Design Synthesis of Microelectromechanical Systems Using Genetic Algorithms with Component-Based Genotype Representation

Ying Zhang
Systems Engineering Program
University of California
Berkeley, CA 94720, USA
yzh@berkeley.edu

Raffi Kamalian
Faculty of Design
Kyushu University
Fukuoka 815-8540, JAPAN
raffi@design.kyushu-u.ac.jp

Alice M. Agogino
Department of Mechanical Engineering
University of California
Berkeley, CA 94720, USA
agogino@berkeley.edu

Carlo H. Séquin
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, USA
sequin@cs.berkeley.edu

ABSTRACT

An automated design synthesis system based on a multi-objective genetic algorithm (MOGA) has been developed for the optimization of surface-micromachined MEMS devices. A hierarchical component-based genotype representation is used, which incorporates specific engineering knowledge into the design and optimization process. Each MEMS component is represented by a gene with its own parameters defining its geometry and the way it can be modified from one generation to the next. The object-oriented genotype structures efficiently describe the hierarchical nature typical of engineering designs. They also prevent MOGA from wasting time exploring inappropriate regions of the search space. The automated MEMS design synthesis is demonstrated with surface-micromachined resonator and accelerometer designs. (Track Category: EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION)

Categories and Subject Descriptors

J.6 [**Computer-aided Engineering**]: Computer-aided design (CAD).

General Terms: Algorithms, Performance, Design

Keywords: MEMS Design Synthesis; MOGA; Component-Based Genotype Representation; Evolutionary Multiobjective Optimization.

1. INTRODUCTION

Microelectromechanical systems (MEMS) are miniaturized mechanical devices and components fabricated with processes similar to those used for integrated circuits (ICs). MEMS have diverse applications in automotive engineering, bioengineering, telecommunications, environmental monitoring, and space exploration. With the rapid growth of MEMS technology and the increase in design complexity, an efficient computer-aided design tool for MEMS design is in great demand to automatically generate promising structures and optimized solutions for new applications and constraints.

Zhou, et al. [1][2] were the first to demonstrate MEMS component synthesis using a multi-objective genetic algorithm (MOGA) on a simple MEMS device, a 'meandering resonator.' Kamalian, et al. [3][4] extended Zhou's work to more challenging MEMS synthesis problems, and they explored the role of geometric constraints and human interaction in MEMS resonator synthesis. The work of Zhou and Kamalian used a fixed data structure of strings and numbers to represent the genotype. Although this research demonstrated a proof of concept for the approach, it did not provide the flexibility and efficiency needed for a general automated computer-aided design tool for MEMS.

In this paper, an extensible, hierarchical, component-based genotype representation is used in the MOGA process, and we have attached an object-oriented component library as a source of practical and efficient genotypes for MEMS design. These components implicitly encapsulate domain-specific engineering experience and thus help focus MOGA into the more promising areas of the solution space. As the literature and our experience in this domain grow, new designs and components will be added to the library, promising even more effective prototyping in the future. To prevent losing good intermediate solutions encountered during the evolutionary search, we have also added an archiving procedure that ranks all designs on a persistent scale and preserves the most promising ones.

To test the new MOGA process, we compare our results to the microresonator design used by Kamalian [3] and also apply it to a commercial micro-accelerometer chip. Both examples show that incorporating engineering domain knowledge via a component-based genotype representation can dramatically improve the effectiveness of a MOGA design synthesis process.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007…$5.00.

2. NEW MOGA SYNTHESIS PROCESS

Genetic algorithms (GAs) are stochastic search techniques used to find approximate solutions to combinatorial optimization problems. They were introduced by John Holland [5] to explain the adaptive processes of evolving natural systems and for creating new artificial systems in a similar way. Goldberg [6] demonstrated how to use GAs in search, optimization, and machine learning. To be successful for engineering design challenges, GAs require a parametric encoding of the evolving phenotypes that is efficient yet flexible enough to describe new and creative solutions. It also requires an efficient way to evaluate the quality, or "fitness", of the various phenotypes, so that the most promising ones have a higher probability of being evolved further with suitable genetic operations, such as selection, mutation, and crossover.

The main challenge is to find a way to properly encode the design parameter set so that changes in the associated genotype are operationally meaningful and flexible enough to have a chance of finding good designs.

2.1 MEMS Design Representation

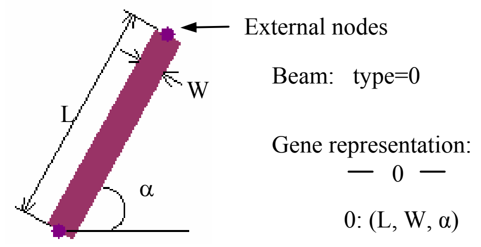
A commonly used genotype representation in GAs uses a fixed length bit string. For complex engineering design problem, however, such as MEMS synthesis, a more sophisticated representation is needed. Zhou, et al [2] used structured real value strings of variable length to represent the genotype for a MEMS design. In what follows, we describe a flexible and computationally tractable framework for a general automated computer-aided design tool for MEMS.

We introduce a tree-structured component-based genotype, implemented with an object-oriented data structure developed by Graf [7]. The prototype MEMS design is represented as a hierarchical assembly of components, ranging from simple primitives, to cluster blocks, and whole operational resonator designs. Typical MEMS primitives are anchors, beams, and mass plates. Cluster blocks may combine many such primitives into poly-line springs, I-shaped resonator masses, or various folded flexure frames. Higher-order parameterized primitives include serpentine suspensions, comb drives, or differential comb drives. At the highest level in the component library we find complete resonator examples.

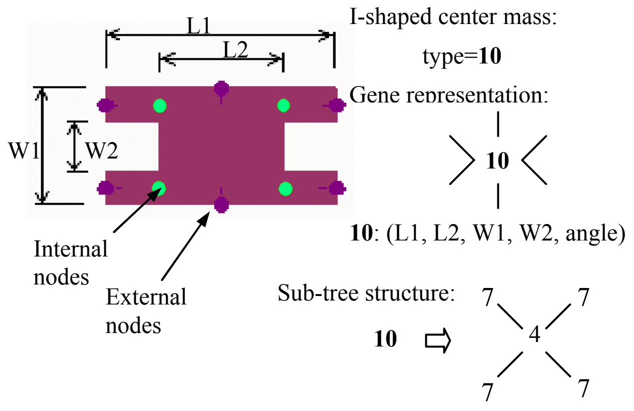
These components are assembled in a global 2D layout coordinate system. Each component typically carries an "angle" parameter that specifies its orientation in the layout plane; while its translational position is derived from the specified connectivity between components via a coincidence constraint of the connection points involved and the relative distance from the point randomly selected as the origin.

In our modular multi-objective GA (MOGA) framework, each MEMS design component type is assigned an exclusive type number and is represented by a gene of its own. This gene carries all salient information about this component: its geometric layout parameters, as well as constraints on how it can be modified and what genetic operations can be applied to it. The actual data fields can be real numbers, integers, strings, or binary flags.

Figure 1 shows examples of primitive and cluster components and their representative genes. Figure 1(a) is a primitive MEMS design component, a beam. We define this gene to be of type 0, and its geometrical parameters are length, width, and its positional angle in the global coordinate frame. Figure 1(b) is a cluster building block, an I-shaped center mass. It is assigned gene type **10** (bold numbers are used to represent cluster genes), which includes four "beams" (gene type = 7) and one rectangular mass plate (gene type = 4). The beams used in this cluster have fewer parameters than the beam shown in Figure 1(a). They have no individual angle variables, since they are all aligned with one another. The geometrical parameters for this cluster are shown on the figure, except for its overall orientation angle. Internal and external nodes are the points of the components through which the building blocks are connected and registered to one another. They implicitly define the placement in the layout coordinate system. A cluster gene has sub-tree structures to represent the internal hierarchy. Even though the genes in the sub-tree have their own geometrical parameters, many of them are coupled to parameters in the parent gene to permit an overall parameterized design description. The node-based hierarchical representation also makes the MEMS design components compatible with our simulation engine – SUGAR [8].



(a) primitive gene type example

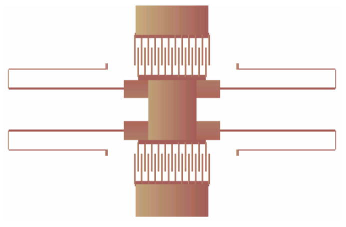


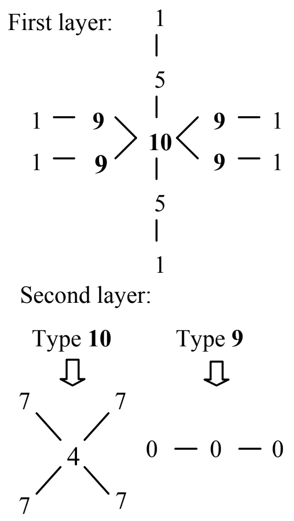
(b) gene type example for clustered building block

Figure 1. Genotype representation for MEMS components

Figure 2 shows a complete resonator design that was evolved from our application test case described later in this paper. Its genotype is a tree structure with two layers. This structure has 26 parameters, overall. Since it is not practical to show them all in the genotype representation, we just show the connectivity diagram, and the hierarchical expansion diagrams for the cluster blocks. This genotype display scheme will be used throughout the paper.

When higher level systems are synthesized, such as a MEMS band-pass filter composed of multiple resonators, clusters like this resonator will be drawn from the component library. Well-designed and individually optimized components are a way to capture engineering expertise and make it readily available for future designs.

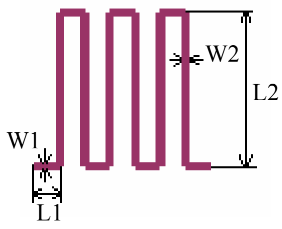




Gene type	MEMS design component
10	I-shaped center mass
9	Polyline spring
7	Beam for I-shaped center mass
5	Comb drive
4	Mass plate
1	Anchor
0	Beam

Figure 2. Genotype representation for resonator example

Another way to make MOGA more efficient is by reducing the dimension of the search space through a severe limitation of the available degrees of freedom. We encode higher-level cluster blocks with just those parameters that are absolutely vital to maintaining enough design flexibility. An example is the serpentine design shown in Figure 3. Even though it is composed of 13 individual beams, which when specified individually would require a total of 39 parameters, this component uses only 6 parameters in its geometric description. Using this cluster block will result in a much more compact layout and in a much more efficient search process.



Serpentine: type=2

Geometrical parameters:

L1, W1, L2, W2

angle, number of

loops

Figure 3. Serpentine component for MEMS design

2.2 Evaluation of Phenotypes

The phenotypes are evaluated using a MEMS simulation tool – SUGAR [8]. Each component in the component library is associated with its net list compatible with SUGAR. A net list file is created based on the genotype, and sent to SUGAR to evaluate the design performances. Based on the design performances of all the phenotypes in the same generation, Pareto ranking [6][9] is applied to the current generation and a fitness value is assigned to each individual design.

Two types of design validity checks are performed for any new generated design during the MOGA process: a design disjointedness check and a self-intersection check. A traversal through the layout tree makes sure that all nodes can be reached. For the intersection test, each component is given one or multiple rectangular bounding boxes. The separating axis theorem [10] is then applied to each pair of bounding boxes. If the new design is disjointed or self-intersected, it is an invalid design and is discarded before being sent to SUGAR, the MEMS simulation engine for performance evaluation. This dramatically reduces the computational power that MOGA might waste exploring irrelevant regions of the search space.

2.3 Controlling the Genetic Operations

Each component has instructions as to the allowed genetic operations during the evolutionary process. Before starting the synthesis process the designer can specify what mutation operations are allowed for a design component, and between which components a crossover operation can be applied. These are the specific operations that our process uses:

Selection. The "Roulette Wheel" selection scheme [6] is used in the MEMS design synthesis process, which follows the principle of fitness-proportionate selection for reproduction. The design with higher fitness value has a higher probability of being selected to be passed to future generations. A small portion of the best designs are passed to the next generation as elitism. Other selected designs are recombined through crossover operation and mutated to generate designs for the next generation.

Crossover. In the crossover operation pieces of genetic information are exchanged between two selected phenotypes. As exchanging arbitrary sets of parameters may not make functional sense, we use multi-point cut-and-splice crossover instead of one-point cut-and splice crossover. Any design component in a genotype can be swapped with a compatible design component in another genotype.

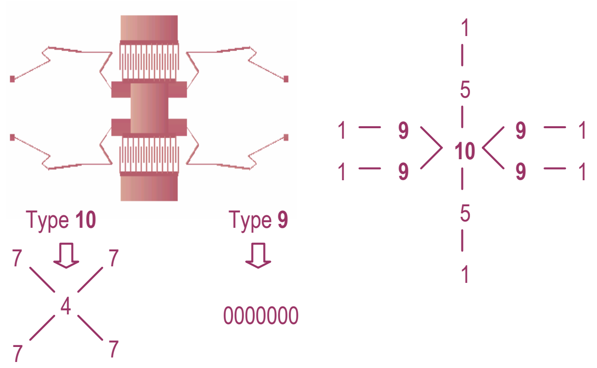


Figure 4. The second resonator design selected for crossover (The polyline springs have multiple beams connected in series, the connecting lines will not be shown from now on.)

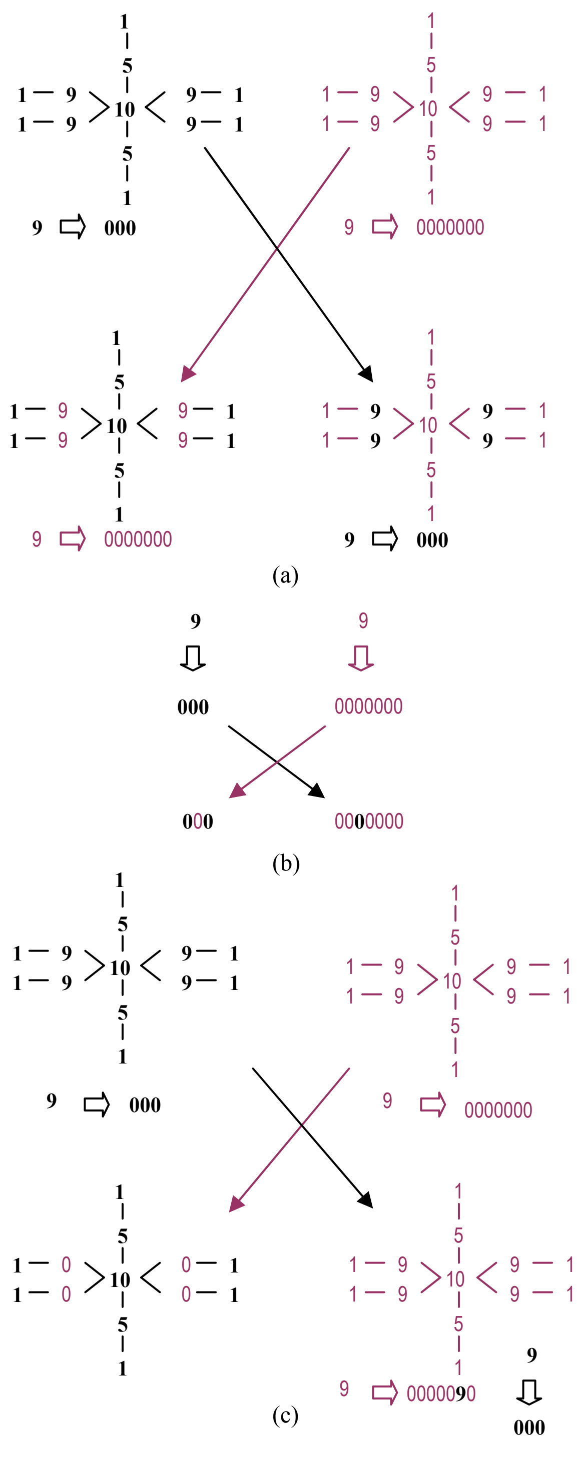


Figure 5. Crossover operations in the genotype domain during GA process

Moreover, we restrict the allowed crossovers even further. The designer determines which genes are compatible genes based on their functionality and connectivity. As an example, if the resonator designs shown in Figure 2 and in Figure 4 are selected for crossover, the operation should occur between the cluster genes in the first layers of the genotypes as in Figure 5(a), between the primitive genes in the second layers of the genotypes as in Figure 5(b), or between the cluster gene in the first layer and the primitive gene in the second layer of the genotypes as in Figure 5(c).

It might appear that it is very tedious to formulate this kind of tree-structured genotype representation with crossover constraints. However, with the integration of MOGA and the hierarchical design component library, in which each component is treated as a gene and embedded with instructions for the crossover and mutation operations, the genotype is naturally formulated during the evolutionary process. The complicated crossover operations are done automatically, following predefined high-level crossover instructions. The designer only needs to think at the appropriate design component level and provide design specifications and constraints in a modular building block fashion.

Mutation. The mutation operation makes a random modification on a selected individual design. The result may be a change in topology of a random selected cluster gene or a modification of the geometrical parameters of a primitive gene.

2.4 MOGA Process for MEMS Synthesis

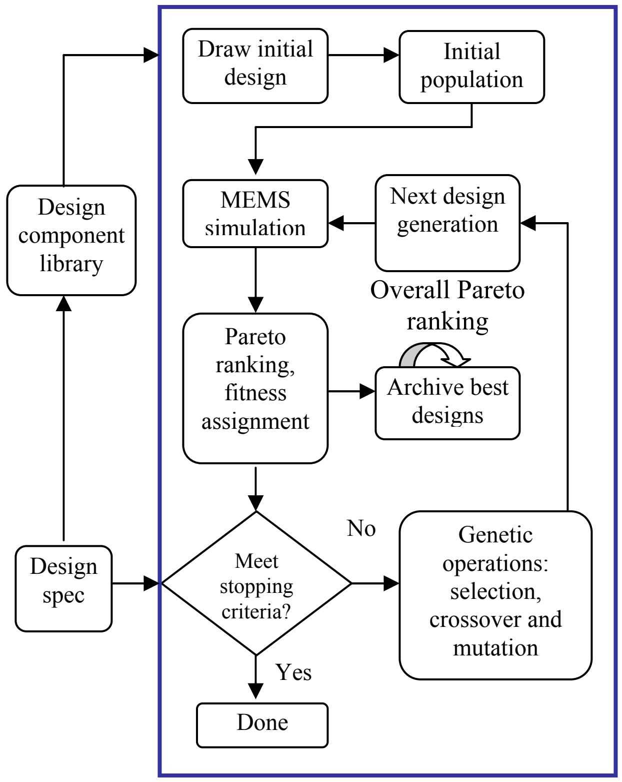


Figure 6. MOGA process for MEMS design synthesis

The overall flow of our MOGA is shown in Figure 6. Before starting the MOGA process, the designer needs to specify the design objectives, constraints, and stopping criteria by filling in the provided forms. The designer also needs to provide a valid initial design [11], possibly taken from the design component library, as well as the special design constraints for the current task. MOGA then makes many strongly mutated copies of the initial design to produce the first generation for the GA process.

The restricted genetic operations are applied to the current generation to generate the designs of the next generation. The new generation goes through the same process again. The MOGA process provides the designer multiple design solutions for comparison when the stopping criteria are met.

As GAs are stochastic global search methods, they are quite likely to generate close-to-optimal solutions at some intermediate point, but later lose them again in the evolutionary process. Therefore, we add an archiving procedure after each generation of the MOGA process. The best designs, based on a Pareto ranking of all designs seen so far, are copied into this archive. The best designs found during the whole evolutionary run can then be presented to the designer at the end of the design synthesis process.

3. TEST WITH RESONATOR DESIGN

To evaluate the efficiency of the new MOGA process for MEMS design synthesis, it was applied to the micro-machined resonator test case described in [3]. There were two design objectives: (1) minimize error from the target resonant frequency of 10,000 Hz and (2) minimize the device area, which is defined as the area of the bounding box of the device without considering the anchor pads. Also, the resonant motion must occur in the y-direction; thus the design must have higher stiffness in the x-direction than the y-direction. Four design cases were tested. Case 1 has asymmetric poly-line legs with unconstrained beam angles; Case 2 enforces a bilateral mirror symmetric layout around the y-axis; Case 3 employs 4-fold symmetry; Case 4 also uses 4-fold symmetry, but with suspensions restricted to be rectilinear and axis-aligned. The constraints on the parameter values are the same as in the comparison approach [3].

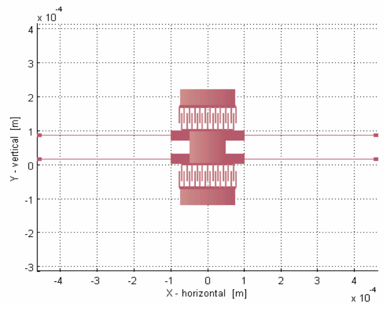
Each MOGA process was run with a population of 400 for 50 generations. The best designs found in 5 MOGA runs for each case were compared with the best results in [3] (Table 1). "Best" design was defined as the one with minimum design area, that also satisfied the following design constraints: resonant frequency within 5% of goal frequency (10,000 Hz), and the stiffness ratio requirement ($K_x/K_y \geq 1$).

Table 1 Comparison of the design area between the best designs synthesized from our new MOGA process and the best designs from the MOGA process used in [3]

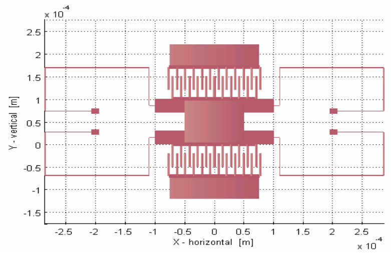
Cases	**Area (m^2)** (New MOGA Process with 50 Generations, 5 runs)	**Area (m^2)** ([3] with 50 Generations, 25 runs)	**Area (m^2)** ([3] with 500 Generations, 10 runs)
Case 1	1.633E-7	*	2.073E-7
Case 2	1.625E-7	1.711E-7	1.713E-7
Case 3	1.463E-7	1.550E-7	1.485E-7
Case 4	1.599E-7	1.703E-7	1.455E-7

* No synthesized design satisfied the design frequency target objective.

From the results shown in Table 1 we can see that the new MOGA process for MEMS design synthesis results in better designs for all the cases compared to the results of [3] with 50 generations and statistical significance (p =0.005; only cases 2, 3, and 4 were evaluated as there was no comparison results for case 1), even though each case only ran the MOGA process 5 times, thus using five times less computational effort compared to the 25 MOGA runs used in [3]. The average improvement in area over the runs was at p=0.04. Compared to the best results of [3] with 500 generations and 10 MOGA runs, which means 20 times more computational cost than our test case, our new MOGA process results in better designs in most cases except case 4. However, Case 4 after 5 runs with 100 generations yields results that exceed those in [3] with a design area of 1.367E-7 m^2. The best phenotype design is shown in Fig. 7 (b) below.



(a) Initial Design



(b) Best Case 4 Design

Figure 7 Initial design (a) and (b) best synthesized design of Case 4 from 5 runs with 100 generations; Layout Area: 1.367E-7 m^2; Resonant frequency: 10,497 Hz; Stiffness ratio (K_x/K_y): 2.7

There are several reasons that enable our new MOGA process for MEMS design synthesis to have better performance than the one used in [3]. First of all, the new MOGA process is integrated with a MEMS design component library and has a component-based genotype representation that enables more efficient GA operations, such as multi-point cut-and-splice crossover instead of one-point crossover restricted by the genotype representation of fixed-structured real-value strings used in [3]. Secondly, the designer has some control over the starting point of the MOGA process and can input their engineering knowledge via the starting design components and object-level constraints. Thirdly, the archiving process in the new MOGA process guarantees that the best design solutions reached in the intermediate generations during the stochastic search process are passed to the evaluation in the last generation. In fact, all of the best designs came from this previous generation cache: generations 37, 46, 38 and 23 for Cases 1, 2, 3 and 4, respectively. Finally, the new MOGA process is more computationally efficient. It avoids unnecessary design evaluation by using a more effective method for detecting invalid designs than the one used in [3].

In comparing the structure of the initial design to the best design for Case 4 in Fig. 7, there is a trend for the emergent designs to be characterized by legs that better utilize the layout space by turning inward. (Note, this trend also occurred with the other cases not shown due to space limitations.) It was also observed that the maximum beam length constraint was usually at the upper boundary of 100 μm. As an experiment, Case 4 was run with this constraint relaxed, but still within manufacturing requirements, and a design of the structure shown in Fig. 2 evolved with f = 10,219 Hz, Area = 1.587E-07 m^2, Stiffness Ratio = 21; maximum beam length 232.5 μm. This trend suggests that serpentine springs might be a useful object structure to add to the component library. To further explore the advantages of incorporating engineering know-how and human observations into the component-based genotype representation, we added a new design component – a simplified version of the serpentine spring shown in Figure 3 with the upper limit on the long beam constraint relaxed to 300 μm. It has only 2 parameters: length L1 and the number of loops. The number of loops can vary from 1 to 5. The length of the short beams (L2) is fixed at 12 μm, which provides enough space for the maximum displacement of the resonator in that direction. The widths of long and short beams are fixed at 2 μm and 6 μm, respectively, 2 μm being a reasonable lower limit set by the fabrication process available. This new test case, number 5, evolved into a synthesized design with performance as shown in Figure 8 based on just one MOGA run over 50 generations with a population of 400. It has the smallest design area of all the design cases. Further gradient-based optimization of the numerical parameters on this structure did not achieve further improvements on the area objective.

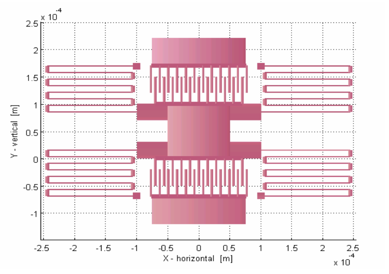


Figure 8 The best synthesized design of case 5. Design Area: 1.171E-7 m^2; Resonant frequency: 10,498 Hz; Stiffness ratio (K_x/K_y): 2; maximum beam length 138 μm

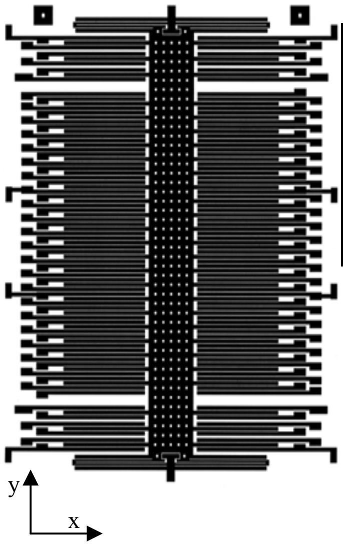
4. ADVANCED DESIGN SYNTHESIS: MICROACCELEROMETER

4.1 Design Synthesis Setup

A surface micromachined accelerometer is used as an advanced test case for the MEMS design synthesis process based on one of the widely used MEMS devices in automotive and other industrial applications. The design specification strongly depends on the particular application. Design automation is an efficient way to redesign the microaccelerometer for a different application and to optimize its performance.

The design topology and performance requirements are based on the commercial ADXL150 microaccelerometer [12], shown in Figure 9. Five design objectives related to the mechanical design of the microaccelerometer have been considered: the target resonant frequency (24kHz), stiffness ratio requirement ($K_x/K_y \geq 1$), maximizing the sense capacitance (no less than 133fF), minimizing the mechanical Brownian motion noise (no larger than $0.2\,mg/\sqrt{Hz}$), and minimizing the layout area, which is still defined as the area of the bounding box of the device.

To further explore the impact of the angle constraint and the advantage of incorporating engineering knowledge into the design component library, four microaccelerometer cases were synthesized; all of them use 4-fold symmetry. The design rules of the MUMP (Multi-User MEMS Process) fabrication [13] were used to guide our MEMS design synthesis, with one exception. Although the minimum design feature of MUMPs is 2 μm, we used a 1.3 μm finger gap in our simulations to be able to compare our results with the performance of ADXL150 [14]. Similarly, most other parameter ranges and constraints are set up to be as compatible as possible with this device (Table 2).



Gene type	MEMS design component
11	Differential comb drive
9	Polyline spring
4	Mass plate
1	Anchor
0	Beam

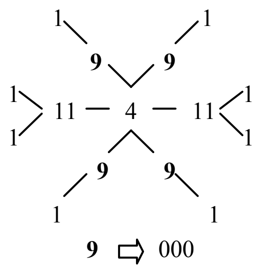


Figure 9 ADXL 150 layout [12] and our design genotype representation

Table 2 Design parameter constraints for microaccelerometer

Design Components	Parameter	Minimum Value	Maximum value
Mass Plate	Length	300μm	600μm
	Width	50μm	150μm
Differential Comb Drive	Finger length	30μm	130μm
	Finger width	2μm	4μm
	Finger overlap	20μm	120μm
	Cell number	1	100
	Finger gap	1.3μm	1.3μm
Polyline spring	Number of beams per leg	1	7
	Beam length	10 μm	100 μm
	Beam width	2 μm	5 μm
Serpentine spring	Number of loops	1	5
	Beam length	10 μm	100 μm
	Beam width	2 μm	5 μm
Simple Serpentine spring	Number of loops	1	5
	Beam length	10 μm	300 μm
	Beam width	2 μm	5 μm

Our MOGA test runs for microaccelerometer design synthesis evolve over 50 generations, using a population size of 100. The MOGA evolution for each case is executed five times, and the best designs are tallied. The "Best" design is defined as the design with the minimum layout area and with a resonant frequency within 5% of the target resonant frequency (24 kHz), which also satisfies the constraints described above.

4.2 Results and Discussion

The best designs for the four test cases with different design constraints are shown in Figure 10.

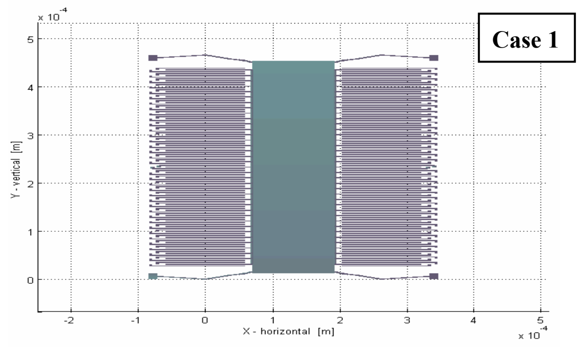
Case 1 uses arbitrary polyline springs. No particular engineering knowledge was incorporated into the spring design. Each beam segment in the spring has three free parameters: length, width and angle. The parameter search space is relatively large with up to 21 free design variables for the polyline springs and 4 variables for the shuttle. It resulted in the design with the largest area in the four test cases.

Case 2 restricts the polyline springs to Manhattan angles. Thus the individual beam angles alternate between 0/180 degrees and 90/270 degrees. This limits the size of the design space and the synthesis process reached a better design than Case 1 with the same computational cost. The design area of the best design is reduced by 21.9% while satisfying all other design requirements. This demonstrates that Manhattan geometry is an efficient design constraint for resonator suspension springs.

Case 3 uses the 6-parameter serpentine spring shown in Figure 3. It resulted in a design with a performance close to that of Case 2.

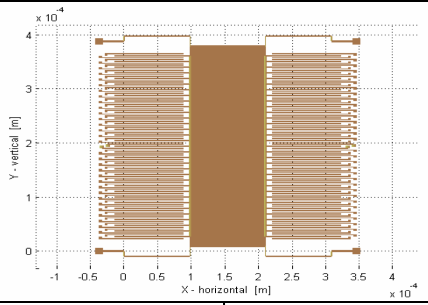
Case 4 employs the simplified axis-aligned serpentine springs described in Section 3 with only two design variables. The width of the long beam is fixed at 2 µm. The length and the width of the short beam are fixed at 10 µm and 5 µm, respectively. This efficient domain-specific encoding and the associated reduction in the search space resulted in a much better design solution than all other test cases – reducing the design area by 21.5% compared to test case 2 while still satisfying all other design constraints.

Case 1



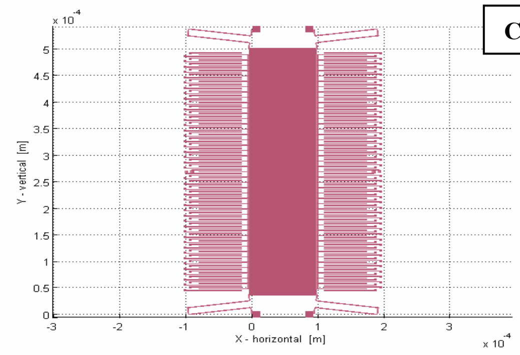
Shuttle dimension	Design Performance
Mass plate: Length 440 µm, Width 123 µm; Differential comb-drive: Finger length: 128 µm, Finger width: 2.4 µm, Cell number: 37, Finger gap: 1.3 µm	f= 23,296 Hz; area= 194,855 µm^2; K_x/K_y= 143; C_{sense}= 199fF; $a_{nf} = 0.143\, mg/\sqrt{Hz}$

Case 2



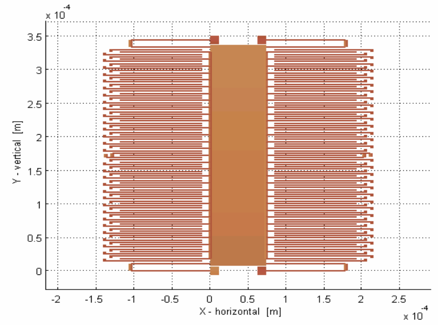
Shuttle dimension	Design Performance
Mass plate: Length 373 µm, Width 112 µm; Differential comb-drive: Finger length: 113 µm, Finger width: 2.2 µm, Cell number: 33, Finger gap: 1.3 µm	f= 23,814 Hz; area= 152,185 µm^2; K_x/K_y= 56; C_{sense}= 154fF; $a_{nf} = 0.144\, mg/\sqrt{Hz}$

Case 3



Shuttle dimension	Design Performance
Mass plate: Length 466 µm, Width 104 µm; Differential comb-drive: Finger length: 76 µm, Finger width: 2.0 µm, Cell number: 45, Finger gap: 1.3 µm	f= 23,341 Hz; area= 156,261 µm^2; K_x/K_y= 14; C_{sense}= 136fF; $a_{nf} = 0.115\, mg/\sqrt{Hz}$

Case 4



Shuttle dimension	Design Performance
Mass plate: Length 329 µm, Width 75 µm; Differential comb-drive: Finger length: 118 µm, Finger width: 2.1 µm, Cell number: 31, Finger gap: 1.3 µm	f= 24,936 Hz; area= 119,469 µm^2; K_x/K_y= 36; C_{sense}= 152fF; $a_{nf} = 0.196\, mg/\sqrt{Hz}$

Figure 10 Best synthesized designs for the four test cases

This once again demonstrates that incorporating engineering domain knowledge into the design synthesis process through component-based genotype representation can dramatically improve the performance of the MEMS design synthesis process.

5. CONCLUSIONS

An automated MEMS design synthesis program has been developed based on a multi-objective genetic algorithm (MOGA). To capture the hierarchical nature typical of many engineering designs, a tree-structured component-based genotype has been developed, which is based on an extensible design component library. This library includes different levels of design building blocks, ranging from simple geometrical primitives to complete resonator assemblies. Each building block in the component library is characterized by its own gene, and the full genotype has multiple genes connected together based on their functionality. This component library provides a highly effective and valuable gene pool and implicitly encodes much domain-specific engineering knowledge. This helps MOGA to converge to good design solutions much more efficiently.

Since MOGA is a stochastic global search method, it has a small probability of generating close-to-optimal design solutions in the intermediate generation, but later losing them during the evolutionary process. An archiving procedure has been added into the MOGA process, which guarantees that the best designs generated during the stochastic search process are remembered and available to the designer.

MOGA can handle multiple design objectives and generate multiple design solutions on the Pareto frontier. Comparing the configuration of the best designs, the designer can observe design patterns and find good design constraints favoring a certain application. The accumulated knowledge can be incorporated into the design component library and design constraints for better performance and for future use. This was demonstrated by both Case 5 of the resonator test case and Case 4 of the microaccelerometer test case. The simple serpentine spring, which was created based on the engineering observation on the emergent behavior of the resonator synthesis, generated the best designs for both resonator and microaccelerometer design examples. The final solutions of both design examples also existed as structures in the search space of other test cases; but they either did not make the Pareto set because they were not optimal under the original constraints or were not likely to be found in a reasonable amount of time because of the large search space involved.

The synthesis results of the test cases using different design components demonstrate that incorporating engineering knowledge can focus the stochastic global search on promising search regions and find better design solutions within a practical computational time. With more and more design components incorporated with engineering knowledge feedback into the design component library, we plan to develop an indexed design case library that provides good initial designs for the MOGA process and suggests reasonable parameter ranges based on the given design specifications.

6. ACKNOWLEDGEMENTS

This research was conducted in part through NSF grant CCR-DES/CC-0306557.

7. REFERENCES

[1] Zhou, N., Zhu, B., Agogino, A. M. and Pister, K. S. J., "Evolutionary Synthesis of MicroElectronicMechanical Systems Design," Intelligent Engineering System through Artificial Neural Networks Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE2001), 11, ASME Press, pp.197-202, 2001.

[2] Zhou, N., Agogino, A. M., and Pister, K. S., "Automated Design Synthesis for Micro-Electro-Mechanical Systems (MEMS)," CD-ROM Proc. of ASME Design Automation Conference, 2002.

[3] Kamalian, R., Agogino, A.M., and Takagi, H., "The Role of Constraints and Human Interaction in Evolving MEMS Designs: Microresonator Case Study," Proceedings of DETC/DAC, Paper # DETC2004-57462, CD ROM, ISBN # I710CD, 2004.

[4] Kamalian, R., Takagi, H. and Agogino, A.M. "Optimized Design of MEMS by Evolutionary Multi-objective Optimization with Interactive Evolutionary Computation," Proceedings of GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, pp. 1030 - 1041, June 2004.

[5] Holland, J. H. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press; 1975.

[6] Goldberg, D. E., Genetic algorithms in search, optimization, and machine learning. New York: Addison-Wesley Publishing Company; 1989.

[7] Graf, S., "GA Building Blocks and Data Structures for MEMS/NEMS Design Automation and Synthesis," Diploma Thesis, Department of Computer Science, RWTH Aachen, Oct. 2004.

[8] http://www-bsac.eecs.berkeley.edu/cadtools/sugar/sugar/

[9] Srinivas, N., Deb., K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," Evolutionary Computation, 1995, 2(3), pp.221-248.

[10] David H. Eberly, 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. Morgan Kaufmann Publishers, San Francisco, CA; 2000.

[11] Zhang, Y., Kamalian, R., Agogino, A.M., Séquin, C.H., "Hierarchical MEMS Synthesis and Optimization," Smart Structures and Materials 2005: Smart Electronics, MEMS, BioMEMS, and Nanotechnology, Proceedings of SPIE Vol. 5763, pp.96-106. CD ROM, Paper # 5763_12.

[12] Samuels, H., "Single- and Dual-Axis Micromachined Accelerometers", Analog-Dialogue, Vol.30, No.4, 1996.

[13] http://memsrus.com/documents/polyMUMPs.DR.V11.pdf

[14] Senturia, S.D., Microsystem Design. Norwell: Kluwer academic publishers; 2001.