than an user-defined factor of the widths of its neighboring solid areas are recognized as holes. Fingers and beams are detected using the neighbor information obtained from canonization. Bins which singly or collectively have a length/width ratio above an user-defined value and have neighbors only on opposite edges are detected as beams. Similarly, fingers are bins which satisfy the ratio requirement and have only one neighbor at one of its smaller sides. Plates are recognized from overlap with etch holes. Any region not released is

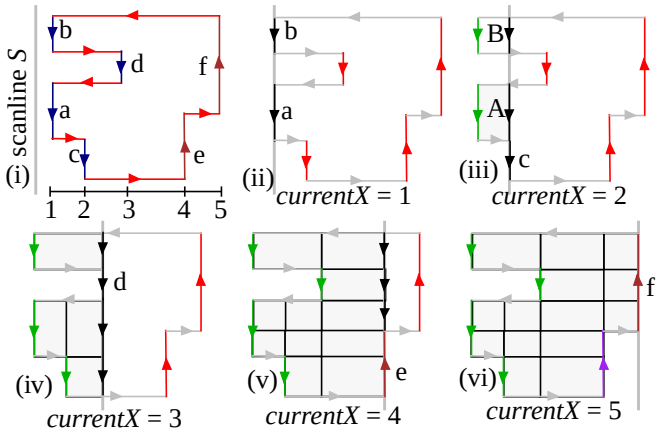


Figure 6: Example demonstrating the canonization procedure. The contents of scanline $S$ and the state of the canonical set of rectangles are shown in consecutive figures are shown for each value of *currentX*

marked as anchor. Empty bins having two of their opposite edges completely covered by fingers are recognized as gaps.

Functional elements include functional plates, comb drives and springs. Two or more contiguous atomic plates which can be combined together form a functional plate and are easily obtained from the superbins. Various permutations of the electrical information along the depth and width of comb fingers can produce a large variety of comb drives having identical mechanical structure. Similarly, unlike the polysilicon process in [4], CMOS designs can have self actuated springs. Hence the electrical information plays an important role in the recognition of comb drives and springs. New graph based algorithms capable of handling such flexibility are used by the extractor to recognize comb drives and springs. The extractor reads in the state transition information from an user-defined library file to create a finite state machine through which contiguous sets of beams, joints, gaps and fingers are passed as inputs to recognize the spring or comb drive. The various springs and comb drives recognized by the current library include serpentine spring, self-actuated serpentine spring, crab-leg, U-spring, folded flexure, O-spring, linear comb drive, differential comb drive and the three-finger comb drive. The list can be extended easily by adding new definitions to the library file.

## RESULTS

### Accelerometer

Figure 7 demonstrates the capability of the extractor using a lateral CMOS-MEMS accelerometer [6]. The complex connectivity for differential comb fingers and the etch holes in plate are shown in Figure 7(a) and Figure 7(b) respectively. 50,000 rectangles are needed to represent its layout (compared to 500 for a purely structural representation as in polysilicon

processes). The new extraction algorithms take 100x less time when compared to the old $O(n^2)$ algorithms. The transient response for the designed and extracted schematics (Figure 7(e) show a 10.5% degradation for the extracted schematic. This demonstrates the capability of the extractor to accurately capture the effects of parasitics in the layout.

## Gyroscope

The capability of the new algorithms to handle large designs is demonstrated by the gyroscope example [7] shown in Figure 8 which requires $10^5$ rectangles for geometric representation. The time required by the new extractor improved by a factor of 400.

## CONCLUSION

A new integrated-MEMS extractor capable of handling complex CMOS-MEMS designs has been presented. The scanline-based algorithms used to obtain a canonical representation and for binning have been explained in detail. The capability of the extractor to handle large designs and to accurately capture the parasitic effects of an integrated-MEMS device has also been demonstrated with the help of an example. Though the paper discusses the extractor using a CMOS-MEMS process, the implementation of the extractor is process independent and can be easily modified for any MEMS process. The availability of such a fast extractor will allow CMOS-MEMS designers to make use of the fast lumped parameter circuit simulators for verification of their integrated MEMS layouts.

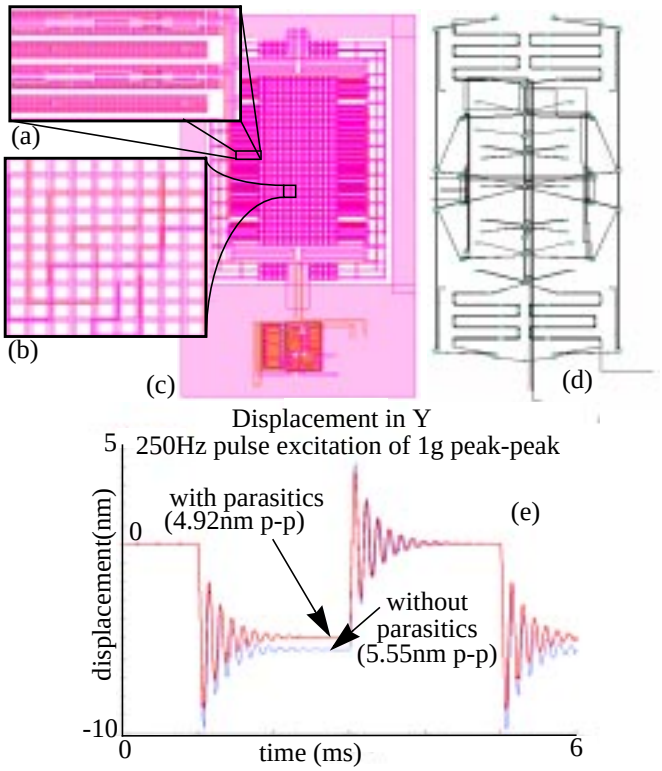


Figure 7: A lateral CMOS-MEMS accelerometer; (a) complex connectivity in differential fingers, (b) perforated plate mass, (c) layout, (d) extracted schematic (e) transient simulation in NODAS

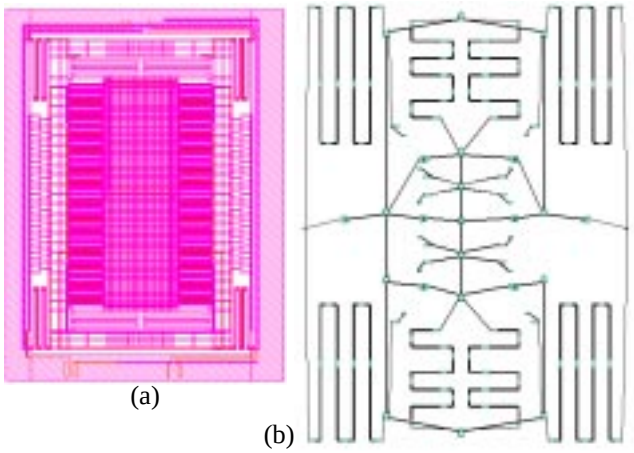


Figure 8: A CMOS-MEMS gyroscope (a) layout, (b) extracted schematic in NODAS

## ACKNOWLEDGEMENTS

This research effort is sponsored by the Defence Advanced Research Projects Agency (DARPA) and U. S. Air Force Research Laboratory, under agreement number F30602-97-2-0323 and F30602-99-2-0545 and in part by the National Science Foundation award CCR-9901171.

## REFERENCES

[1] G.K. Fedder et al, “Laminated high-aspect-ratio microstructures in a conventional CMOS process,” *Sensors and Actuators*, v. A57, no. 2, pp. 103-110.

[2] G.K. Fedder, Qi Jing, “A Hierarchical Circuit-Level Design Methodology for Microelectromechanical Systems,” *TCAS II*, v. 46, no. 10, Oct. 1999, pp. 1309-1315.

[3] B. Baidya et al, “Feature-Recognition for MEMS Extraction,” *CDROM Proc. 1998 ASME DETC*, Atlanta, GA, Sept. 13-16, 1998.

[4] B. Baidya et al, “MEMS Component Extraction,” *Proc. MSM ‘99*, San Juan, Puerto Rico, Apr. 19-21, 1999,pp. 143-6.

[5] T.G. Szymanski, C.J.V. Wyk, “Space Efficient Algorithms for VLSI Artwork Analysis,” *ACM IEEE 20th DAC Proc.*, Miami Beach, FL, USA, 27-29 June 1983, pp. 734-9.

[6] H. Luo et al, “A 1mG Lateral CMOS-MEMS Accelerometer,” *Proc. MEMS 2000*, Miyazaki, Japan, Jan. 23-27, 2000, pp. 502-7.

[7] H. Luo et al, “An Elastically Gimbaled Z-Axis CMOS-MEMS Gyroscope,” *CDROM proc. International Symposium on Smart Structures and Microsystems 2000*, Hong Kong, China, Oct. 19-21, 2000.