Proceedings of DETC 2002:
Design Automation
Sep. 29-Oct. 2, 2002, Montreal, Canada

AUTOMATED DESIGN SYNTHESIS FOR MICRO-ELECTRO-MECHANICAL SYSTEMS (MEMS)

Ningning Zhou
Department of Mechanical Engineering
UC Berkeley, Berkeley, CA 94720-1740
nzhou@me.berkeley.edu

Alice Agogino
Department of Mechanical Engineering
UC Berkeley, Berkeley, CA 94720-1740
aagogino@me.berkeley.edu

Kristofer S.J. Pister
Department of EECS
UC Berkeley, Berkeley, CA 94720
pister@eecs.berkeley.edu

ABSTRACT

This paper proposes a general architecture for using evolutionary algorithms to achieve MEMS design synthesis. Functional MEMS devices are designed by combining parameterized basic MEMS building blocks together using Multi-objective Genetic Algorithms (MOGAs) to produce a pareto optimal set of feasible designs. The iterative design synthesis loop is implemented by combining MOGAs with the SUGAR MEMS simulation tool. Given a high-level description of the device's desired behavior, both the topology and sizing are generated. The topology or physical configuration includes the number and types of basic building blocks and their connectivity. The sizing of the designs entails assigning numerical values to parameterized building blocks. A sample from the pareto optimal set of designs is presented for a meandering resonator example, along with convergence plots.

INTRODUCTION

Micro-Electro-Mechanical Systems (MEMS) are miniaturized mechanical devices and components, often integrated or interfaced with electronics and fabricated on silicon wafers (Petersen, 1982). To facilitate the rapid development of MEMS technology, synthesis tools are in great demand to automate the MEMS design process. The goal of MEMS synthesis is to automatically generate feasible optimal solutions. There are two stages of MEMS design: device synthesis and fabrication process design. This research focuses on methodologies for device synthesis.

MEMS design usually begins with high-level specifications that describe the desired behavior of devices. Both the physical configuration and geometries must be chosen such that the resulting device satisfies the design objectives and operating constraints. There has been some progress toward direct device synthesis in the form of mask layout, fabrication and associated device modeling, given specified structures (Li, 1998). Lo et al. (1996) developed tools to enable designers, starting from a given high level topological abstraction of the system, to extract parameterized coupled electro-mechanical models, perform equivalent SPICE simulations, optimize the design parameters, and finally synthesize layout. This tool only handles parametric, not topological, synthesis. Mukherjee et al. (1997) and Jing (2000) generated valid layouts of commonly used MEMS device topologies from high-level engineering design specifications. This approach first identifies the design variables for a given design, models the problem as a formal numerical optimization problem, and then solves it with conventional optimization techniques.

All of the synthesis methods reported above, however, are tailored to specific tasks and are not general purpose synthesis tools. This paper proposes an evolutionary architecture to realize automated device synthesis. Evolutionary algorithms are global stochastic optimization techniques (Goldberg, 1989) with high robustness. They are non-problem specific, and can be applied to any problem with a well-formulated objective function. Another motivation for using evolutionary algorithms is their ability to concurrently search for multiple solutions in parallel using a population of solutions. This motivates their application to multi-objective optimization problems. Multi-Objective Genetic Algorithms (MOGAs) have been successfully applied in many fields in engineering optimal design (Narayanan et al., 1999). This paper incorporates MOGAs into MEMS design because of their ability to incorporate multiple non-commensurable objectives (Schaffer, 1985; Tamaki et al., 1996; Zitzler et al., 2000). Pareto optimality provides MEMS designers with a family of 'equally good' or 'non-dominated' solutions, therefore providing more design flexibility.

SYNTHESIS ARCHITECTURE

Genetic algorithms strive to find the best (or at least, a very good) solution to a problem by maintaining a population of individuals over a series of generations. Each individual in the population represents a candidate solution to the given problem. This candidate solution in the original search space is called a phenotype. A phenotype is encoded as a string of characters or real numbers called a genotype. It can also be encoded as other data structures such as trees. An initial population is randomly generated. Each individual in the population is evaluated by an objective function and assigned

fitness values. The GA transforms a population of individuals into a new generation of the population using genetic operators. The iteration continues until an individual or a family of individuals is found to meet the objectives.

A software architecture has been created to implement such an evolutionary-algorithm-based framework. Figure 1 illustrates an evolutionary iteration loop for MEMS design. Solutions are first encoded into the genotype format and an initial random population of designs is produced in this format. Each design is first checked for geometrical validity. A cost vector to be minimized is then calculated to quantify performance of an individual design. SUGAR (Zhou et al, 1998; Clark et al, 2002) is incorporated as a forward simulator into the MOGA's iteration loop to evaluate the performance of each individual candidate design. SUGAR is a MEMS simulation package developed at the University of California at Berkeley (http://bsac.berkeley.edu/cadtools/sugar/). If the performance doesn't meet the objectives, the whole population of the current generation is ranked using pareto optimality. A fitness value is assigned to each design based on its rank. Elitism, selection/crossover and immigration are then applied to the current generation to form individuals in a new generation. This MOGA loop iteratively searches for optimal functional designs that meet the specifications.

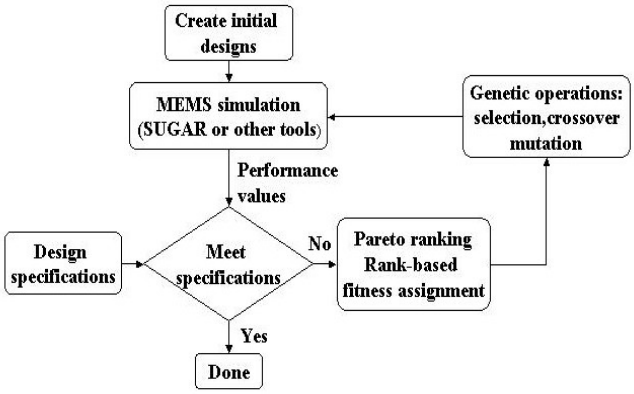


Figure 1 Flow chart of the evolutionary MEMS synthesis approach

MEMS GA REPRESENTATION

The basis of MEMS GA synthesis is the coding scheme that represents the features of a problem. This paper decomposes a MEMS device into MEMS GA building blocks. Many devices can be represented as a rooted acyclic tree of building blocks. 'Rooted' means that there exists exactly one distinguished node as the reference node. All other structures are laid out with respect to this node. MEMS GA building blocks are connected together at their nodes to form a tree. As an example, figure 2 shows a micro-resonator with a center mass supported by four serpentine spring structures. There are two sets of electrostatic comb drives on both sides to drive this resonator vibrating horizontally. This resonator, for example, can be decomposed into seven building blocks as shown in figure 3. There is not a unique way to decompose a device. In order to take advantage of the multiple configurations possible with the GA algorithm, we recommend that this decomposition be performed according to functionality, rather than physical structure.

The MEMS GA building blocks are pre-defined construction materials with which the devices can be built. Two levels of building blocks are defined here, however multiple levels in between these are possible in MEMS design (e.g., clusters of clusters). Our primary building blocks are (1) primitive elements, such as anchors, beams and electrostatic gaps, as well as (2) clusters that are comprised of primitive elements. A cluster is a connected configuration with more than one primitive element. It is represented as a subnet in SUGAR. For example, a cluster could be a serpentine spring or a folded-flexure spring (Fedder, 1994) composed of several beams and anchors. It could also be an electrostatic comb-drive composed of a number of electrostatic gaps (Tang et al. 1989). Each building block (primitive element or cluster) can be seen as a parameterized black box. It interacts with the surrounding building blocks through its ports or nodes. The inside configuration is defined with a set of governing parameters.

A building block is fully described by the following information:

- A building block type identifier;
- A list of ports or nodes through which this block can be connected to others;
- Governing parameters.

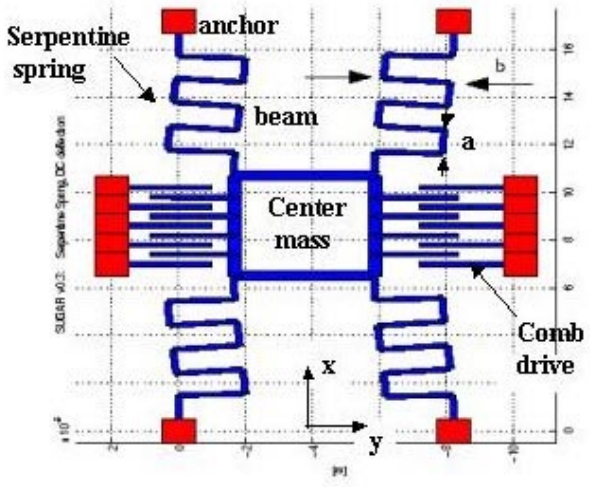


Figure 2 A MEMS resonator with four meandering springs

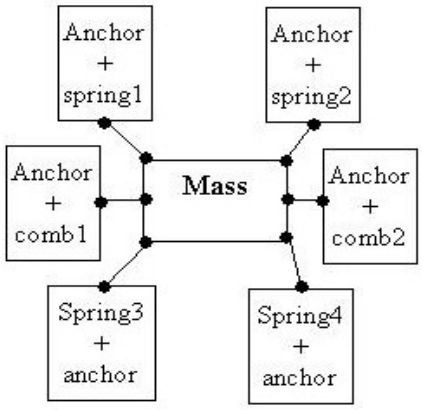
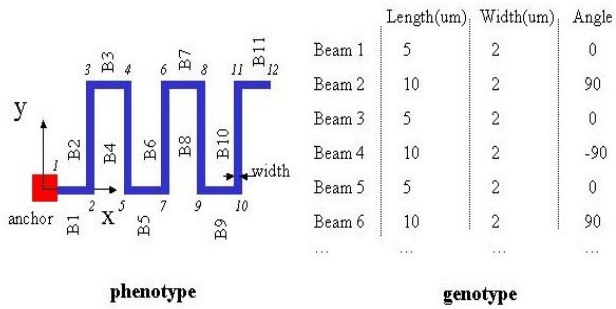


Figure 3 The building blocks and their connectivity

A device synthesis problem has to be defined before genetic algorithms are applied. First, the available types of building blocks are defined and supplied to the GAs. Second, a concept design with viable configurations are identified for a specific device and supplied to the GAs. For example, as an electrostatic actuator usually requires an anchor, spring and electrostatic gaps in order to actuate properly, a viable GA configuration must include these elements. The concept design for a certain device can be identified with heuristic knowledge by designers or by other knowledge-based tools. The GAs are then used to choose the best topologies and the numerical values of the associated parameters, finalizing the MEMS device designs.

PROBLEM STATEMENT

The synthesis of MEMS meandering resonators is demonstrated in this paper. Figure 2 shows one example of a meandering resonator. A center mass is supported by four serpentine spring structures. Each serpentine spring is a shown in figure 4. Beam 1 (B1) extends from anchor point node 1 to node 2 with length 5um, width 2um, at an angle of 0 with respect to the x axis; beam 2 (B2) extends from node 2 to node 3 with length 10um, width 2um, at an angle of 90 with respect to the *x* axis and so on. Each serpentine spring is encoded into a matrix of *N* by 3 as the genotype shown in figure 4, where *N* is the number of beams. Each row contains the length, width and angle of one beam. The matrix is ordered so that the first row corresponds to beam 1 that is the nearest to the anchor, the second row corresponds to beam 2 that is the second nearest to the anchor and so on. Serpentine springs with the angle of 0 and 90 degrees are special cases of meandering springs. With arbitrary length, width and angle of each beam, we can fully describe the configuration and geometries of a meandering flexure. More description of the synthesis of a single meandering spring to achieve the specified stiffness along the *x* and *y* directions can be found in Zhou et al. (2001).



	Length(um)	Width(um)	Angle
Beam 1	5	2	0
Beam 2	10	2	90
Beam 3	5	2	0
Beam 4	10	2	-90
Beam 5	5	2	0
Beam 6	10	2	90
…	…	…	…

Figure 4 A meandering spring and its representation

The goal of this problem is to design a micro-resonator by combining five parameterized building blocks: one center mass and four meandering springs. Each spring is a cluster of one-anchor and *N* beams connected subsequently. The parameters of each spring include the number of beams *N* and the length, width and angle for each beam. Each spring building block has only one node connected to the center mass. The center mass is defined as a four-node plate with fixed mass or a variable mass plate with parameters length and width. The design task is to choose the right building blocks and the numerical values of the associated parameters to achieve the specified design objectives.

ENCODING

A schematic of a meandering resonator is shown in figure 5. The center mass has node 1, 2, 3, 4 connected to building block 1, 2, 3, 4 respectively. This design is encoded as a 5x3 cell array as follows.

[mass]	[1 2 3 4]	[L W]
[building block 1]	[1]	[l_1 w_1 θ_1 l_2 w_2 θ_2 ……]
[building block 2]	[2]	[l_1 w_1 θ_1 l_2 w_2 θ_2 ……]
[building block 3]	[3]	[l_1 w_1 θ_1 l_2 w_2 θ_2 ……]
[building block 4]	[4]	[l_1 w_1 θ_1 l_2 w_2 θ_2 ……]

Each row of the cell describes one building block. The first column of the cell represents the building block type. The second column of the cell represents the building block node connection. The third column represents the building block parameters. The center mass has length L and width W. Building block 1 and 4 have similar configurations of one anchor plus N beams with l_1 (length) w_1 (width) θ_1 (angle) associated to the closest beam to the anchor and so on. Building block 2 and 3 have similar configurations of N beams plus one anchor with l_1 w_1 θ_1 associated to the closest beam to the center mass and so on. A meandering resonator can be fully described by this cell array.

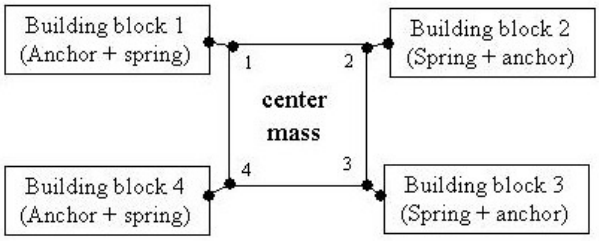


Figure 5 Schematic of a meandering resonator

During the initialization process, 30-40% of the randomly generated designs were found to be self-intersecting and thus were not practical from a fabrication perspective. A rejecting strategy was adopted to handle any illegal solutions.

CROSSOVER

Two kinds of crossovers are carried out. In parametric crossover, building blocks of the same type are grouped together first. For building blocks of meandering springs, the arithmetic crossover is carried out for the first N_{min} beams. The parameters of the center mass of the two parents are also mated in a similar way as for beams. The arithmetic crossovers are defined as the linear combination of two parents.

$$C_1 = \lambda_1 x_1 + \lambda_2 x_2$$
$$C_2 = \lambda_1 x_2 + \lambda_2 x_1$$

Where x_1 and x_2 are two parents, C_1 and C_2 are two child genes, λ_1 and λ_2 are two multipliers, and $\lambda_1 + \lambda_2 = 1$.

Two types of cut and splice crossover are performed in this problem. One parent design is broken into two pieces at a one node of a randomly selected building block. The second parent has to be broken at a similar location, for example, both at node 1. The corresponding pieces of two parents are exchanged. Splicing the two new pieces generates two offspring. This crossover ensures that each offspring will maintain the required functional configuration of one mass and four springs. The second cut and splice is carried out for beams within each spring building block.

MUTATION

The mutation operator is applied in this problem because the immigration reported previously (Zhou, 2001) was found to be inefficient. After 2 or 3 generations of evolution, almost no newly generated immigrants in a population were selected as parents for crossover by roulette wheel selection. This suggests that the immigration operator does not work effectively to introduce new genes into the evolutionary process.

To counter this problem, uniform mutation is applied to every design variable according to the mutation probability independently. Once a design variable is selected for mutation, this variable is replaced by a randomly generated real value within its boundaries.

SYNTHESIS RESULTS

As an illustrative example, a design problem with specifications taken from an existing serpentine resonator is shown in figure 2 with b=80um, a=60um, each spring beam

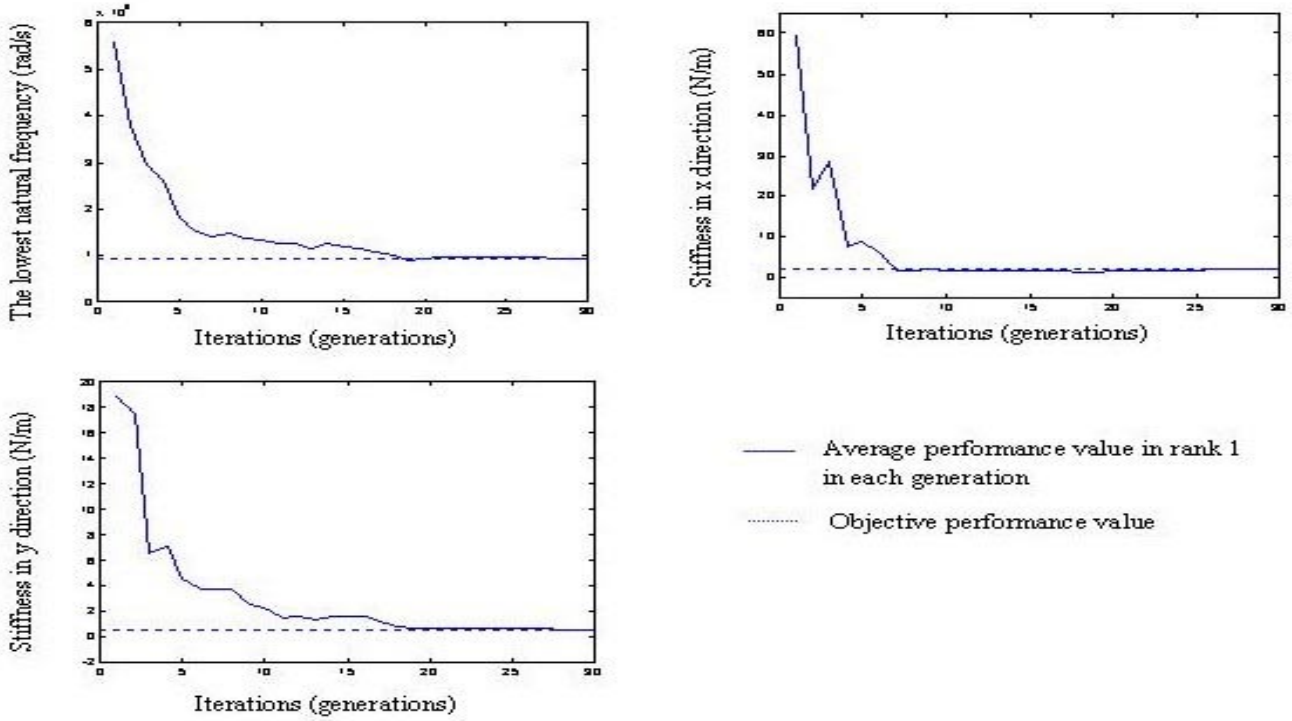


Figure 6 Convergence curve of the rank 1 solutions

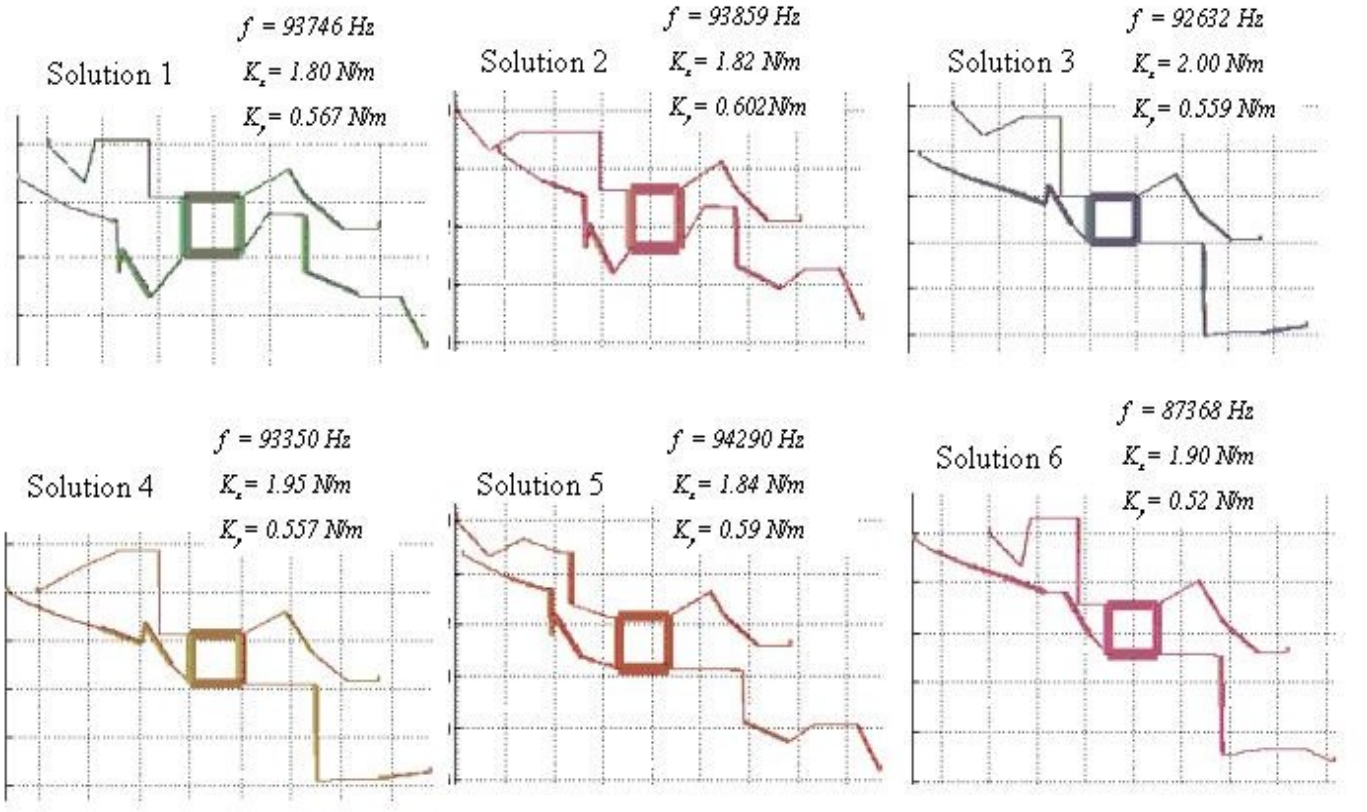


Figure 7 Design examples of meandering resonators from on MOGA run

cross-section 2x2 um, and each center mass beam 120x12.5x2 um. The goal is to design alternate meandering resonators to match the lowest natural frequency and the stiffness in x, y directions of the above serpentine resonator. The center mass is assumed to be the same as in the serpentine resonator.

Design Specifications:

- the lowest natural frequency f = 93723 rad/s;
- stiffness in x direction: K_x = 1.90 N/m;
- stiffness in y direction: K_y =0.56 N/m;

Design variables: number of beams N, length of beams l, and width of beams w, angle of beams θ for each meandering spring.

Design Inequality Constraints: $w_{min} \le w \le w_{max}$, $w < l \le l_{max}$,

$\theta_{min} \le \theta \le \theta_{max}$, $1 \le N \le N_{max}$. w, l, θ are real numbers, N is an integer.

The design and MOGA parameters are shown in Table 1, where n_{pop} is the population size, n_{gen} is the number of generations, P_c is the crossover probability, P_e is the total percentage of elite, and P_m is the mutation probability.

Table 1 Design constraints for meandering resonator

Design variables	w_{min}	w_{max}	l_{max}	θ_{min}	θ_{max}	N_{max}
	2um	20um	400um	-90	90	6

MOGA configuration	n_{pop}	n_{gen}	P_c	P_e	P_m	λ
	400	30	0.7	5%	0.1	0.3

Figure 6 shows how the evolutionary process converges over iterations. The solid curves represent the average natural frequency, x stiffness, and y stiffness of the best-performed individuals (rank 1) in each iteration. The dotted lines represent the specified goals. Three curves show how three performance values converge to their objectives (minimum distance from the specified goals) concurrently in a single MOGA run. The process converges faster at the early stage and slower at the later stage. No significant improvement is found in performance after 25-30 iterations.

A pareto optimal set of 26 designs is found from one MOGA run. Figure 7 shows some of the 'equally good' pareto solutions with different configurations. Solution 1 is the best in terms of the frequency; however, the kink in the lower left leg would probably be sub-optimal from both a fatigue and fabrication standpoint. Solution 6 is the best in terms of K_x. Solution 3 is the best in terms of K_y. Designers will choose the

right design from the pareto set that reflects their multiple objective trade-offs.

CONLUSION AND FUTURE RESEARCH

This paper presents an evolutionary algorithmic framework to automatically synthesize MEMS designs. Given a higher-level description of the device's desired behavior, both the topology and sizing of the devices are generated using a Multi-Objective Genetic Algorithm (MOGA). The iterative design synthesis framework is implemented by combining MOGAs with a MEMS simulation tool called SUGAR. MOGAs, starting with a set of pre-defined building blocks and a concept design configuration, present designers with a pareto optimal set of different design configurations that meet multi-objective design specifications. The synthesis results for a meandering resonator example demonstrate the feasibility of this method.

In the current MEMS GA architecture, a concept design is supplied to the MOGA process as an initial starting design. The results will vary depending on this initial design concept. We envision the MEMS GA algorithms as part of a larger MEMS synthesis architecture, shown in figure 8, where case-based reasoning for MEMS design is used to choose a concept design based on expert knowledge, coupled with machine learning algorithms. Case-based reasoning is an approach to solving new problems by using knowledge gained from solving similar problems in the past. A case library has to be collected and established in an indexed database. The reasoning tool then finds those cases in the library that have solved problems similar to the current problem, and proposes a ballpark starting point which is adapted to fit the current problem.

The research and development in the MEMS area has accumulated an increasing number of successful designs, sub-assemblies, and building blocks. Many have been useful and have been adopted in new devices. A library of MEMS designs with useful GA building blocks (clusters & primitives), indexed by function, materials, etc., would be a valuable shared resource to the MEMS design community. As shown in figure 8, designs could be stored as clusters of building blocks in the library. Case-based reasoning tools could be used to select the set of most closely matched designs to meet the input specifications. These could provide the initial conceptual design configurations for design case adaptation through MOGA optimization. Final configurations and parameters would be evolved to meet the input specifications. The newly designed devices are then fabricated, tested and added into the case library.

There are several issues in developing this system architecture:

1. The acquisition of design cases.
2. The development of a representation and organization of design cases.
3. The development of an indexing structure for storing and recalling design cases.
4. The development of a metric for selecting and retrieving design cases or subcases.
5. The design case adaptation and the method to verify the results of design case adaptation.

The long term research goal is to develop a synthesis CAD package for robust and efficient MEMS design.

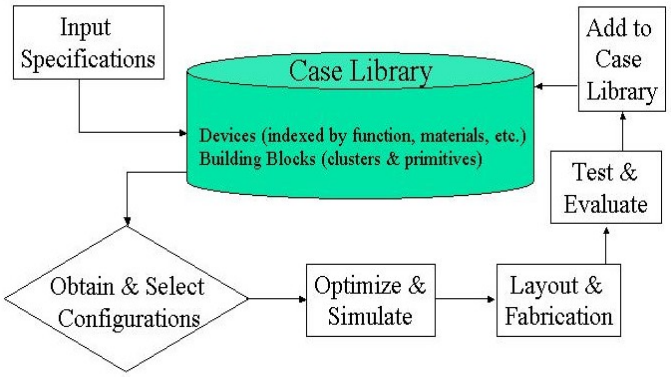


Figure 8 The case-based reasoning tool for MEMS synthesis

REFERENCES

1. Clark, J.V., Bindel, D., Zhou, N., Nie, J., Kao, W., Zhu, E., Kuo A., Pister, K.S.J., Demmel, J., Govindjee, S., Bai, Z., Gu, M. and Agogino, A.M., 2002, "Addressing the Needs of Complex MEMS Design," *Proceedings of the 15th IEEE International MEMS Conference*, (Jan. 20-24, 2002, Las Vegas, Nevada), IEEE, ISBN 0-7803-7187-9, pp. 204-209.
2. Fedder, G., 1994, *Simulations of Microelectromechanical Systems*, Ph.D thesis, UC Berkeley.
3. Goldberg, David E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
4. Jing, Q., Luo, H., Mukherjee, T., Carley, L.R. and Fedder, G., 2000, "CMOS Micromechanical Bandpass Filter Design using a Hierarchical MEMS Circuit Library'', *Proceedings IEEE Thirteenth Annual International Conference on Micro Electro Mechanical Systems*, pp. 187-192, Miyazaki, Japan.
5. Li, H. and Antonsson, E.K., 1998, "Evolutionary Techniques in MEMS Synthesis'', *Proc. DETC'98, 1998 ASME Design Engineering Technical Conferences,* Atlanta, GA.
6. Lo, N.R., Berg, E.C., Quakkelaar, S.R., and Simon, J.N., Tachiki, M., Lee, H.J. and Pister, K.S.J., 1996, "Parameterized Layout Synthesis, Extraction, and SPICE Simulation for MEMS'', *Proc. ISCAS*, pp. 481-484, Atlanta, GA.
7. Mukherjee, T. and Fedder, G., 1997, "Structured Design of Microelectromechanical systems'', *DAC '97*, Anaheim, CA.
8. Narayanan, S., and S. Azarm, 1999, "On Improving Multiobjective Genetic Algorithms for Design Optimization", *Structural Optimization*, Vol.18, pp. 146-155.

9. Petersen, Kurt E., 1982, “Silicon as a Mechanical Material”, *Proceedings of the IEEE*, Vol.70, No.5, pp. 420-457.
10. Schaffer, J.D., 1985, Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proceedings of the first International Conference on Genetic Algorithms*, pp. 93-100.
11. Tamaki, H., Kita, H. and Kobayashi, S., 1996, “Multi-Objective Optimization by Genetic Algorithm: A Review”, *Proc. 1996 IEEE International Conference on Evolutionary Computation*, pp. 517-522, Nagoya, Japan.
12. Tang, W. C., Nguyen,T.-C H., Judy, M.W. and Howe, R.T., 1989, “Electrostatic-comb drive of lateral polysilicon resonators’’, *Proc. of 5th International Conference on Solid-State Sensors and Actuators*, Montreux, Switzerland,
13. Zhou, N., Zhu, B., Agogino, A.M. and Pister, K.S.J., 2001, “Evolutionary Synthesis of MEMS MicroElectronicMechanical Systems Design’’, *Intelligent Engineering System Through Artificial Neural Networks, Proceedings of the Artificial Neural Networks in Engineering*, Vol.11, pp.197-202, ASME Press.
14. Zhou, N., Clark, J.V. and Pister, K. S. J., 1998, “Nodal Simulation for MEMS Design Using SUGAR v0.5”, *1998 International Conference on Modeling and Simulation of Microsystems Semiconductors, Sensors and Actuators*, pp. 308-313.
15. Zitzler, E., Deb, K. and Thiele, L., 2000, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation* Volume 8, Number 2, MIT Press, 173-195.