Evolutionary Synthesis of MEMS

by

Raffi Roupen Kamalian

B.S. (University of California, Berkeley) 1996

M.S. (University of California, Berkeley) 1998

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy
in
Engineering - Mechanical Engineering
in the
GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Alice M. Agogino, Chair
Professor Albert P. Pisano
Professor Kristofer S.J. Pister

Fall 2004

Evolutionary Synthesis of MEMS

by

Raffi Roupen Kamalian

The dissertation of Raffi Roupen Kamalian is approved:

Professor Alice M. Agogino, Chair Date

Professor Albert P. Pisano Date

Professor Kristofer S.J. Pister Date

University of California, Berkeley

Fall 2004

Abstract

Evolutionary Synthesis of MEMS

by

Raffi Roupen Kamalian

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Alice M. Agogino, Chair

An evolutionary synthesis framework for Microelectrical Mechanical System (MEMS) design is presented. MEMS based technologies promise to bring a revolution to the world we live in just as the integrated circuit has done in recent decades; better design tools are critical to this revolution. More complex design objectives and constraints demand automation to generate successful devices. Genetic algorithms and other stochastic evolutionary synthesis approaches are used to design surface micromachined MEMS using flexural suspensions and electrostatic actuation.

A general MEMS synthesis approach is presented, as well as data-structures for describing designs and applying synthesis algorithms. Synthesis is based on the use of reduced order modeling to simulate design performance. The application of the concept of shape grammars for MEMS synthesis is discussed and applied to the generation of viable initial resonator designs. Human Interactive Evolution Computation (IEC) is applied to MEMS to improve synthesis performance; user studies show an increase in output quality through the use of human interaction compared to our non-interactive synthesis tool.

The applicability of our approach, design encoding, objectives and constraints are discussed for several MEMS examples, including resonating masses, accelerometers and gyroscopes. We validate of our approach through fabrication and characterization, successfully generating MEMS devices with measured performance that matches simulation. The results of the characterization are studied to further improve our method for more accurate synthesis.

Professor Alice M. Agogino, Chair Date

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would first like to acknowledge the support of Professor Alice Agogino in getting me to the point where I am writing the acknowledgement section of a thesis! I could not be here without her faith, generosity and time.

It has been a long road from when I came to UC Berkeley as a freshman to this point many, many years later; many faculty and staff have helped me along the way, I can not thank them all, but I will try my best to mention a few of the standouts: I would like to acknowledge Professors Al Pisano and Kris Pister not only for being on my thesis committee but also for being two of my most inspiring teachers and advisors throughout the years. I am continually amazed at their energy and enthusiasm. Dr. Dennis Lieu has been an inspiration to me as a student, as a teaching assistant and even a martial arts classmate. There are many other professors at UC Berkeley I must acknowledge for their contribution to my education, Professors Demmel, Govindjee, Der Kiureghian Fernedez-Pello, Auslander and Dharan. Finally I must thank Professor Kazerooni, my master's degree advisor, if not for him, I would not have had the opportunity to continue at UC Berkeley and realize that a Bachelor's degree is only the beginning, not the end.

I would like to acknowledge my lab mates at the BEST Lab, the SUGAR group, and fellow BSAC researchers who were willing to be my synthesizers, proofreaders, and constructive criticizers. Specifically, I would be nowhere without the SEM skills of Lixia Zhou and the simulation expertise of David Bindel, the help of Ying Zhang, Sebastian

Graf, Corie Cobb, Vinu Ranganathan, and David Garmire. Lastly I would like to acknowledge Ningning Zhou whose footsteps I tried to follow.

Looking towards the future, I would like to acknowledge Professor Hideyuki Takagi at Kyushu University for introducing me to human interactive evolution computation, a significant part of this thesis and my future research, working in his laboratory as part of the NSF-JSPS East Asian Summer Institutes program has literally led me to many new places.

Breadth of knowledge is an important feature of the Berkeley education that I now greatly appreciate. I would like to acknowledge Li Laoshi and Xing Laoshi inspiration and understanding and the Department of East Asian Languages and Cultures for their support of my linguistic divertimenti.

I would like to acknowledge the support of a few of my best friends throughout my years of study: Eric, Patrick and Jimmy, Brandon, Lisa, Kevin, and, of course Mae. Lastly, I must acknowledge my parents, who helped define who I am, what I do, and how I do it.

Chapter 1: Introduction

The goal of this thesis is to present synthesis methodologies suitable for Microelectrical Mechanical Systems (MEMS) design. We will present work done in further developing tools to automatically generate valid, useful MEMS configurations optimized for a given set of performance goals and constraints. We are limiting this work to the design of surface micromachined MEMS devices using flexural suspensions and electrostatic actuation.

This thesis focuses on synthesis using stochastic methods, particularly evolutionary methods. Human interactive stochastic methods and classical deterministic optimization methods are also included. Research on the impact of design constraints for synthesis is presented to illustrate their role in optimal MEMS synthesis applications. We demonstrate validation of our approach through fabrication and characterization and also study how to further improve our method for more accurate synthesis. Finally a survey of current ongoing and future research in this area is presented.

In the following section, an overview of the field of MEMS will be given; this will be followed by a review of MEMS simulation and the motivations for MEMS synthesis and optimization.

1.1. Introduction to MEMS

The field of MEMS popularly referred to as *micro-machines,* is already a well-established field of academic research and is quickly becoming a mature technology with a wide array of commercial applications. As the name suggests, the term MEMS usually includes both electronic and mechanical components, but the term is also applied to microscopic devices from other domains, including fluidic, thermal and magnetic systems.

Current MEMS commercial applications include inertial sensors, such as accelerometers and gyroscopes; microfluidic systems, such as inkjet printer heads; pressure and chemical sensors; microrelays, optical switches, display technology, to name a few. Other MEMS applications such as micro power generators, DNA sequencers, in-vivo health monitoring systems, RF signal processing and distributed sensor networks are being researched in academia [1,2,3,4].

Just as integrated circuit based technology has grown in leaps and bounds since the transistor was invented in 1945 at Bell Labs, moving far beyond its initial application areas, MEMS development over the next decades may take us to many new application areas not yet considered. The ability to add ubiquitous sensors (and intelligence) to any portion of a vehicle or tool or household object, or even the human bodies will allow us control over our environment that we have not even dreamed of yet.

One driving force in the development of MEMS applications is the advance in microfabrication technology. MEMS are often created using the same type of microfabrication used in the microelectronics industry. This is usually a lithography-based patterning combined with thin film deposition and chemical or plasma etching. An additional benefit is that this fabrication method allows for the creation of MEMS systems including not only electromechanical components, but also control electronics, integrated onto a single monolithic chip [5,6].

As microfabrication technology pushes ever further towards the sub-micron level, new avenues of MEMS devices are also made available, including using carbon nanotubes for their unique mechanical and electronic properties [7].

In concert with the development of MEMS fabrication technology is the rise of standard MEMS processes, offered through commercial fabrication houses [8,9]. This allows designers the ability to focus on device design and bypass process design. Allowing commercial and academic designers to prototype MEMS designs without access to a multi-million dollar microfabrication lab and months of training. Despite the speed, cost savings and reliability offered by these standard processes, the time required from submission of layout to receiving a chip back is still a bottleneck in the design process. This can be compensated for eliminating the conventional ‘build and break’ iterative process through the use of better design tools.

Advances in MEMS fabrication allow for more advanced geometries and more advanced applications require larger, more complex layouts with more strict performance requirements. Using simulation software to predict the performance of a design before it is fabricated can lead to great improvements in shortening the design cycle and reducing cost, but simulation is only one part of the design equation. The demands of multiple design objectives combined with multiple design variables and constraints calls for a better solution than simply using simulation tools. In parallel with the development of MEMS fabrication and simulation technology, MEMS design tools must also be developed.

Process design and simulation are also very important sectors of MEMS development, but we will focus our attention specifically to the area of surface micromachined electromechanical MEMS design for standardized processes. In the next section we will discuss the development of CAD tools to predict MEMS performance, which is a critical step in our approach to automated MEMS design.

1.2. Introduction to MEMS Simulation with Reduced Order Modeling

Due to the broadness of the definition of MEMS, the simulation requirements vary greatly across the spectrum. From a hierarchical point of view, Senturia describes four levels of MEMS modeling [10]: *system*, *device*, *physical* and *process*. For the purposes of this thesis, we are focusing on the *physical* level, although the methods presented here could be used at the *device* or *system* level as well. Furthermore by designing in a standard process, we forgo the need to design on the *process* level. Senturia describes the *physical* level as real devices in the three dimensional continuum. At a mathematical level, these devices can usually be described by partial differential equations, however for more complex designs, analytical or approximate methods of modeling performance are needed. An issue particular with the simulation of MEMS devices is that they inherently involve multiple domains: at the least mechanical and electrical, often also including optical, thermal, magnetic and/or fluidic. The complicated interaction between these domains requires numerical analysis tools.

Numerical simulation methods, built on finite element, boundary element or finite difference methods have increasingly become popular to simulate the performance of complex MEMS devices [11]. Finite Element Analysis (FEA) discretizes the structure being modeled into small elements. Each element's behavior can be described by several relatively simple equations, and using a computer, the effect of each element's behavior can be aggregated across the entire structure. The data from these elements can be used to find the displacement in a structure or calculate the stresses inside of an object. The accuracy of these calculations depends on the size of the elements and the element type used. Unfortunately smaller elements results in more computation, requiring longer simulation time.

Commercial FEA developers and MEMS layout software authors are increasingly simplifying the process of passing geometric models between design and simulation tools. Companies such as ANSYS [12], ALGOR [13], ABAQUS/Coventor [14] specifically market tools for numerical modeling of MEMS structures.

The biggest limitation of FEA however is the speed required for a simulation. A common MEMS design can contain thousands or tens of thousands of elements. If a higher-level device is being simulated, this can grow exponentially. For complex analyses minutes, hours or days can be required for a single simulation; these types of time requirements make iterative design optimization with FEA less practical. Furthermore many commercial and academic FEA packages are not easily interfaced by external programs, as is required in design synthesis.

A solution to both these limitations is to use reduced order simulation software, such as SUGAR, an open source simulator created at UC Berkeley [15,16]. SUGAR uses a Modified Nodal Analysis (MNA) approach to look at designs, MEMS designs are broken down into discrete components at a higher level than FEA, rather than describing a suspension into hundreds of elements, it describes them it into individual beams, connected at nodes. These components together form a coupled system of ordinary differential equations, which can be solved numerically to find the behavior at each node. This approach is similar to a signal flow graph representation. For mechanical systems it is similar to the way a circuit is analyzed using Kirchoff's Current Law, except the potential and flow variables are force and velocity rather than voltage and current [17].

This method does not capture the same level of internal behavior as FEA, nor can it be used to accurately model all types of behavior, but, as it reduces the order of the model by orders of magnitude, it is much faster to simulate. Furthermore the method of describing individual components lends itself well to parametric synthesis.

SUGAR is written in C++ but can be interfaced directly from MATLAB, allowing for easy integration into an optimization routine. MEMS design information is written to a netlist file that can be loaded, along with a process file that describes how the design would be fabricated, into SUGAR and evaluated. SUGAR also has the ability to translate a netlist into a layout file that can then be imported into a layout tool such as Cadence or a FEA package for further modeling. For these reasons we have chosen SUGAR as the simulation engine for our iterative MEMS synthesis approach.

1.3. Motivation for MEMS Synthesis and Optimization

Synthesis can be described as the generation of a design with performance objectives subject to specified constraints. Optimization is the process of searching for solutions that correspond to extreme values of one or more performance objectives. Antonsson and Cagan describe synthesis as:

> *"the reverse of the analytical process… design begins not with a description of of a device or system, but rather with a description of a desired function or behavior… The objective is to produce a description of a system that will exhibit the desired behavior."* [18]

Synthesis is concerned with finding a valid design that meets the desired goals as closely as possible without violating any constraints. When at least one of the desired goals are to be minimized or maximized, the synthesis problem becomes an optimization problem:

an attempt to find a solution that is best (e.g. *strongest, fastest, cheapest, most sensitive*, etc.) for the given situation

In MEMS design, like macro machine design, engineers often are given high-level performance goals and design parameters. A current trend in MEMS design is to replace macro or electronic components in current products, such as inertial sensors and RF elements with MEMS-based components. In cases like these the higher-level device requirements are already stipulated, and a viable MEMS device that matches or exceeds the performance is desired. The engineer may have a strict set of constraints and multiple competing objectives, which makes finding a valid design difficult without assistance from automation.

Furthermore, with appropriate synthesis tools, MEMS design need not be limited to expert MEMS designers. This type of expertise is usually gained through years of experience and requires a strong foundation in several disciplines (mechanical engineering, electrical engineering, solid mechanics, and microfabrication, to name a few). Through the proper encoding of synthesis tools, intelligent MEMS design can be brought to the non-MEMS expert. This aspect of MEMS synthesis could lead to great strides in commercial MEMS development, as it would allow designers from other disciplines to incorporate MEMS components into their products.

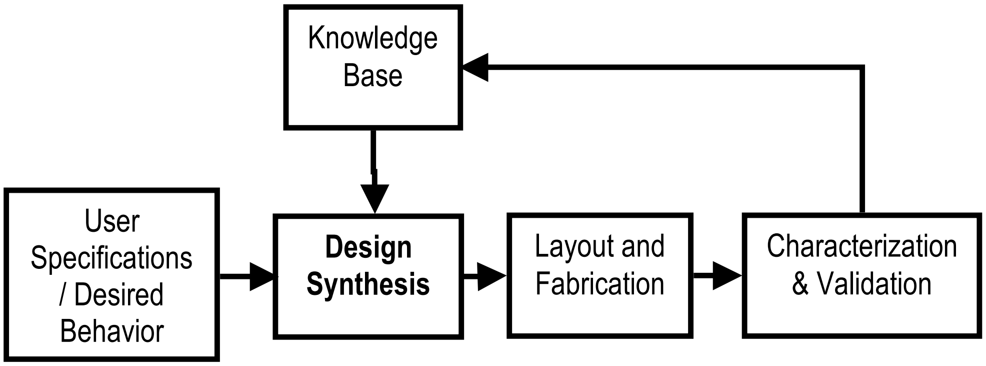

Knowledge Base
User Specifications / Desired Behavior
Design Synthesis
Layout and Fabrication
Characterization & Validation

Figure 1.1: Flowchart of MEMS Synthesis Concept

1.4. Approach to MEMS Synthesis

Our conceptualization of the synthesis of MEMS is more than the generation of topologies to meet specifications; it also includes characterization and validation of our synthesis output and use of a knowledge base to incorporate existing designs, and characterization data taken from previously fabricated designs.

Figure 1.1 shows a flowchart for the entire approach. A detailed view of the design synthesis module, as envisioned in this research is presented in Figure 1.2.

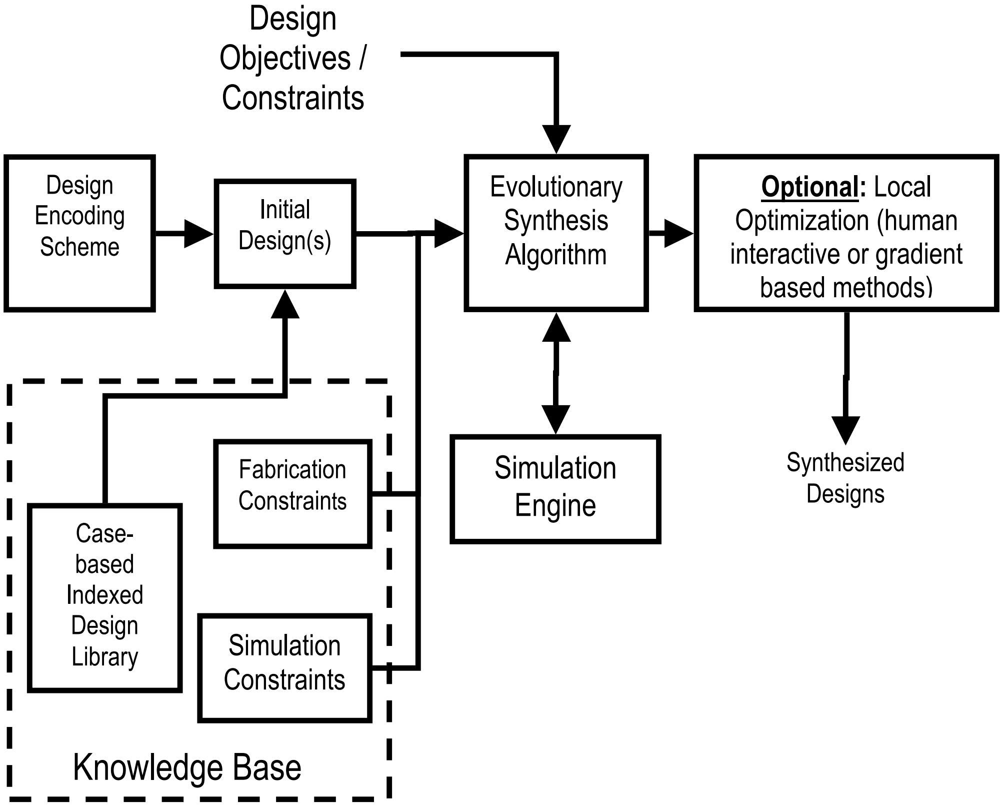

Design Objectives / Constraints
Design Encoding Scheme
Initial Design(s)
Evolutionary Synthesis Algorithm
Optional: Local Optimization (human interactive or gradient based methods)
Synthesized Designs
Simulation Engine
Fabrication Constraints
Case-based Indexed Design Library
Simulation Constraints
Knowledge Base

Figure 1.2: Detailed Flowchart of Synthesis Module. Knowledge base provides information on constraints taken from previous characterization and initial designs drawn from past designs. Fine-tuning of the synthesis output can be performed by gradient based optimization or human interactive evolutionary optimization.

An evolutionary synthesis algorithm such as Zhou's genetic algorithm [19] is at the core of this module. It receives objective goal information from the user as well as constraint information from the knowledge base. These constraints include fabrication oriented constraints (limitations on what types of geometries can be fabricated) as well as simulation directed constraints (limitations of types of geometries and configurations can be simulated). Initial designs can be generated once an appropriate design-encoding scheme is chosen; at that point designs can be randomly generated or drawn from a library of existing designs. The output of the evolutionary synthesis algorithm represents good, valid designs, but fine-tuning post-optimization may be desired. This can be done using either a gradient based local optimization or a human interactive evolutionary optimization.

Many of the individual components presented in

Figure 1.1 and Figure 1.2 have been developed and tested independently, and are presented in this thesis. The next section will discuss the organization of this thesis and where each component can be found.

1.5. Organization of Dissertation

The remainder of this thesis is divided into 9 chapters. Chapter 2 presents details of the design objectives and initial design generation necessary before synthesis can occur. Chapter 3 introduces several synthesis and optimization approaches suited to MEMS design problems. The emphasis is on stochastic methods known as genetic algorithms; this method uses evolution to search the design space. Chapter 4 presents an adjunct to the evolutionary synthesis approach, involving the use of Interactive Evolutionary Computation (IEC), a human interaction means of optimization. This work represents the first application of IEC to microdesign.

Chapters 5 and 6 present synthesis implementations for several practical MEMS applications as a way of demonstrating the applicability of our synthesis approach. Specifically the design of resonators, accelerometers and gyroscopes will be discussed, along with the impact of objectives and constraints on their optimal design. Chapter 7 represents a critical portion of this thesis. We endeavored to “close the loop” between synthesis, simulation and characterization by fabricating output generated by our synthesis tool and comparing the measured performance.

Through this work we identify both strengths and weaknesses in our approach, in chapter 8, we present a discussion of future work in evolutionary synthesis for MEMS. This includes proposed extensions of the human interaction research presented in chapter 4 as well as work currently underway to broaden the scope of the MEMS devices possible using this approach. Lastly chapter 9 is the conclusion of this thesis.

Chapter 2:
Design Objectives and Instantiation for MEMS

This chapter will discuss implementation design objectives and initial design instantiation that must occur before evolutionary synthesis can be performed. Design objectives suited to MEMS applications will be presented as a motivation for the use of multiobjective optimization methods. The application of shape grammars to MEMS device represntation is presented as a means of initial design instantiation. A resonating mass example is presented to illustrate the flexibility available under this approach.

2.1. Design Objectives for MEMS

As synthesis is our goal, the performance objectives used in our methods are critical to the success of the algorithm and the usefulness of the design. For the class of devices we are focusing on, comprised primarily of vibratory suspensions and actuators, common performance numbers that the designer may want to use are resonant frequency, bode plot shape, spring stiffness, actuation voltage, cross axis sensitivity, displacement, and design area.

In cases where we are asked to synthesize for a specific performance goal, the objective function we are minimizing can be expressed as:

$$objective(x) = \left| f_o - f(x) \right|$$

where f_o is the desired goal for performance and $f(x)$ is the performance of the design expressed by the geometry x.

A useful question that should be asked when coding a synthesis program is, “what impact will these objectives have on the geometry of the design produced?” Even if the designer is not an expert in MEMS, the correlation of device geometry on objectives can be predicted in advance, even if it is only qualitative. For example, maximum capacitive sensitivity as an objective for an ADXL-type accelerometer (Figure 2.1). This device is comprised of a suspension and a center mass with multiple parallel capacitive fingers, where capacitive sensitivity can be defined as:

$$Capacitive\_Sensitivity = \frac{\partial C}{\partial x} = \frac{\varepsilon_o N L_o t}{g^2}$$

where N is the number of fingers, L_o is the overlap per finger, t is the out-of-plane thickness of the device(generally a fixed dimension) and g is the gap between fingers.

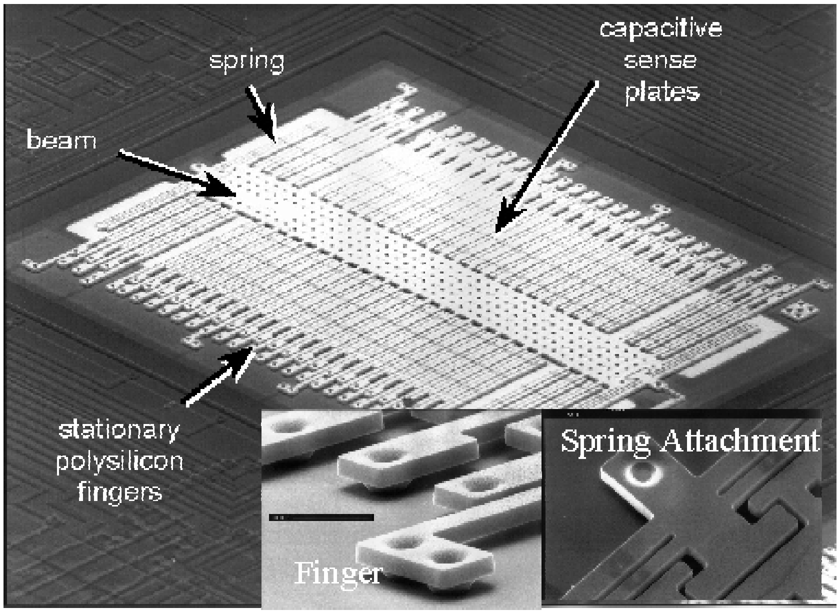


Figure 2.1: SEM of ADXL Accelerometer [20]

Inspection of this capacitive sensitivity relationship shows immediately that maximum capacitive sensitivity would be limited by the constraints of the problem, meaning the best solution would have the maximum number of fingers, the maximum finger overlap length and the minimum gap. In other words, for this objective only, the relationship with the geometry variables is monotonic.

However, MEMS problems rarely are limited to only one objective. More often there are multiple, potentially competing objectives that must be weighed against each other to generate the most suitable design. For example, in the case of the accelerometer, if a second objective calls for a very high resonant frequency, this value will be affected by the mass associated with the sense fingers' geometry, which is critical to the first objective - capacitive sensitivity. Moreover, if the objective is acceleration sensitivity, then not only the capacitive sensitivity but the suspension stiffness also directly impacts this number. This requirement for multiple, competing objectives is one motivation for the use of multiple objective evolutionary synthesis approaches that will be discussed more in chapter 3.

Regardless of the synthesis approach taken, identifying the relationship between objectives in advance can allow the designer to avoid spending computing power on variables and objectives that do not need to be included. Likewise, they can help the designer avoid setting goals and constraints that make synthesizing a viable solution impossible. For example, an accelerometer with extremely high sensitivity and extremely high resonant frequency objectives may be impossible to realize without violating the constraints on minimum feature size dictated by the fabrication process selected.

2.2. Instantiation for Synthesis - Device Grammars for MEMS

The evolutionary synthesis and optimizations algorithms presented in this thesis all share a similar approach – they start with one or more initial design and then modify that design or designs as they search the design space for suitable solutions. Initial designs can either be randomly generated or drawn from pre-existing configurations.

Generating a valid initial design and encoding its parameters into a data structure and then modifying that data is essential to all approaches. The concept of ‘shape grammars’ is particularly suitable to these tasks.

Stiny first defined shape grammars for design in 1980 [21]; they can be defined as a language that creates a shape through successive application of shape transformation rules, starting with an initial shape. Cagan also describes shape grammars as the following [18]:

> *Through shape grammars, classes of known artifacts can be recreated, but also through shape grammars, unique and novel solutions can be created. Furthermore within engineering design, spaces of artifacts can be explored through directed search algorithms to drive designs towards characteristics that best meet any given set of design objectives and constraints.*

Shape grammars have been applied to product design and particularly architecture. A shape grammar for Frank Lloyd Wright designed prairie houses was developed by Konig and Eisenberg [22]. Once a set of grammar rules had been identified, Wright's houses could be fully described by these rules, and furthermore, an infinite number of new variations could be produced in the style. It is the generation and modification aspects of shape grammars we are interested in utilizing in this thesis.

Shape grammars are non-deterministic, however, in that there are multiple ways of defining rules for any given class of object and multiple orders in which transformations are applied from the initial to the final shape.

2.2.1. MEMS Resonating Mass Shape Grammars

Agarwal, Cagan and Stiny developed a grammar for MEMS resonators in 2000 [23,24]. Their goal was to create a generative system for valid resonating masses. Their grammar started with an initial shape that included at least one spring and one actuator and then modified the design with a set of transformation rules, a few of which include:

a) add anchor
b) add comb drive
c) add beam segment
d) add folded flexure

This grammar, while allowing creative unconventional configurations in many ways, was also somewhat traditional in that all geometries involved were Manhattan. This MEMS resonator generative grammar was proposed, but never implemented unfortunately.

2.2.2. Resonator Instantiation Example in SUGAR

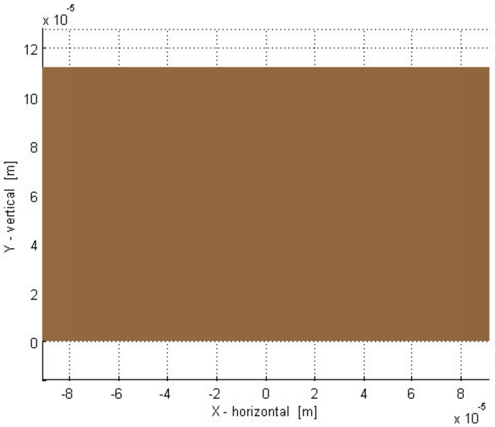
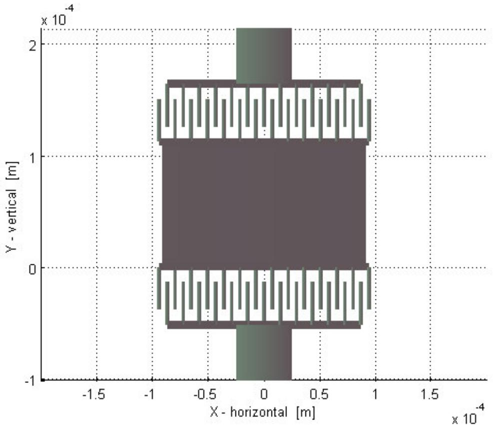
A very basic resonator generation routine was implemented in SUGAR. This function works similar to the Agarwal/Cagan/Stiny grammar to automatically generate a SUGAR netlist with random dimensions and configuration. The function generates a resonating mass in steps as presented in pseudo code in Table 2.1:

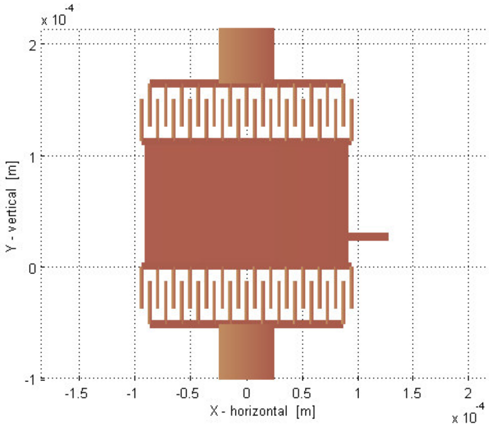
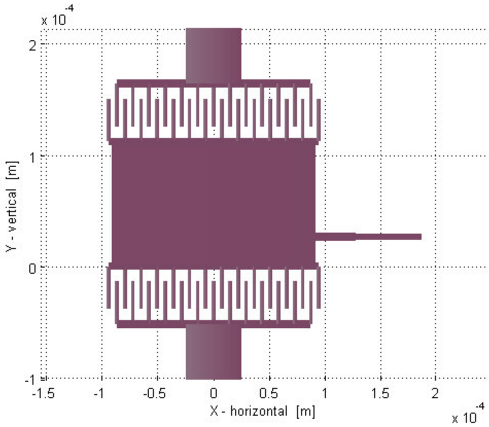
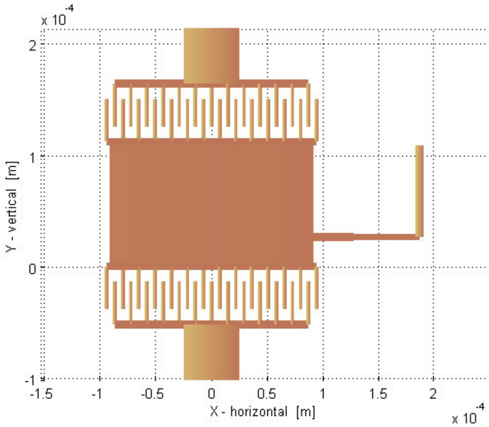
Table 2.1: Pseudo Code Representation Of Resonator Mass Random Generation Function

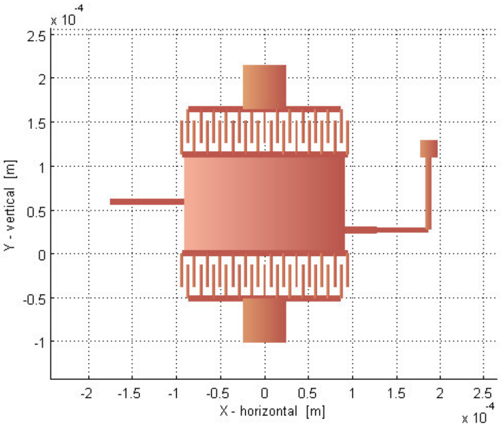
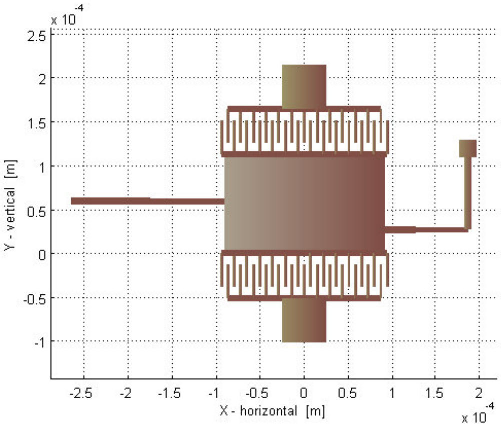
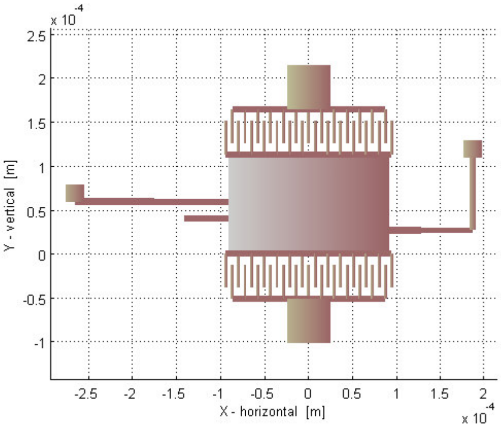
Step	Action	Variables randomly generated
1	Create Center Mass	L_{plate}, W_{plate}
2	Add two comb drives at top and bottom of center mass	$N_{fingers}$, $L_{fingers}$, $W_{fingers}$, g, $L_{finger\_overlap}$
3	Decide how many legs, set k=1	N_{legs}
4:	Add k^{th} leg	
4a	Pick side for leg (left or right)	*ConnectionX(k)* $[+W_{plate}$ or $-W_{plate}]$
4b	Choose location point on plate edge	*ConnectionY(k)*
4c	Decide number of beams for this leg, set i = 1	$N_{beams}(k)$
4d	Generate i^{th} beam segment	$L_{beam}(k,i), W_{beam}(k,i)$, ${}_{beam}(k,i)$
4e	Check for beam crossover violation, if valid, i = i+1	
4f	Add additional beam segments by repeating steps 4c-4e til $i = N_{beams}$	
4g	Terminate leg with anchor, then set k = k+1	
5	Add additional legs by repeating step 4 til $k = N_{legs}$	

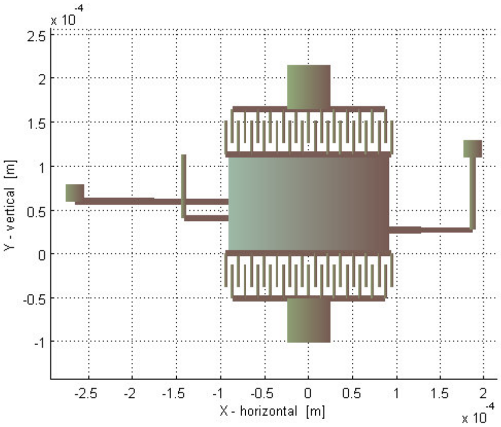
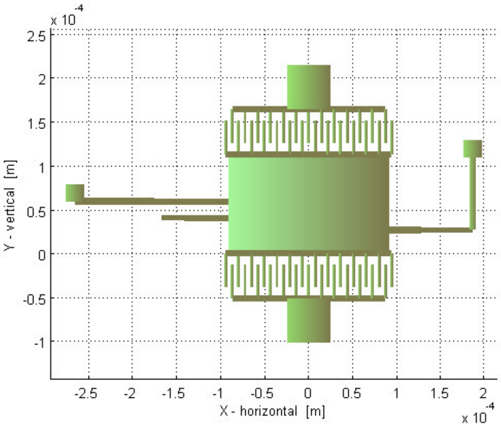
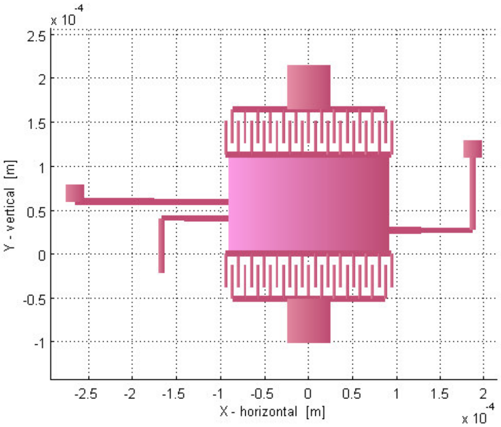
An example of a SUGAR based implementation of this instantiation function is presented in Table 2.2. Of key interest is step 9, where the crossover constraint is violated and the function steps back to the last good configuration and starts over again. This ensures that all shapes generated are valid. Other specifics have been added for the implementation presented here, including Manhattan angles only on beam segments, and constraints on the number of beam segments allowed and their min/max size.

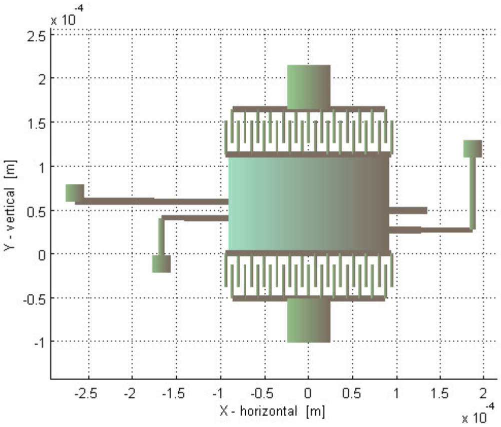
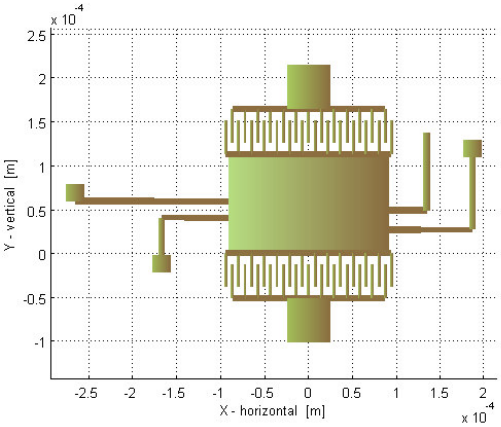
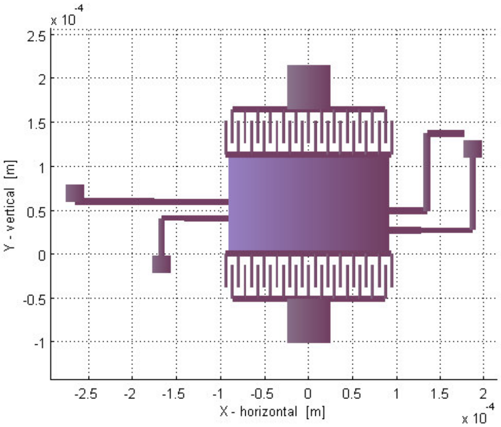
Table 2.2: Step By Step Example Of Four-Leg Resonator With Manhattan Angle Constraints Generated With This Method. Up to 4 beam segments per leg are allowed.

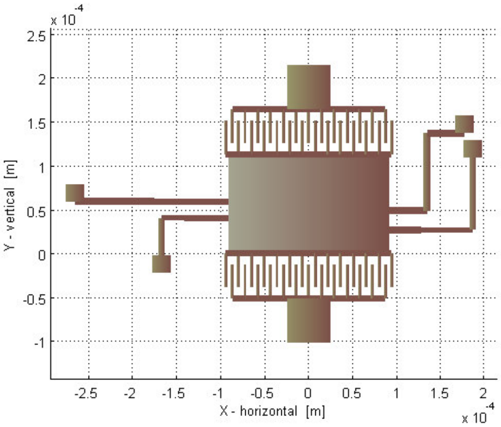
Iteration	Action	Shape generated
1	Generate Center Mass	 x 10 -5 12 10 8 6 4 2 0 Y - vertical [m] -8 -6 -4 -2 0 2 4 6 8 X - horizontal [m] x 10 -5
2	Add comb drives to top and bottom	 x 10 -4 2 1 0 -1 Y - vertical [m] -1.5 -1 -0.5 0 0.5 1 1.5 X - horizontal [m] x 10 -4

3	Generate first beam segment of first leg (in this version we stipulated normal angle on all first segments	 x 10^{-4} 2 1 0 -1 Y - vertical [m] -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^{-4}
4	Add beam to current leg	 x 10^{-4} 2 1 0 -1 Y - vertical [m] -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^{-4}
5	Add beam to current leg	 x 10^{-4} 2 1 0 -1 Y - vertical [m] -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^{-4}

6	Beam limit for this leg reached, add anchor to leg, start next leg	 x 10 -4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 X - horizontal [m] x 10 -4
7	Add beam to current leg	 x 10 -4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10 -4
8	Beam limit for this leg reached, add anchor to leg, start next leg	 x 10 -4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10 -4

9	Add beam to current leg ***** this beam violates crossover rule, must be discarded and replaced *****	 x 10-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10-4
11	Revert to last valid shape	 x 10-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10-4
12	Add beam to current leg	 x 10-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10-4

13	Beam limit for this leg reached, add anchor to leg, start next leg	 x 10^-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^-4
14	Add beam to current leg	 x 10^-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^-4
15	Add beam to current leg	 x 10^-4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10^-4

16	Beam limit for this leg reached, add anchor to leg. Leg number limit reached, design is finished	 x 10 -4 2.5 2 1.5 1 0.5 0 -0.5 -1 Y - vertical [m] -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 X - horizontal [m] x 10 -4

More examples of designs randomly generated by this instantiation routine can be seen in Figure 2.2. Also this method can be easily modified to include other types of suspension elements besides beams. SUGAR allows user defined elements in addition to the standard components. Figure 2.3 shows an example of the instantiation when meandering beams are allowed as well. Note in the case of these meandering beams (defined as a grouping of beam elements), due to the larger real estate occupied by one elemental piece, crossover is much more common and therefore, many more iterations are required to create viable designs. Also, the examples presented here are all asymmetric cases, but symmetric connection points or legs could also be implemented using a similar method. Furthermore, a method could be designed to create more complex geometries (including legs that branch, etc).

The reader should note that the designs presented in this section represent randomly generated topologies that can be used as initial starting points for synthesis, but are not final practical designs. In the interest of generality, we have implemented as few rules and constraints as possible for this example. More restricted instantiations for specific MEMS design applications will be presented in chapters 5 and 6. In those chapters, the design objectives, data structures and design objectives for more complex examples, such as accelerometers and gyroscopes will be presented.

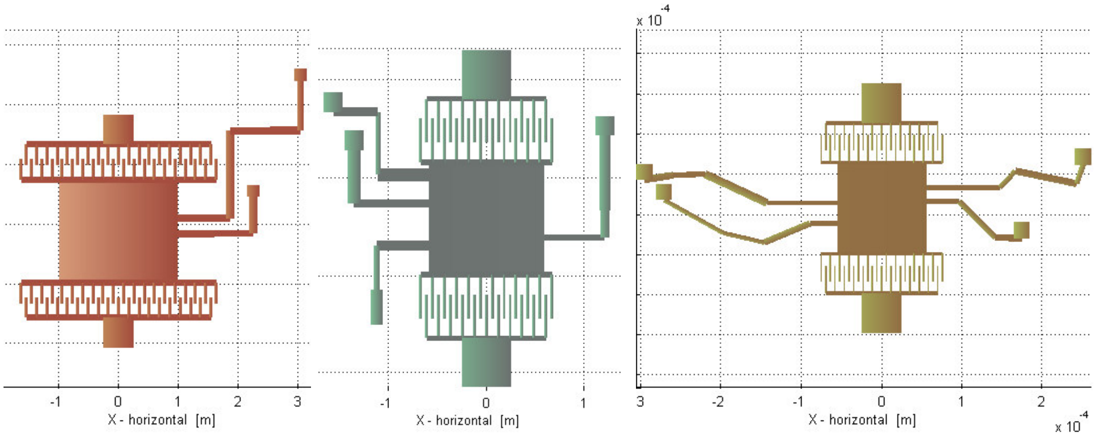


Figure 2.2: Examples from Resonating Mass Random Instantiation Function. Including With and Without Manhattan Angle Constraints

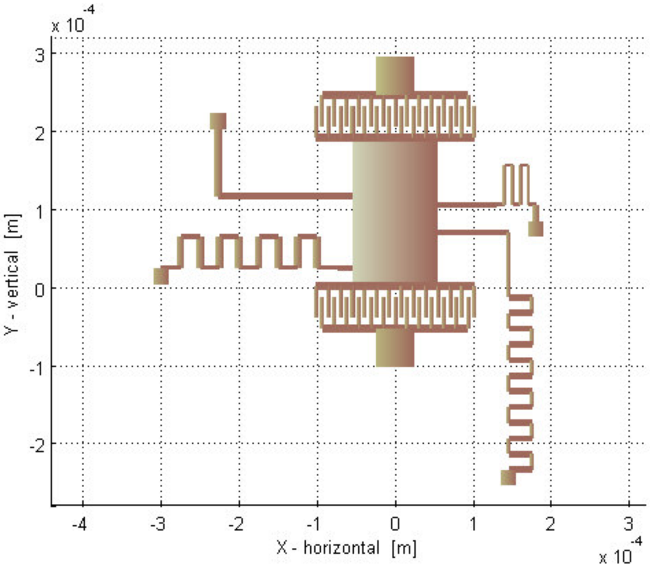


Figure 2.3: Example of Random Instantiation Incorporating More Complex Elements. In this case, meandering spring elements in addition to conventional beam elements. This does not represent an optimum MEMS design, but rather an example of the range of MEMS geometry that can be generated as an initial starting point for synthesis.

While this method allows a designer to generate random MEMS topologies, it does not include a means to manipulate the shape, which is the 2nd part of a shape grammar and critical to synthesis. In the example presented above a netlist, in the form of a text file, is generated that describes the shape (that in turn can be loaded into SUGAR for plotting or simulation). But this does not allow us to go back at a later point in time to modify the parameters of a particular element in that netlist. In order to do this a data structure that can properly encode the design information (both configuration and sizing) and be manipulated by a synthesis routine is needed. A more flexible method currently being developed is described in chapter 8 as part of the future work.

Because of the limitations of the simulation tools available at present, we are limiting ourselves to the realm of surface micromachined electro-mechanical MEMS, but the ideas presented here can be extended to other domains if the ability to predict design performance is available.

Chapter 3:
Synthesis and Optimization of MEMS with Stochastic Methods

Stochastic optimization is the minimization (or maximization) of an objective or multiple objectives through the use of random variation. Methods of stochastic optimization include multistart methods [25] and simulated annealing and genetic algorithms [26]. Each process has its own advantages and disadvantages that will be discussed in more depth in this chapter, but a key feature of most stochastic methods is that they are recursive optimization algorithms that rely on measurements of only the objective function being optimized, not on direct measurements of the gradient of the objective functions. These methods therefore do not require a detailed analytical model directly relating the problem variables with the objective function [27]. Likewise these methods can be effective for complex problems with nonlinearities and a high number of variables and constraints or for problems that contain discrete parameters. In the case of MEMS, discrete variables could include the number of comb drive fingers, number of crenulations on a serpentine spring, etc. Furthermore some stochastic methods are very readily adapted for parallel computing, whereas gradient-based optimization methods are not [28].

For these reasons, stochastic methods are well suited for classes of design synthesis problems as well. By not requiring direct relationships between objective functions and design parameters, stochastic methods are well suited not only to size optimization, where a design’s general layout is set, and only the values of the parameters of that layout are tuned, but also configuration optimization. In the case of MEMS, a simple example of configuration optimization would be a resonating mass. The design may be comprised of actuators, a center mass and a suspension, but the shape of the center mass, or number of beams in the suspension and their connectivity need not be explicitly set in advance.

In this section, we will discuss the stochastic optimization techniques of simulated annealing and genetic algorithms. We will also compare the advantages and disadvantages of the two methods relative to each other. Finally we will briefly review other stochastic methods applicable to MEMS synthesis and optimization.

3.1. Naïve Search, Simulated Annealing

Naïve search, also known as the random walk approach, is a simple search technique where a candidate design is randomly perturbed, if that perturbation increases fitness it is kept, otherwise it is rejected and the previous design is kept. This process can continue until either a viable solution is reached, a certain number of perturbations have occurred and/or no improvements can be found after a certain number of perturbations.

Perturbations can be of fixed or random size. The type of perturbations allowed and the number of parameters affected by each perturbation can be adjusted to suit a particular problem. A key limitation of this random walk approach is that it is only suited for global optimization if the solution space is convex, e.g. – there are no local minima, as this method has no way to prevent getting stuck in the first local minima encountered.

Simulated annealing (SA) is a similar method, except it does incorporate a mechanism to jump out of local minima. SA is similar to a random walk approach, except it is willing to take a 'worse' solution (lower fitness) with a certain probability. As the number of perturbations increases or the proposed solution gets closer to the desired goal, the probability of accepting a lower fitness solution decreases [29].

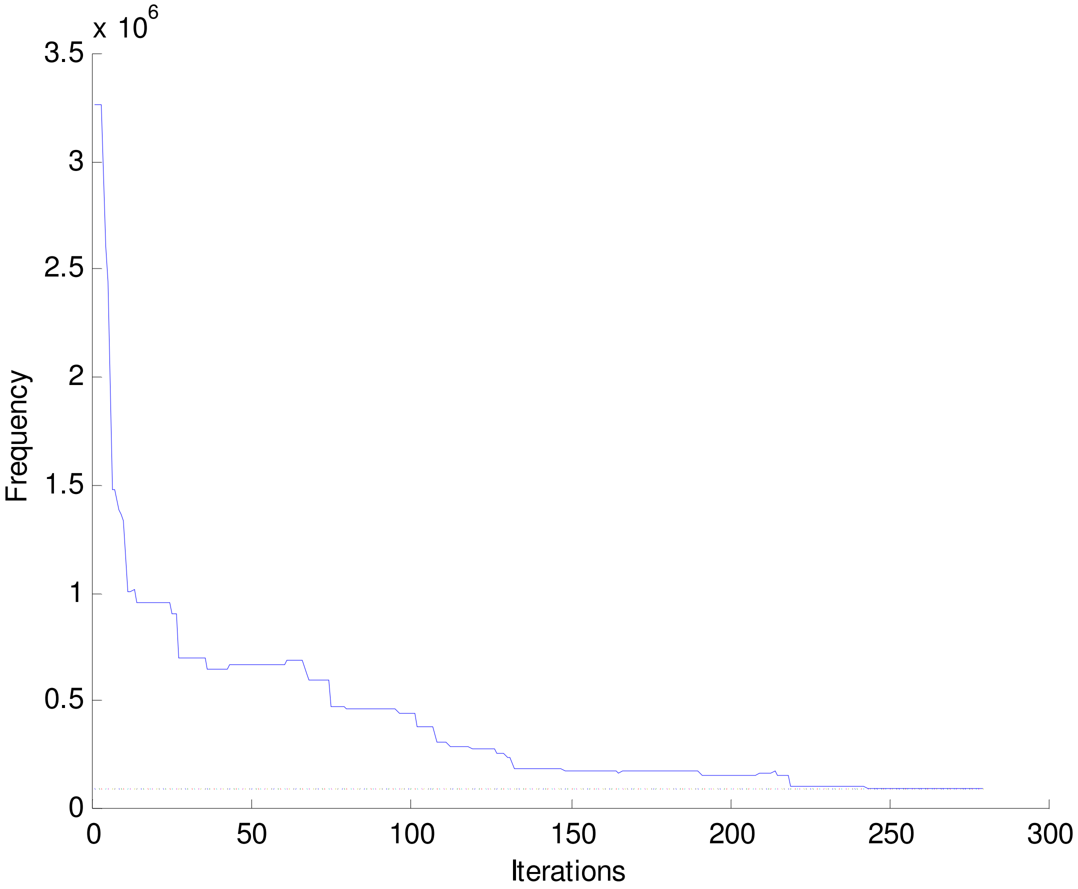


Figure 3.1: Example of Simulated Annealing. The dashed line represents frequency goal value. Frequency error decreases with successive perturbations but several instances where a worse solution is chosen. Stop condition is reached when acceptable solution is found.

As the name implies, simulated annealing exploits an analogy between the way in which a heated metal cools into a minimum energy state and a stochastic optimization algorithm that slowly “lowers the temperature” in stages to eventually “freeze” at the global optimum. SA randomly perturbs a given initial design, whose variations are accepted as the new design with a threshold probability, known as the Metropolis Condition, which decreases as the computation proceeds. The slower the rate of probability decrease, the more likely the algorithm is to find an optimal or near-optimal solution (Figure 3.1, Figure 3.2).

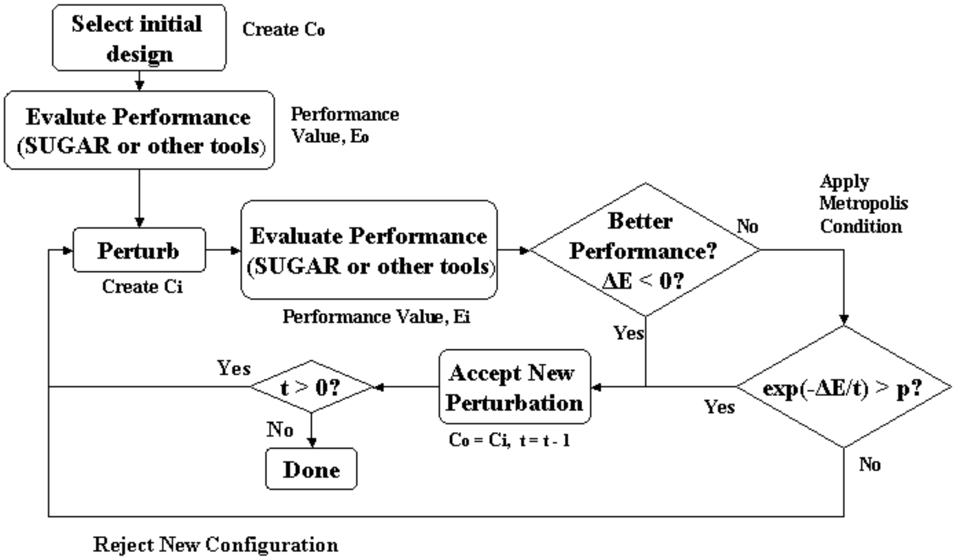


Figure 3.2: Simulated Annealing Synthesis

Both the naïve search method and the simulated annealing method perform an unconstrained optimization for a single fitness function, *F(x).* A multi-objective problem requires a weighted sum of the objective functions:

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x)$$

where n is the number of objectives being optimized, and w_i is the weighting value for the i^{th} objective. The relative weighting of multiple objectives can be problematic. Deriving arbitrary importance ratios between objectives functions can be subjective. Also the minimized solution found could be a compromise solution that does not perform well in any one objective.

Constraints must also be incorporated into the single objective function by adding a penalty function. The penalty function adds a penalty if the constraint is violated, therefore the optimization routine will attempt to minimize the constraint violation. The objective function becomes:

$$F(x) = \sum_{i=1}^{n} w_i f_i(x) + K_1 g_1(x) + K_2 g_2(x) + \ldots + K_m g_m(x)$$

where m is the number of constraints applied, $g_i(x)$ is the constraint function, and K_i is the weighting value for the i^{th} constraint. The K values must be significantly larger than the weighted objective values [30]. Penalty functions can be applied to both equality and inequality constraints, in the case of inequality constraints, $g_i(x)$ is only non zero if the constraint is violated.

3.2. GA / MOGA

Genetic Algorithms (GAs) have become a popular choice for optimization and synthesis because of their ease of use, applicability and global search abilities [28, 31, 32, 33]. GAs are styled after the principles of evolution in the natural world, based on the concepts of genetics and natural selection. A key feature is that very little problem information is required for a GA, other than the ability to encode and alter a proposed design and the ability to evaluate its performance.

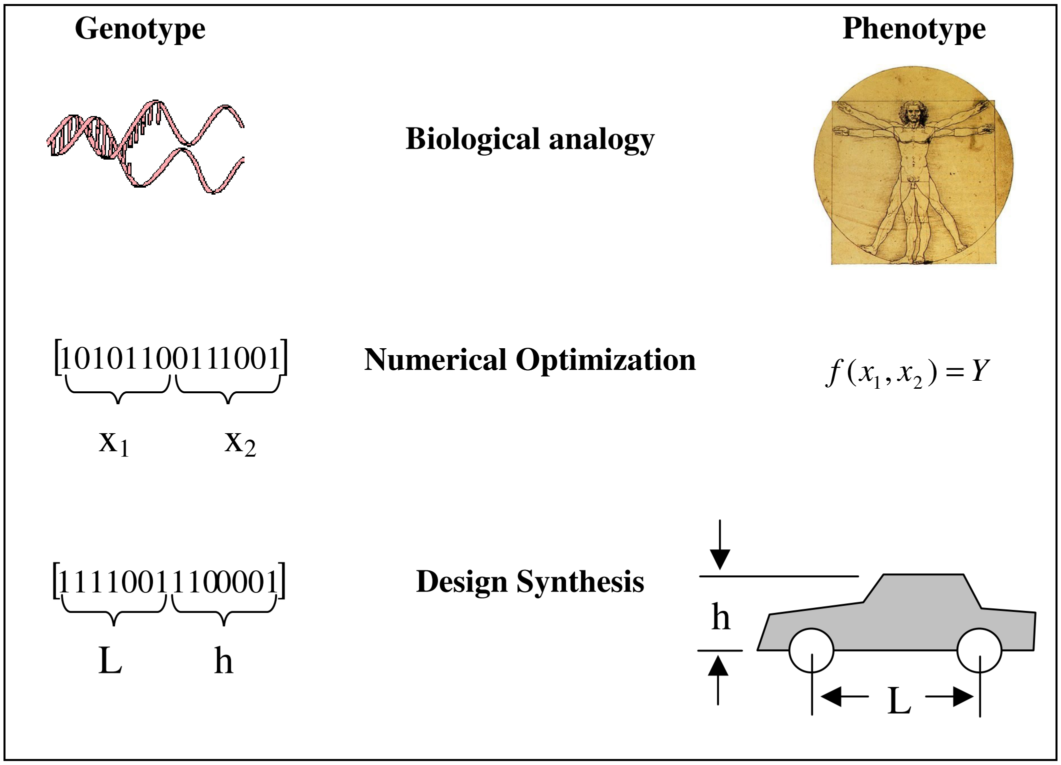


Figure 3.3: Genotype vs. Phenotype

The most basic form of GA relies on binary encoding. An individual is described by a binary string of 1's and 0's, where specific groups of digits of that string represent specific design variables. The binary string is a pseudo-chromosomal representation of a solution akin to a DNA string in biology, where certain portions of the DNA string define specific traits of a human being. The binary string is referred to as the genotype representation and the physical manifestation of those variables is known as the phenotype (Figure 3.3).

The binary string represents design variables for an individual design or solution. When this string is evaluated it gives an objective function value, this objective function value can used to assign fitness to the individual. Fitness can also be modified by constraints, for example a design violating a constraint might be given a poor fitness value regardless of its objective function value.

Figure 3.4 shows the basic flow of a Genetic Algorithm used for optimization or synthesis. Unlike traditional gradient methods or Simulated Annealing, a GA searches the design space with a population of solutions/designs, not just one. Once a population is generated at random, the individual designs (also referred to as individuals in a population) are evaluated and assigned a fitness value. The population is then modified by genetic operations and a new population is created. The cycle repeats itself until a stop condition is reached.

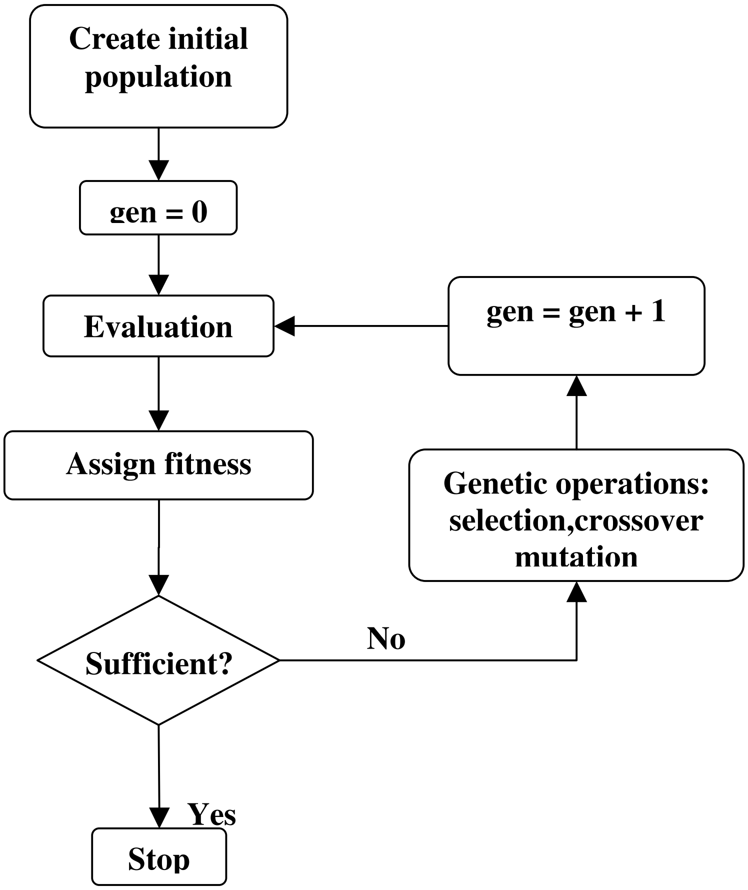

Create initial population
gen = 0
Evaluation
gen = gen + 1
Assign fitness
Genetic operations: selection,crossover mutation
No
Sufficient?
Yes
Stop

Figure 3.4: Simple Flowchart of Basic Genetic Algorithm Implementation. [28].

The design of genetic operations in a GA is a key feature in the success of the optimization output and the speed of convergence. There are many types of operations, inspired by the biological analogy; three of the most common are *selection*, *crossover* and *mutation*. Selection is where copies of individuals are selected for the following generation. There are different types of selection; one is 'roulette wheel selection', where the probability of an individual to be selected for the next generation is proportional to its fitness value. This is analogous to Darwinian 'natural selection', where more fit individuals are more likely to survive (and replace poor fitness designs). Mutation in terms of GA is quite similar to the perturbation in simulated annealing; a design's variables are stochastically changed, resulting in a new design, with at least a slight difference than its antecedent in the genotype. Crossover is where two design's genotypes are cut-and-spliced into two new designs. This follows the biological analogy, where two individuals mate and produce offspring, who are made up of genetic material from both parents. Therefore the original designs are often referred to as 'parents' and the resulting designs as 'children'. Parents can either be randomly selected or selected in a roulette wheel method, the idea being that better parents produce better children.

As with other stochastic techniques, one of the benefits of genetic algorithms is that they only require discrete function evaluations, meaning discrete or discontinuous functions that cannot be optimized by gradient methods can be handled by GAs. Another benefit is that GAs return a population of solutions, rather then a single best solution. This becomes especially important in multi-objective GAs, which we will also discuss in this section. Finally, GAs, unlike SA and gradient-based methods, are well suited for parallel computing. For a given generation, the objective function evaluation for each individual is not dependent on the function call of any other individual; therefore these function calls can be computed in parallel on multiple computers. Once the entire population has been evaluated, it can be ranked and genetic operations applied. Depending on the speed of each objective function evaluation, parallel computing can significantly improve the genetic algorithm's optimization search.

For optimization problems with multiple objectives, the simple GA introduced above could be used by combining several objective functions into a weighted sum, as with simulated annealing. The same issues remain, however, in how best to derive the relative weightings, especially for problems with more then a few competing objectives. One solution to this problem is the use of Pareto ranking to determine fitness.

Pareto ranking comes from the concept of Pareto optimality - for a problem with *n* objective functions, an individual in a population is Pareto optimal if there is no solution that is better than it in all *n* objectives. This Pareto optimal individual would also be referred as non-dominated. The Pareto frontier is the set of all non-dominated solutions. For a two objective problem, the Pareto frontier forms a curve. In a minimization problem the non-dominated individuals would be those with no individuals to the lower left of them when plotted in the solution space (Figure 3.5). In three or more dimensions, the Pareto frontier is an *n*-dimensional surface.

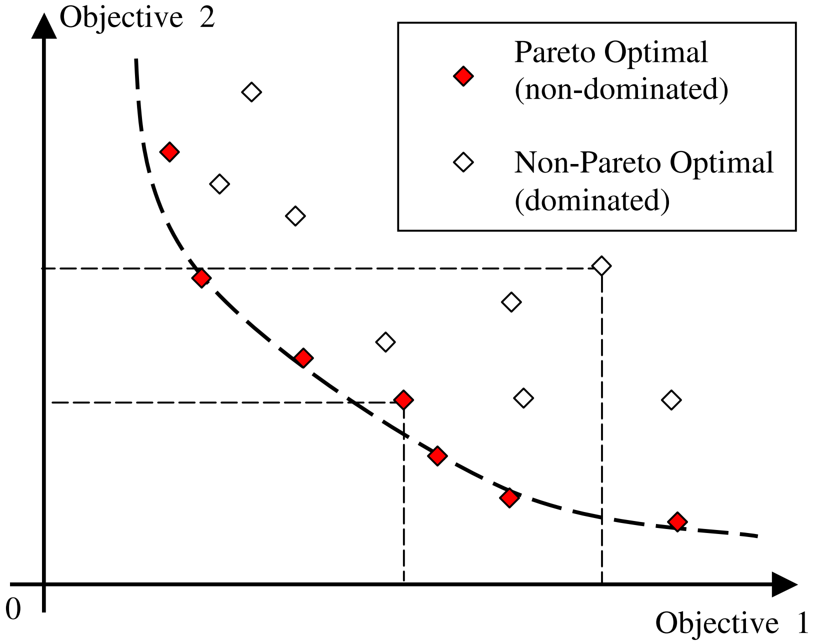


Figure 3.5: Example Pareto Frontier For Two Objective Minimization Problem. Non-dominated individuals have no other individuals in its lower left quadrant.

A set of solutions can be ranked based on Pareto Optimality in a number of ways; Goldberg's method [31] assigns rank 1 to all non-dominated individuals in the population and removes them, then assigns rank 2 to the non-dominated individuals in the modified population, and continues until the entire population has been given a rank. Pareto ranking therefore identifies individuals that excel in one or more objectives, regardless of their performance in other objectives, as well as those that are good compromises in several objectives.

Multi-objective Genetic Algorithms (MOGAs) differ from the single-objective GA presented above in that they assign fitness based on Pareto ranking rather than directly based on the objective function value. Pareto ranking is based upon the individual's performance relative to the rest of the population, and as the best individuals are passed along from generation to generation via the genetic operations, the MOGA will continually push the Pareto frontier forward in search of the best solutions.

A critical issue with GAs is how to encode a design into a phenotype. The binary string is the simplest implementation, where blocks of digits represent binary numbers for each of the design variables. One limitation of this is design variable precision; more precision requires longer strings, which leads to increased computational complexity. Another significant limitation is the "Hamming Cliff", for some binary numbers, an increment to the neighboring number requires the alteration of many bits: e.g. 01111111 = 127, and 10000000 = 128; while the phenotype is very close, the genotype is not [34]. The cliff represents an artificial impediment to a gradual search through a continuous search space. Another problem with encoding parameters as a binary string is that it requires a fixed string length. This limitation is less an issue in the case of size optimization, where all strings are of equal length, but becomes important when applied to configuration optimization problems, where the number of parameters described in a genotype can change.

One solution is to use real-parameter encoding, for example using a vector of real numbers representing the design variables. This resolves the Hamming Cliff and precision issues. Furthermore, the vector can change length depending on the number of variables for a specific individual. The drawback of using real parameters in encoding is in the lack of ease of implementation; the meaning of crossover and mutation becomes less obvious. Mutation and crossover in terms of a binary string is very close to the biological inspired concept of DNA manipulation (e.g. flipping bits for mutation).

In 2001, Zhou developed a real-parameter encoding scheme and Pareto Rank-based MOGA implementation for evolving MEMS devices [19,35]. For full details on the genetic operations used, the genotype encoding format, etc, please refer to these papers. The work presented in this thesis uses the same fundamental approach and encoding methods for MEMS synthesis.

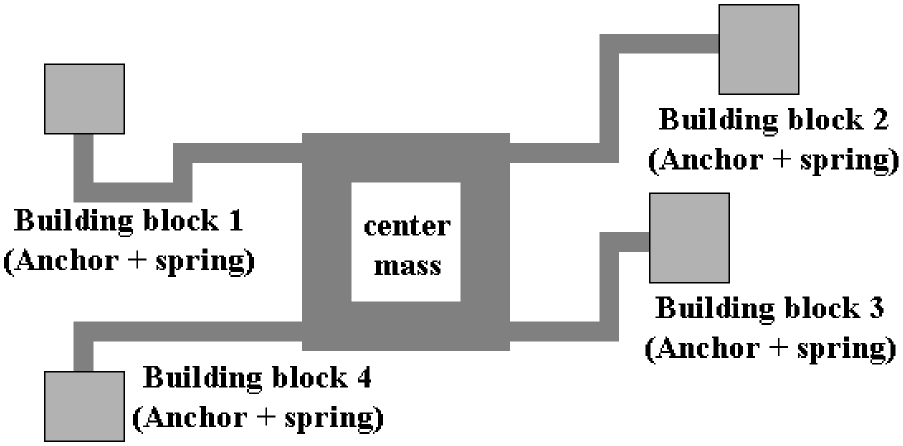
Other work combining GA for MEMS has been done by Antonsson – [36,37], but applied specifically to mask generation and process design for MEMS fabrication, to optimize the correspondence between a fabricated design and a goal design shape. Antonsson also chose real-parameter encoding due to its suitability for describing the mask design variables. At a higher level, Fan has worked on using system-level synthesis of RF band pass filters made up of MEMS components using genetic programming techniques on bond graphs [38]. This work called upon pre-existing parameterized models of MEMS resonators and used genetic programming to design their connectivity and sizing.

3.3. Comparison between SA and MOGA for MEMS Synthesis

In this section we compare Zhou's GA approach with a simulated annealing method based on the same encoding scheme. We present two design examples: in the first we apply a single objective GA and SA to a meandering resonator design with one objective and in the second we apply MOGA and SA to an electrostatic actuator system with multiple objectives.

3.3.1. Implementation

A meandering resonator synthesis problem, as described by Zhou and Agogino [39] was chosen as the first example to synthesize. It is comprised of a center mass connected to four clusters comprised of a series of beams and an anchor (see Figure 3.6). This design is based on similar configurations modeled and fabricated by Fedder [40]. The design goal for this example is to have the lowest natural frequency at *f* = 93,723 Hz. Maximum and minimum values were set for the beam properties (length, width, angle) and a maximum number of beams per resonator leg were also specified. The center mass properties were fixed.



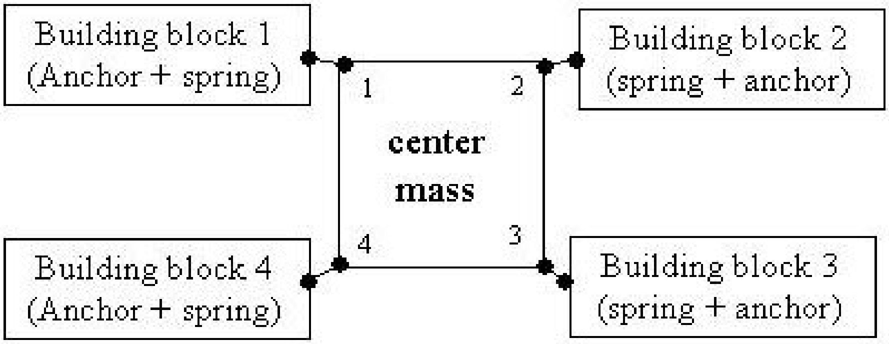


Figure 3.6: Example 1 – Schematic representations of meandering resonator[41]

To better compare with SA, we chose to use a single objective, resonant frequency, therefore single objective GA, (SOGA) was utilized rather than MOGA. Zhou [39] presents full results of the MOGA synthesis for this example with three objectives, resonant frequency, and lateral and vertical stiffness. Table 3.1 shows the designs and constraints used in this example, which with the exception of the parameter 'maximum number of beam's per leg,' is identical to Zhou's original example.

For the SOGA implementation, a population of 400 designs was evolved for 30 generations. The crossover rate was set at 70%, mutation rate at 10%, elitism rate was 5% with new immigrants making up the balance.

For the SA implementation, the probability of changing any single beam parameter was set to 10%, as was the probability of adding or deleting a beam segment from each of the four clusters. The maximum number of iterations was set to 5,000, but a conditional statement can stop the synthesis if the objective error is below an accepted value (frequency error < 100 Hz).

Table 3.1: Design Parameters/Constraints Used for MEMS Resonator Example.

Parameter	**Value**
Center mass dimensions	4 100µm x 20µm beams connected in a square pattern, leg elements join at connection nodes between these four beams
Max number of beams per leg	14
Min number of beams per leg	1
Max beam length	100 µm
Min beam length	10 µm
Max beam width	10 µm
Min beam width	2 µm

To further demonstrate the feasibility of these design tools, SA was also used with more strict constraints, including forcing Manhattan geometry (all beam elements are orthogonal) and forcing symmetry between the clusters (all legs are mirror images of each other).

The second example presented is to synthesize the optimum design for an electrostatic actuator/spring device. This example, also developed originally by Zhou in [19] is inspired by a homework assignment in UC Berkeley's graduate level MEMS Introductory course, EECS 245:

> *Create a device as small as possible that can achieve a displacement of 20 µm at an input voltage of 15V utilizing electrostatic actuation.*

The representation chosen for synthesis is comprised of two clusters, a comb-drive array (where the number of combs, number of fingers per comb, length of fingers, gap between fingers can all be varied) and a serpentine spring (where the beam width, length, and number of crenulations can be varied) (Figure 3.7).

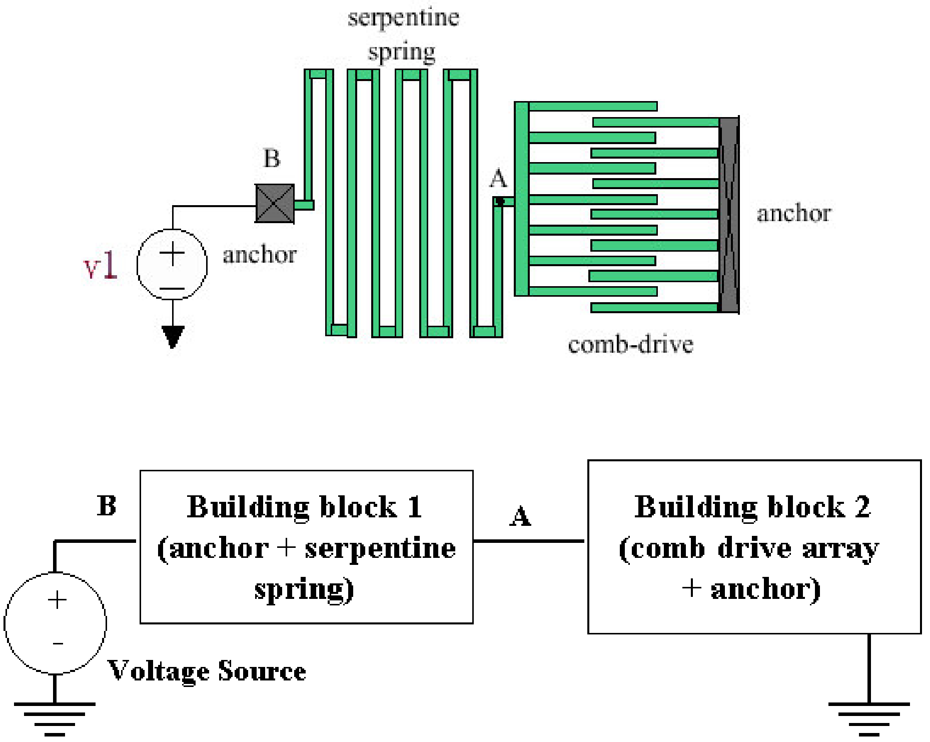


Figure 3.7: Example 2 – Schematic Representations Of Electrostatic Actuator. From [19]

In example 2, the GA implementation has two objectives: 1) achieve a displacement greater then 20 µm at 15V and 2) minimize the overall area of the entire structure. MOGA is well suited to tackle these two competing objectives. The same genetic operation settings were used for this example as the first one - the crossover rate = 70%, mutation rate = 10%, elitism = 5%. A population of 400 was used for 25 generations.

For SA to reach a solution, the minimum required displacement is added as a constraint through the use of a penalty function. If the constraint is violated, a penalty is added to the objective we wish to minimize (the device area in this example). As the SA minimizes the objective function it also minimizes the constraint violation. The objective function is:

$$F(x) = f(x) + kg(x)^2$$

where $f(x)$ is the area of the device, $g(x)$ is:

if displacement > 20 μm, g(x) = 0,

else g(x) = 20 μm - displacement

k is chosen to be large enough so that the penalty function dominates when $g(x)$ is non-zero. In our research we used k values ranging from $1\text{x}10^3$ to $1\text{x}10^5$.

A simple example where a generated design would be invalidated during synthesis would be where a generated spring crossed itself or another component in the design. Figure 3.8 shows examples of invalid configurations that would be rejected.

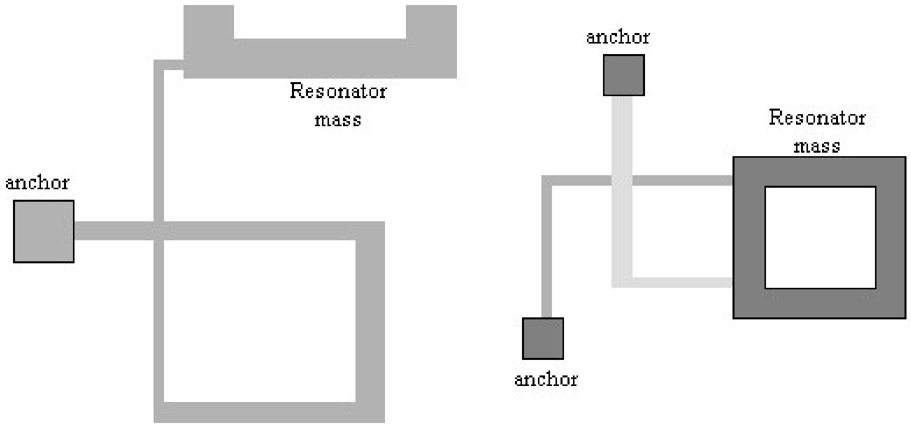
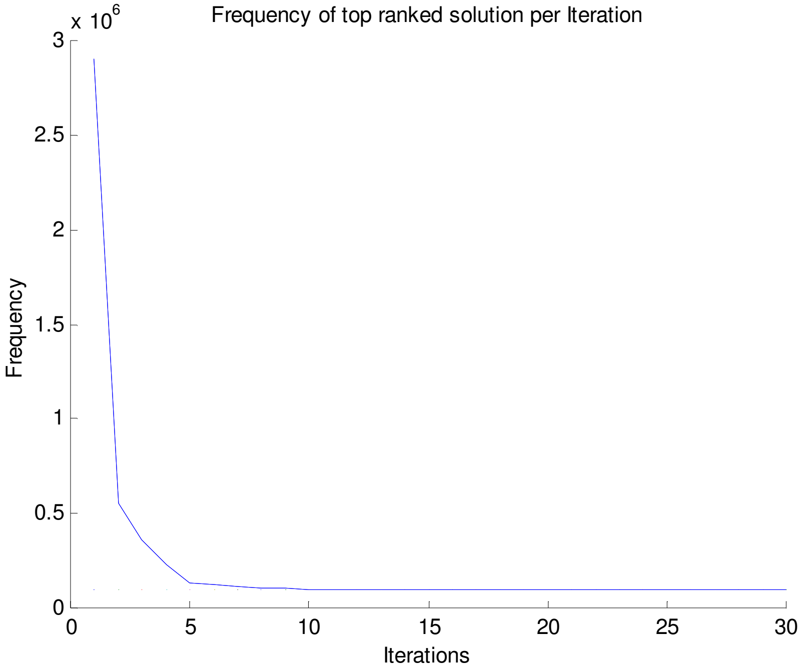


Figure 3.8: Example of Invalid Configurations Of The Resonating Mass Example. Left: Leg crosses itself. Right: Two legs cross each other

All synthesis tests were run on a Pentium III-450 MHz PC, running MATLAB 5.0 and SUGAR 2.0a [15]. The best performance metric available for comparison of GA and SA is execution time. We could compare the number of simulator executions before a solution is reached, this neglects the time spent in the GA on genetic operations, instantiating individuals, etc, a significant source of computation time for relatively simple designs to simulate.

3.3.2. Results

For the meandering resonator example SOGA produced a population of good designs that converged to the objective quickly. Figure 3.9 shows an optimal design, along with a plot of the frequency of the best-ranked design in the population for each generation. The best-ranked design converges to the objective value within 15 generations. The 30-generation run took 4-5 hours on average to complete.



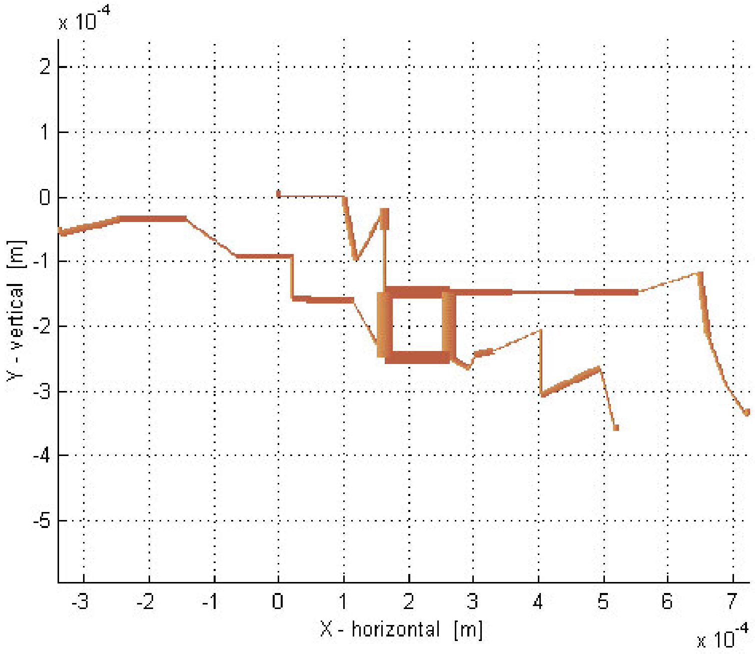
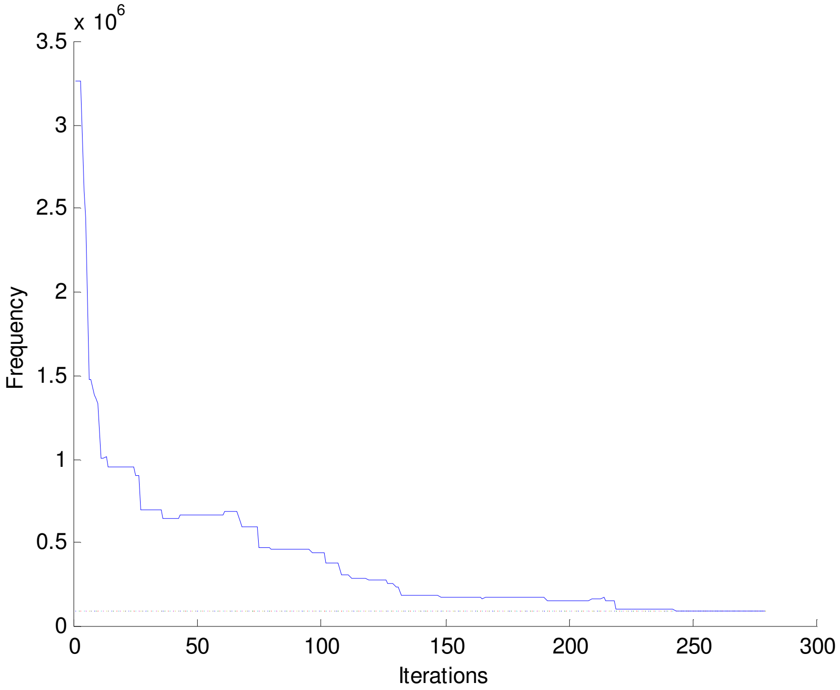


Figure 3.9: SOGA Results For Example 1 – Meandering Resonator. Top: Natural frequency of best solution per generation. Bottom: layout of best-ranked design.



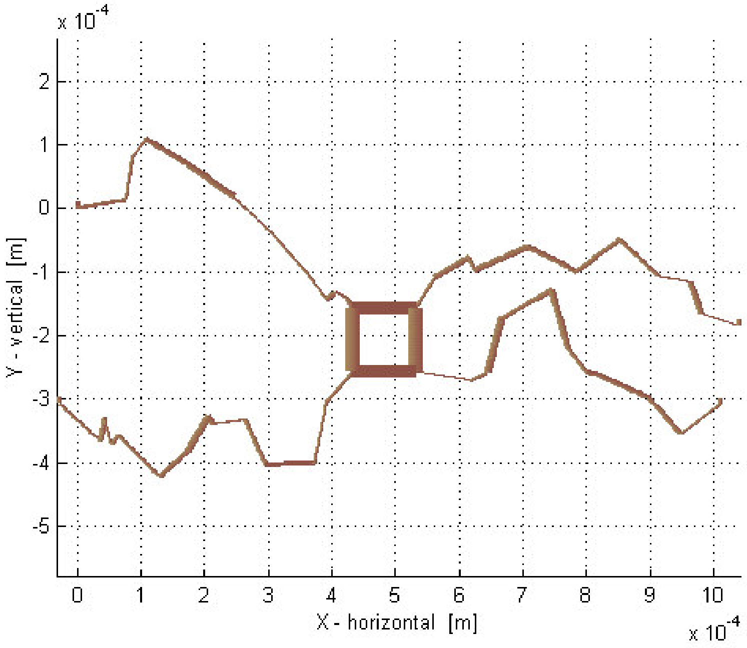


Figure 3.10: SA Results for Example 1 – Meandering Resonator. Top: Solution reached in fewer then 300 iterations. Bottom: layout of final design.

The SA implementation was also able to achieve a good solution quickly. Figure 3.10 shows a solution along with a plot of the resonant frequency per iteration. In this case the solution was reached quickly, only taking 279 iterations (approximately 1 hour). For different starting points, the average time was in the range of 4 to 5 hours. Figure 3.11 shows a solution with forced Manhattan geometry. Figure 3.12 is an example solution with Manhattan geometry as well as forced symmetry between the four legs. This more constrained version reached valid solutions anywhere from 0.5 to 1 hour.

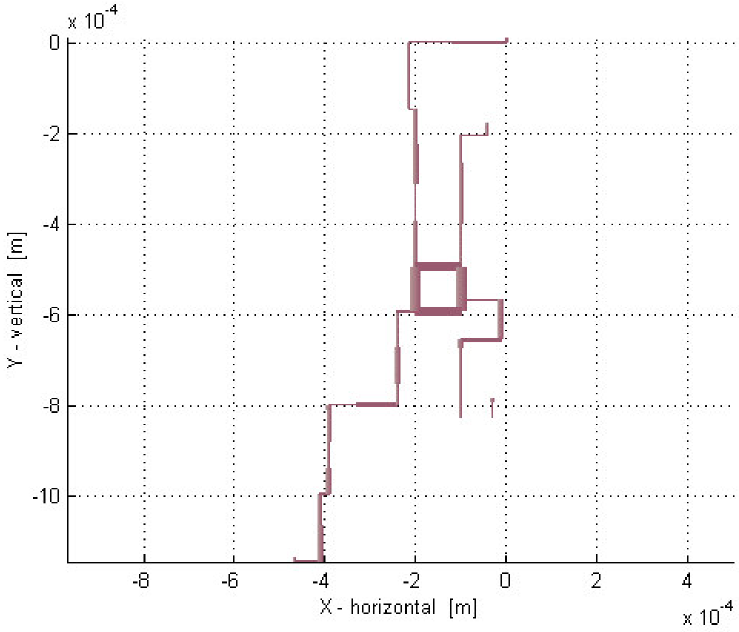


Figure 3.11: SA Results For Modified Example 1 – Meandering Resonator With Manhattan Geometry.

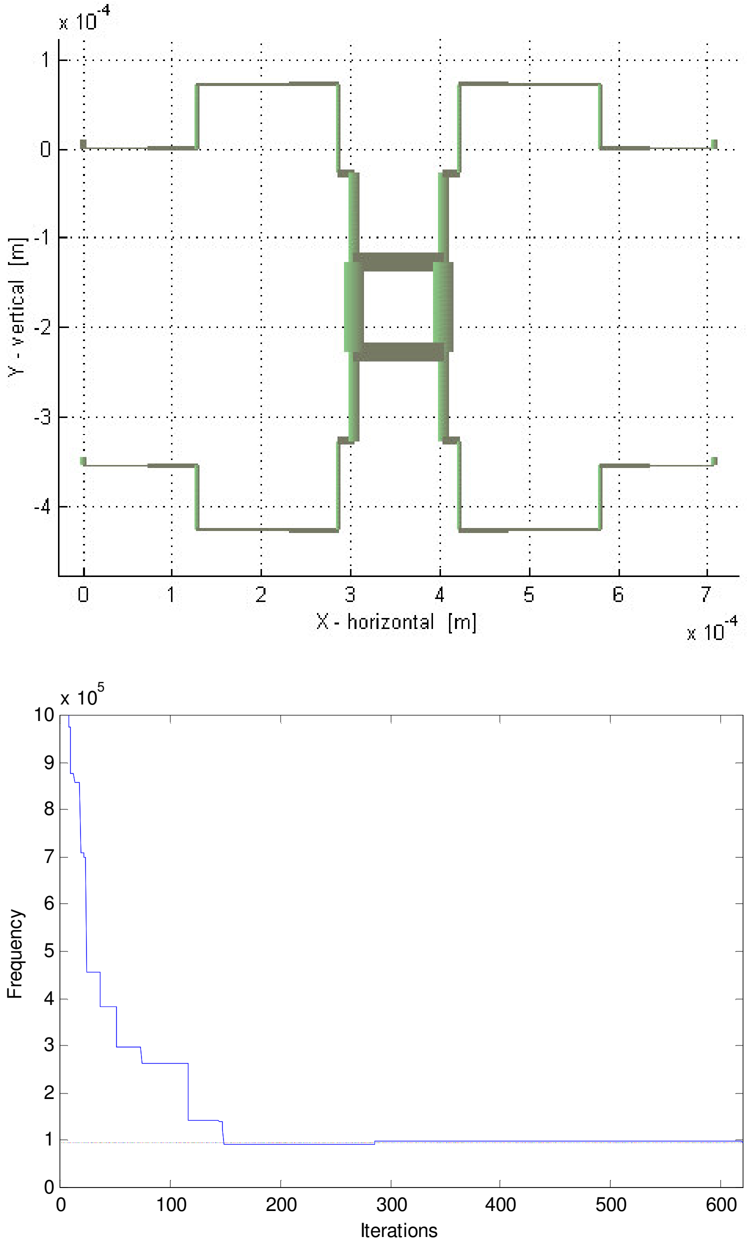


Figure 3.12: SA Results For Modified Example 1 – Top: Meandering resonator with Manhattan geometry and symmetry constraints. Bottom: Solution was reached in 620 iterations.

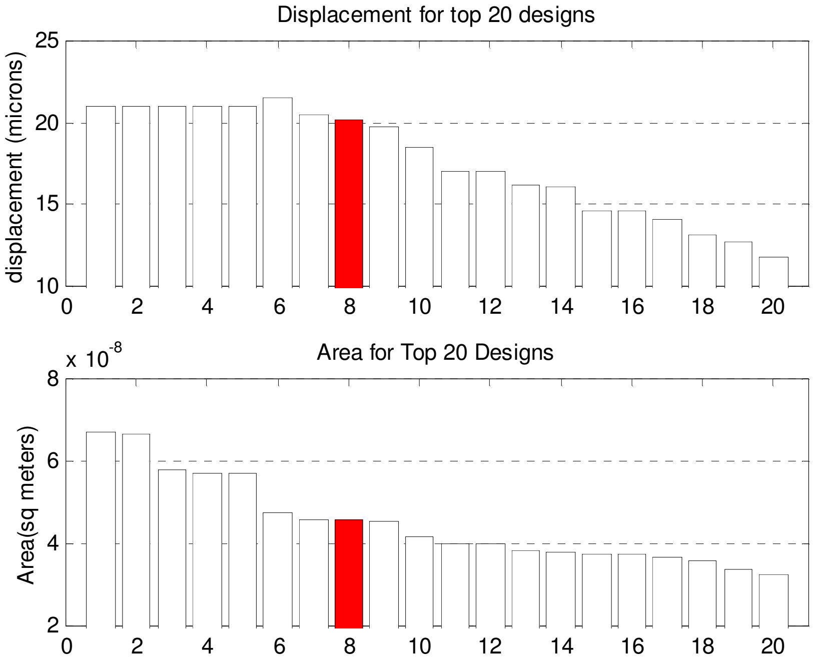


Figure 3.13: Example 2 – Displacement And Device Area Of Top 20 MOGA Solutions. The optimum is highlighted.

Due to the greater computation required in simulating electrostatic actuator forces, the simulation times for the electrostatic actuator example were significantly longer. The MOGA was able to develop a set of solutions within approximately 9 hours on average. Figure 3.13 shows the objective values for the top 20 solutions found. Of these, the top 8 are valid (displacement greater then 20 μm), with number 8 (highlighted) having the smallest area, 4.57e-8 m^2. Figure 3.14 shows the layout of this design. It should be noted that the smallest of the 34 designs developed by the graduate students in the UC Berkeley MEMS class the same year through traditional methods was only 4.90e-8 m^2.

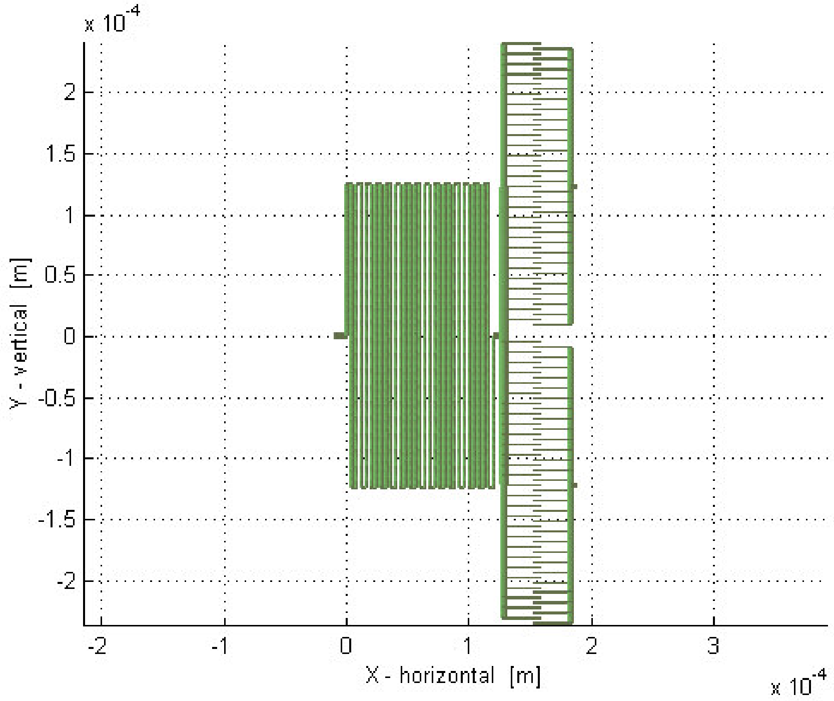


Figure 3.14: Example 2 – Layout of Smallest of the Pareto Optimal Solutions Found by MOGA.

In example 2, SA was not able to reach the globally optimum solution reached by MOGA. After extensive adjustments of the penalty function and the cooling rate, the best SA trial only reached an area of 6.65e-8 m^2 after 7,000 iterations requiring over 18 hours (Figure 3.15 and Figure 3.16). This difference in area of nearly 50% compared to the GA could possibly be overcome with further adjustment of the penalty function weighting and increasing the cooling rate to avoid the SA from getting stuck in a local minimum.

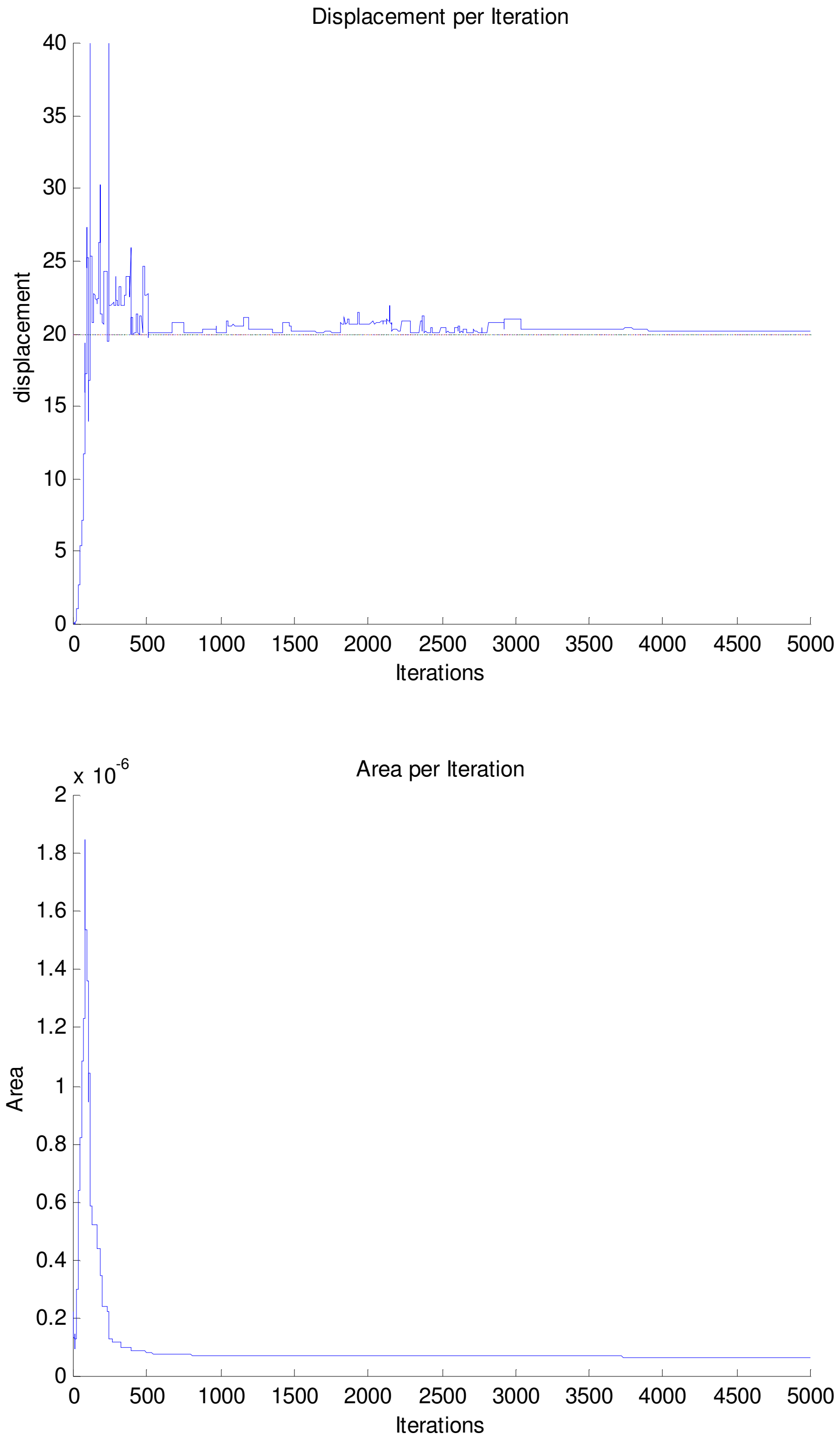


Figure 3.15: SA Results For Example 2 – The Displacement (Top) and Device Area (Bottom) Per Iteration For The Best SA Solution Generated. In the objective function the displacement error was weighted heavily to keep displacement above 20mm.

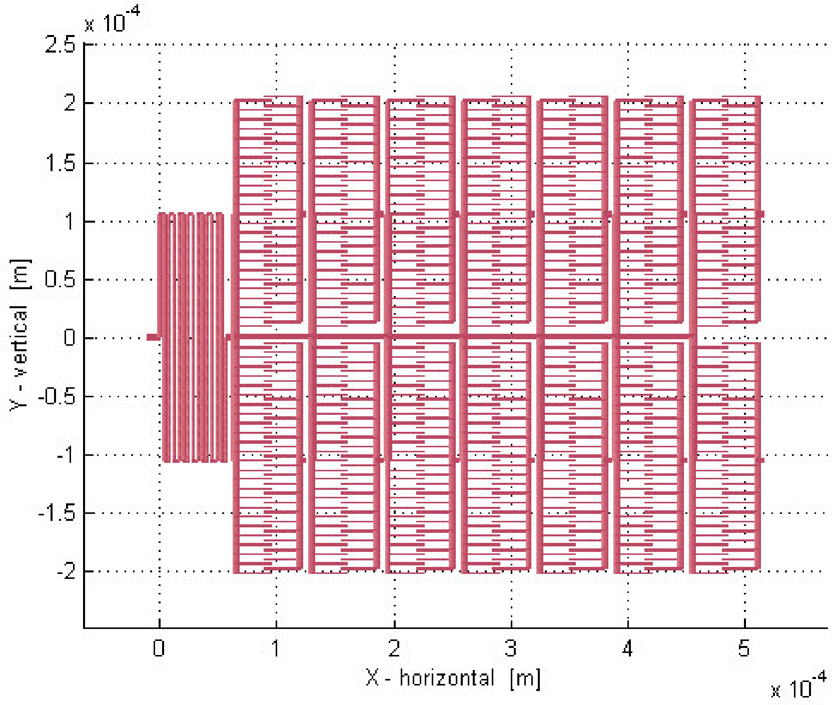


Figure 3.16: Layout of Best Example 2 Solution Found By SA.

3.3.3. Summary of Comparison

The results show that simulated annealing can in some cases synthesize valid designs faster then genetic algorithms. Especially in convex, or nearly convex problems such as the resonating mass example, although in these cases traditional methods may actually be better suited. However SA's dependence on a single objective function and the difficulty in finding the global optimum indicate that it is a less robust method for many MEMS synthesis problems in comparison to genetic algorithms.

3.4. Other Stochastic Methods

There has been a limited amount of work investigating other stochastic methods for MEMS synthesis or optimization. Shea and Vale are currently applying their machine learning-based algorithm known as a "burst algorithm" for multi-objective design synthesis to Zhou’s meandering spring resonator example [42,43]. The burst algorithm perturbs a design in an iterative manner, similar to a naïve search, but it incorporates machine learning as well. It does this by keeping an archive of previously discovered Pareto optimal designs. If a perturbation fails, a design from the archive is called up and perturbed. As this cycle repeats the archive becomes a collection of designs scattered along the Pareto frontier. Their initial work, also using SUGAR, has shown that they are able to achieve comparable results as MOGA with fewer simulations.

Campbell has initiated work in applying a version of the taboo search method (often spelled ‘tabu’ in the literature) to synthesis problems including MEMS design [44,45]. Taboo search is similar to the Burst algorithm, in that it stores previously evaluated designs. Taboo’s unique feature is that it attempts to better traverse the search space by guiding the search away from areas that have been searched already. Campbell’s optimization tool, known as A-design was applied to accelerometer synthesis, but primarily due to a lack of a domain specific formulation, appears to have been unsuccessful for valid MEMS design generation [45].

3.5. Non-stochastic Optimization for MEMS design

3.5.1. Parametric Optimization

Most engineers are familiar with classical non-stochastic optimization methods, such as Newton-Rapheson, steepest descent (gradient), grid search, etc. For direct search methods such as grid search, only the objective function and constraints need to be evaluated, as in the case of the stochastic methods presented in the preceding sections.

Gradient-based methods use the first and/or second-order derivates of the objective function and/or constraints to search. Direct methods tend to require much iteration, whereas gradient methods require derivative information, which may be difficult to calculate, if possible at all, making their use on discontinuous functions problematic. Both methods are generally deterministic, meaning given an initial starting point they will always find the same solution, and tend to get stuck at suboptimal solutions in complex nonconvex spaces.

For these reasons application to MEMS synthesis have been limited to sizing optimization of fixed parametric models [28,46,47,48]. For example, Fedder has developed a tool for generating a folded flexure dog bone resonator [49]. A grid search method was used to find the appropriate suspension and comb drive sizing for a given objective function, in this case, a weighted sum of device area, actuation voltage and oscillation amplitude.

Sedivec [50] developed a unique implementation that combined grid search, steepest decent and random walk in a quasi-stochastic hybrid method. It was applied to synthesizing MEMS resonating masses with a desired resonant frequency, while simultaneously minimizing the effect of fabrication uncertainty (the topic of fabrication uncertainty is discussed further in chapter 7 of this thesis).

3.5.2. Local optimization of GA output

One drawback of genetic algorithms is the difficulty in finding the globally optimum solution in a practical amount of time (it can find good solutions, but not optimal solutions). As mentioned above, gradient-based methods are well suited for finding a local minimum. This has prompted idea of combining these two methods, finding better solutions in a reasonable amount of time.

In conjunction with the evolutionary synthesis of MEMS work presented in this thesis, Ying Zhang has applied gradient-based optimization to output from our MOGA tool. The genetic algorithm's ability to generate a good, valid topology is exploited and then mated with the gradient-based optimization's ability to best size that topology for the given objectives. A flowchart of this work is shown in Figure 3.17.

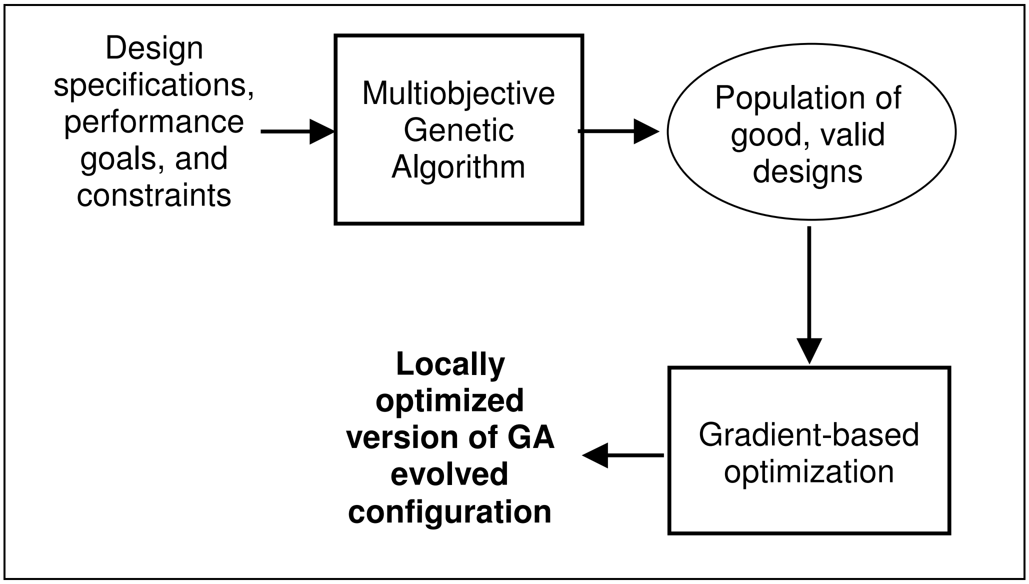

Design specifications, performance goals, and constraints
Multiobjective Genetic Algorithm
Population of good, valid designs
Gradient-based optimization
Locally optimized version of GA evolved configuration

Figure 3.17: Flowchart For Local Optimization Of GA Output

3.5.2.1. Gradient-based local optimization formulation

This local optimization was performed for a two objective resonating mass design problem (see section 5.3). The objectives for the GA were to reach a goal resonant frequency (10kHz) and minimize device area (defined as a bounding box about the center mass and suspension). For the purposes of this gradient optimization, we were interested in taking this configuration and then minimizing the area. The objective function combined both area and frequency:

$$F(x) = A(x) + K(\omega(x) - 10kHz)^2$$

where A(x) is the area of the design (defined by area enclosed by a bounding box around all the beam segments), ω(x) is the resonant frequency of the design and K is a scaling constant. Minimizing this objective function means reducing both the design's area as well as its deviation from 10kHz resonant frequency.

In this case resonant frequency was also applied as an inequality constraint, in order to be valid, the design could not deviate more than 500 Hz from 10kHz. There were also constraints on the minimum and maximum beam dimensions allowed (identical to those in Table 3.1). All starting point designs complied with these constraints at the outset.

The design's general configuration generated by the GA was fixed - the number of beams in a leg and their angle - and only the sizing was adjusted – length and width of each beam segment in the legs. The popular *fmincon* optimization routine in MATLAB was used for this exercise [51]. *Fmincon* can utilize both gradient and direct methods to optimize. Therefore we attempted to estimate gradients with a finite difference calculation, using a 0.1μm step size on design variables.

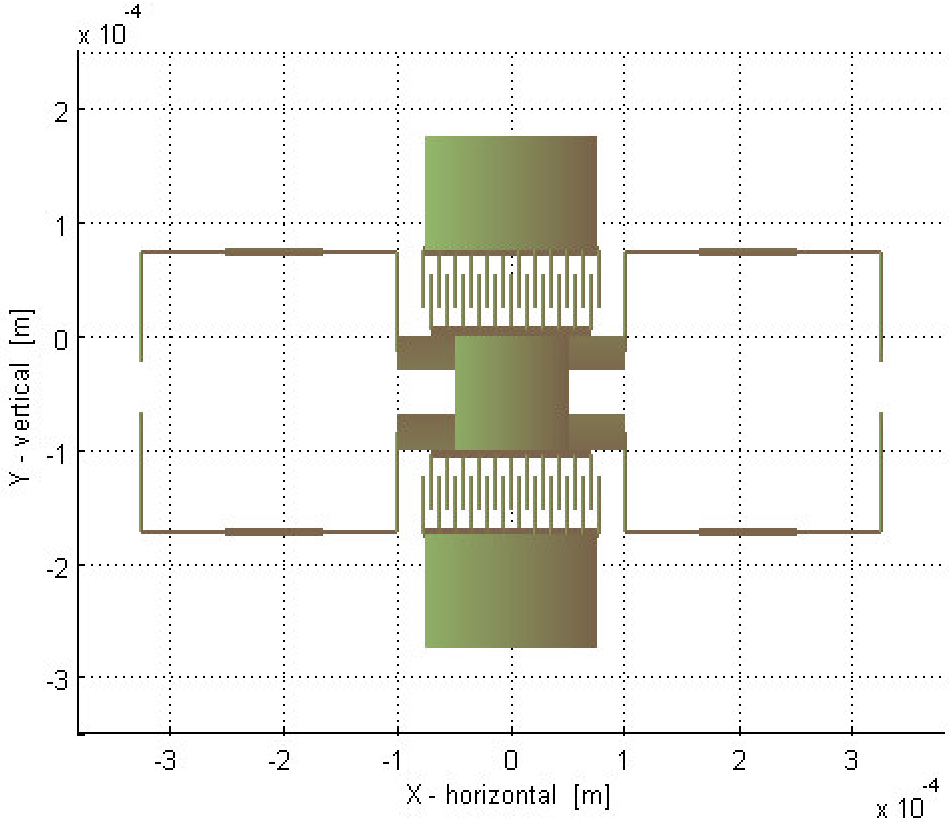


Figure 3.18: MOGA Output Design Used as Initial Configuration for Gradient-Based Optimization. The configuration of this design (number of beams per leg, and angle of beams) is fixed but the sizing of the legs is variable.

3.5.2.2. Gradient-based Optimization Results

The results were encouraging for the test cases using this fine-tuning approach. Figure 3.18 shows a design as generated by the 2-objective MOGA. The resonant frequency of the MOGA output is 10.0 kHz and design area is 16.1 x 10^8 m^2. After gradient optimization (Figure 3.19), the same configuration can be reduced to an area of 14.2 x

10^8 m^2, an area reduction of 12%. This design still maintains a frequency of 10.3 kHz, within the acceptable frequency range.

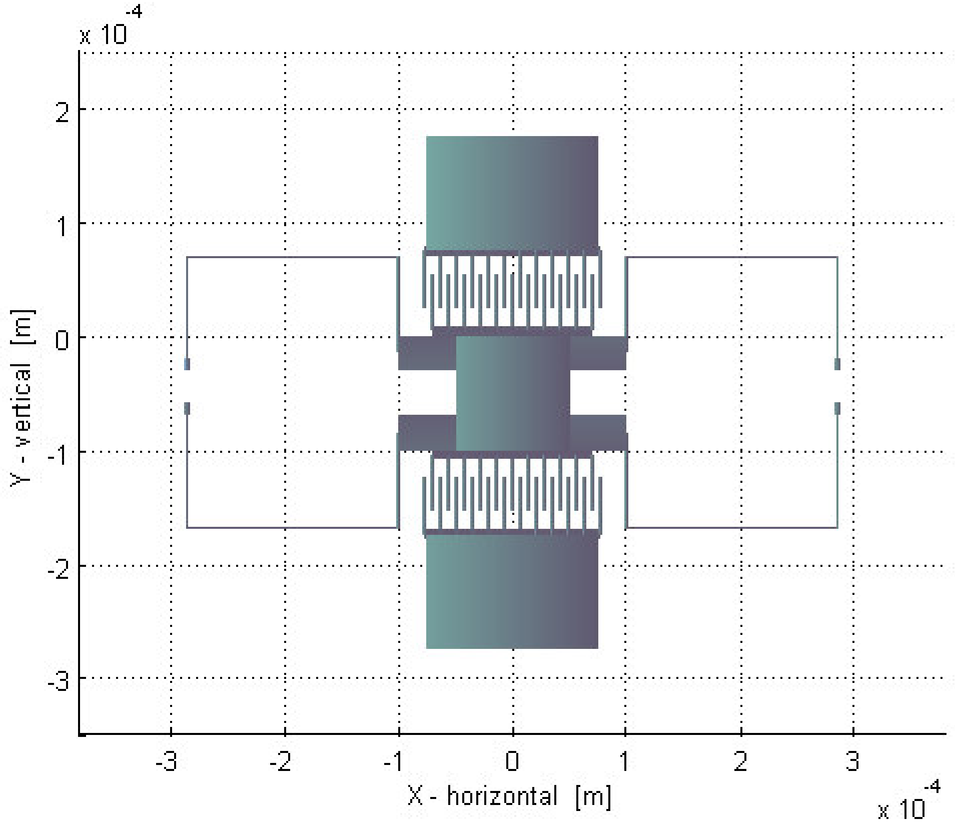


Figure 3.19: Output Of Gradient Optimization. Area is reduced by 12% from initial design.

It is important to note the fact that the frequency did not reach the limit of the inequality constraint (9.5kHz or 10.5kHz). This is due to the fact that the objective function includes frequency error, so *fmincon* also attempts to force the design towards 10.0 kHz, not only reducing area.

The results presented in this example here demonstrate the viability of this hybrid concept of gradient-based fine-tuning of evolutionary synthesis output. It should also be noted that for some MOGA output that we tested, a negligible improvement, in some cases less than 0.5% reduction in area, was found using gradient-based optimization, particularly for output from long MOGA runs (as much as 500 generations). The indirect implication is that our MOGA is capable of finding designs at or very near the optimal size for a given configuration, if given enough evolution time. The impact of evolution time in MOGA is discussed more in depth in section 5.3.

Chapter 4:
MEMS Synthesis and Optimization using Human Interaction Methods

4.1. Introduction to MEMS Synthesis Using Human Interaction

One of the limitations of our current evolutionary approach is that it depends on simulation software to evaluate design quality. As mentioned in section 1.2, there are many design issues that cannot be currently detected by the simulation software, particularly SUGAR, which was chosen over more accurate FEA due to its much greater simulation speed. These issues lead to poor performance or potentially premature failure. While many of these potential problems are clearly visible to a human user instantly, they would be difficult or extremely computationally costly to mathematically model and simulate in a means fast enough to keep iterative design methods such as GA practical.

To address this problem, we combined the MOGA approach described in the previous sections with interactive evolutionary computation (IEC) techniques to embed the human user's visual inspection and domain knowledge into the computer-aided MEMS design process. IEC uses evolutionary optimization to evolve a group of designs similar to GA, but instead relies on a human user to rate each design based on his or her subjective evaluation for performance and shape. This allows the human’s judgment and preferences to further shape which designs are developed, avoiding potential design flaws that are not included in our software and also avoiding regions of design space that the human’s experience tells them is not a fruitful direction of search.

4.2. MEMS Synthesis

The same resonating mass design example used in the standard GA test case is chosen for this IEC demonstration. In this case four objective functions are formulated as a minimization of the distances to four goals: 1) resonant frequency (100 kHz), 2) suspension stiffness in the lateral direction (100 N/m), 3) stiffness in longitudinal direction (1 N/m), and 4) device area (device area goal = 0, i.e. area is minimized). The device area is defined by the area contained within a rectangle bounding the resonator's center mass, comb drives and beams, but not the anchors and contact pads.

Table 4.1: Design Parameters/Constraints Used For MEMS Resonator Synthesis Comparison

Parameter Name	Value
Center mass	5.3066e-011 kg
Leg symmetry constraint:	On
Manhattan angle only constraint	Off
Max number of beams per leg	7
Min number of beams per leg	1
Max beam length	100 μm
Min beam length	10 μm
Max beam width	10 μm
Min beam width	2 μm
Max beam angle	π/2
Min beam angle	-π/2

As in our standard GA implementation, each beam variable has a set of constraints (max/min length, width and angle), and there is a limit on the number of beams per leg

(Table 4.1). The center mass is considered a parameter and not a design variable for this research. Basic geometrical checking is performed to prevent beams from crossing each other as such designs could not be fabricated or operated. For this research, we limit ourselves to symmetric legs only.

4.3. Evolutionary Optimization for MEMS

4.3.1. MOGA Optimization

In order to compare a MOGA-only approach to the MOGA+IEC approach we first solve the MEMS resonator example with four objectives using a standard MOGA implementation with an initial population size of 400, and genetic operators of cross-over and mutation. A non-dominated or Pareto set of approximately 60-100 designs is achieved after 30 generations for this example.

Although these designs are Pareto optimal in the solution space according to the objective performance simulated in SUGAR, they may contain flaws that prevent them from being suitable designs for fabrication. These flaws include potential stress concentrations, ‘*near misses*’ where two legs come very close to each other without actually crossing in the static case, but may collide while the structure is being resonated, as well as other flaws or features that a human engineer may reject based on previous experience or their engineering knowledge. Of the 60-80 designs in a typical Pareto set returned by the MOGA, only 25-30 are within 50% of our most critical objective, resonant frequency. Of these, only approximately four or five designs are free of non-simulated design flaws. The current simulation software has no means to simulate residual stress effects, stress concentrations, transient responses of structures, and other features that a designer might perceive as awkward or difficult to fabricate or use. Therefore, to improve the viability of the synthesis output, an alternative means of evaluating a design is added to the evolution process.

4.3.2. Interactive Evolutionary Computation (IEC)

IEC is a method for optimizing a system using subjective human evaluation as part of the optimization process. It is well suited for optimizing systems whose evaluation criteria are preferential or subjective, such as graphics, music and design, and systems that can be evaluated based on expert's domain knowledge. Fields in which this technology has been applied includes graphic arts and animation, 3-D CG lighting, music, editorial design, industrial design, facial image generation, speech and image processing, hearing aid fitting, virtual reality, media database retrieval, data mining, control and robotics, food industry, geophysics, education, entertainment, social system, and others [52].

The IEC implementation used for this research is a single objective genetic algorithm, similar in structure to the MOGA, except instead of using a simulation tool to gauge the performance of a particular design for multiple objectives, the design is given a single integrated preference score by an IEC human user, and this score is used as the sole objective for ranking design individuals by fitness. In this case, IEC is used to measure the human user’s satisfaction with a particular design based on its shape and quantitative simulator performance. This allows human knowledge and expertise to be embedded into the synthesis process.

4.3.3. Integration

Human fatigue is one of the difficulties faced in implementing an IEC approach [52]. Human interaction is required for every evaluation, thus limiting the population size and number of generations that can be used. The current MOGA requires on the order of 12,000 evaluations (a population size of 400 over 30 generations) to evolve a group of good solutions from randomly generated initial design individuals. This magnitude of evaluation is not feasible for human interaction, therefore randomly generated starting points for IEC is not practical.

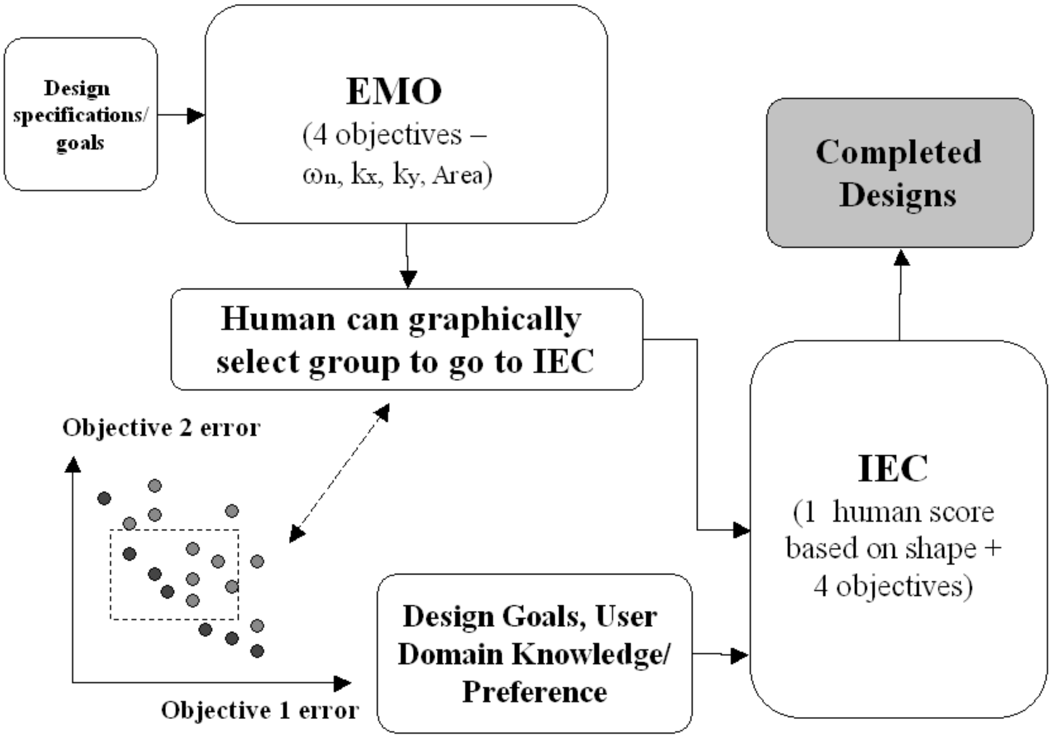
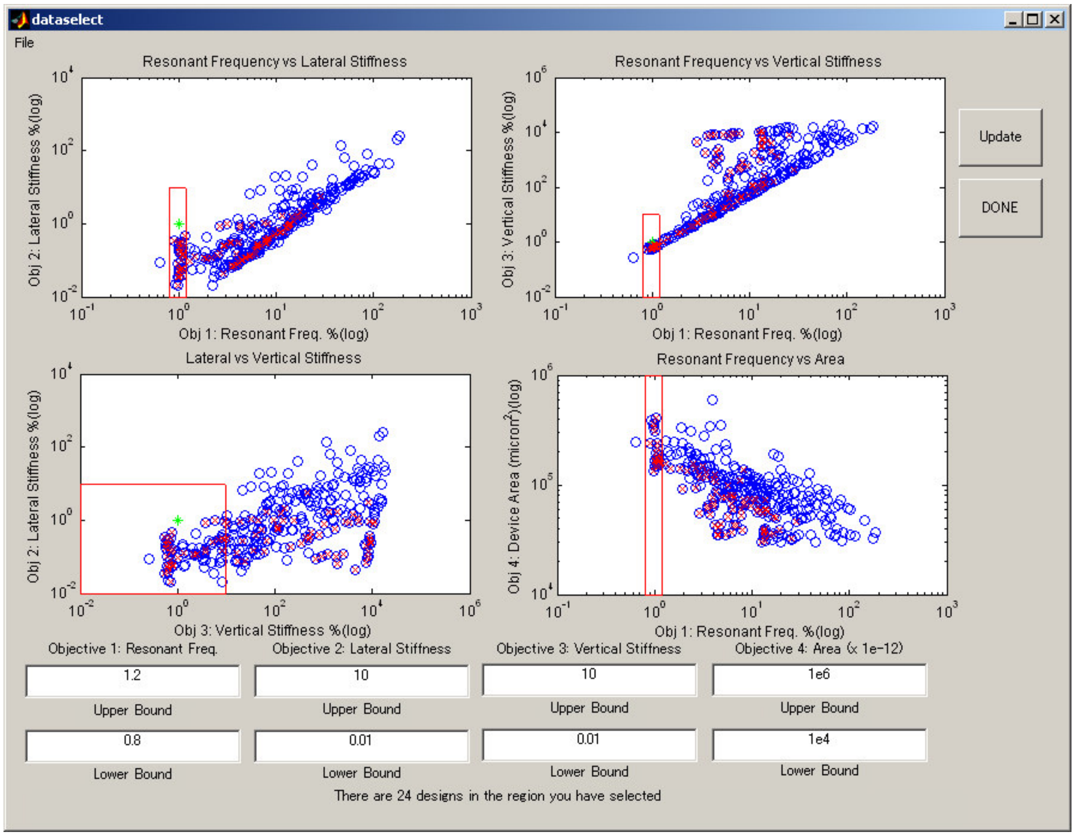

Design specifications/ goals
EMO
(4 objectives – ωn, kx, ky, Area)
Completed Designs
Human can graphically select group to go to IEC
Objective 2 error
Objective 1 error
IEC
(1 human score based on shape + 4 objectives)
Design Goals, User Domain Knowledge/ Preference

Figure 4.1: Integration of MOGA System and IEC System in this Experiment

One solution to this problem is to combine the non-interactive GA and IEC together to use the best attributes of each: the tireless, rapid synthesis of MOGA with the ability of IEC to overcome many design flaws and embed human knowledge. The method of combining MOGA and IEC chosen was serial; MOGA is run, and the individuals of its final evolutionary generation are then used by IEC as an initial population for further interactive evolution. This integration of MOGA and IEC is illustrated in Figure 4.1

As the population produced by the MOGA process is much larger than that of IEC, a manual data selection step between the two components is added. This allows the human user to select only the designs that are contained within a hyper-rectangle in the objective space (Figure 4.2). Although we use a hyper-rectangle in our experiment, the integrated MOGA+IEC concept need not restrict the shape of data selection area.


dataselect
File
Resonant Frequency vs Lateral Stiffness
Obj 2: Lateral Stiffness %(log)
Obj 1: Resonant Freq. %(log)
Resonant Frequency vs Vertical Stiffness
Obj 3: Vertical Stiffness %(log)
Obj 1: Resonant Freq. %(log)
Lateral vs Vertical Stiffness
Obj 2: Lateral Stiffness %(log)
Obj 3: Vertical Stiffness %(log)
Resonant Frequency vs Area
Obj 4: Device Area (micron^2)(log)
Obj 1: Resonant Freq. %(log)
Update
DONE

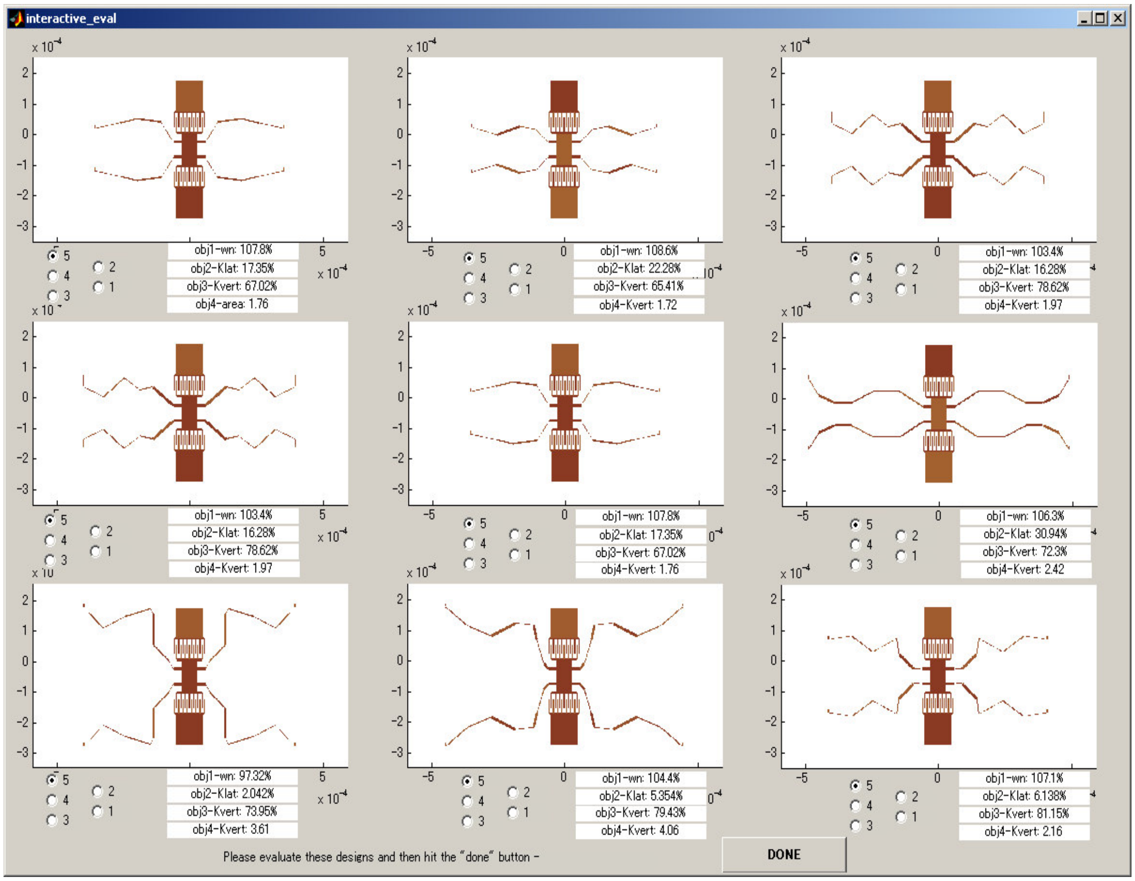
	Objective 1: Resonant Freq.	Objective 2: Lateral Stiffness	Objective 3: Vertical Stiffness	Objective 4: Area (x 1e-12)
Upper Bound	1.2	10	10	1e6
Lower Bound	0.8	0.01	0.01	1e4

There are 24 designs in the region you have selected

Figure 4.2: Data Selection Interface to Select Initial Population for IEC

This step serves to cull the initial population for IEC from a wide array of designs spread around the objective space down to a smaller number of high-ranking designs centered around the objective goals. For example, there is little value in further evolving a design that has a resonant frequency that is an order of magnitude or more away from our goal resonant frequency.

Once a region of the MOGA solution space is selected, the design candidates contained within that space are passed on to the IEC component. These designs serve as a pool from which the initial population is drawn as well as new immigrants in later IEC generations.



● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 107.8%
obj2-Klat: 17.35%
obj3-Kvert: 67.02%
obj4-area: 1.76

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 108.6%
obj2-Klat: 22.28%
obj3-Kvert: 65.41%
obj4-Kvert: 1.72

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 103.4%
obj2-Klat: 16.28%
obj3-Kvert: 78.62%
obj4-Kvert: 1.97

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 103.4%
obj2-Klat: 16.28%
obj3-Kvert: 78.62%
obj4-Kvert: 1.97

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 107.8%
obj2-Klat: 17.35%
obj3-Kvert: 67.02%
obj4-Kvert: 1.76

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 106.3%
obj2-Klat: 30.94%
obj3-Kvert: 72.3%
obj4-Kvert: 2.42

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 97.32%
obj2-Klat: 2.042%
obj3-Kvert: 73.95%
obj4-Kvert: 3.61

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 104.4%
obj2-Klat: 5.354%
obj3-Kvert: 79.43%
obj4-Kvert: 4.06

● 5 ○ 4 ○ 3 ○ 2 ○ 1
obj1-wn: 107.1%
obj2-Klat: 6.138%
obj3-Kvert: 81.15%
obj4-Kvert: 2.16

Please evaluate these designs and then hit the "done" button -

Figure 4.3: Rating Window for IEC: User gives preference score to nine designs at a time based on shape and performance.

For the MEMS resonator example, the IEC interface presents the user with nine designs at a time. Each design is graphically displayed, and its performance simulated by SUGAR is presented as a percentage of the goal for each of the four objectives (Figure 4.3). The users select a preference score from *'1'* (worst) to *'5'* (best) based on their impression of the shape and performance numbers. As a larger population size will give better, more diverse results, the IEC population size can be set to more than nine. For this paper, a population size of twenty-seven was used; therefore three windows of nine designs were presented to the user for each evaluation at evolutional generation.

At the completion of each generation's evaluation, the user preference scores are used by the MOGA to evolve the next generation. This process continues until the user chooses to end the evolution process.

4.4. Experimental Evaluation

To gauge the effectiveness of the MOGA+IEC implementation, a blind evaluation test for MOGA+IEC vs. MOGA-alone was performed. First, the MEMS design with MOGA-only was conducted, and then IEC applied to the MOGA outputs, using the proposed approach. Designs obtained by both approaches were visually evaluated based on users' domain knowledge. Statistical tests were applied to the difference of evaluation to both approaches.

4.4.1. Experimental Setup

Users started with an identical population of 400 resonators with symmetric legs generated with MOGA. They selected a region to serve as the IEC initial population and scored designs for several generations. Users continued until either satisfied with the results or until the tenth generation was reached.

As a rule of thumb, IEC users were instructed to focus their preference scoring on the resonant frequency, the primary objective of the resonator design, and modify that score up or down based on the performance of the other objectives as well as the user's subjective evaluation of the design shape based on their domain knowledge.

A subjective test for comparison by the same user was conducted after each IEC test was completed. The population of the final generation of MOGA+IEC designs was compared side by side to the MOGA-only final Pareto set, using the same interface as the IEC window. The MOGA+IEC and MOGA-only designs were interspersed to ensure unbiased and consistent scoring between the two sets of designs. The users were not informed as to which approach produced any of the designs being evaluated.

Due to the large size of the Pareto design sets – 60 to 80 returned by the MOGA-only step in the process for the 4 objectives – only a subset was shown to the user for scoring in this final subjective comparison with MOGA+IEC. This subset was generated by taking only the members of the Pareto set within 50% of the resonant frequency goal, reducing the number from 60-80 down to a more manageable 25-35. This step removes the irrelevant designs, only requiring a human score on potentially good designs. This can be justified because no user will give a resonator design significantly deviating from the primary objective goal a *'5'* score, and it saves the human user the tedium of scoring dozens of poor performing designs; recall that the Pareto set contains designs that are optimal in one or more objective, but could perform very poorly in the other objectives.

The scores given to each approach were tallied and compared. The quantity of '5' scores given by the users for each approach was chosen as the primary metric for effectiveness as we believed that the number of highly ranked feasible designs best measures the ability of the MOGA+IEC system to further hone designs developed by the MOGA and incorporate design issues that are difficult, if not impossible, to be numerically optimized.

The degree of MEMS experience in the background of the users was also recorded. Although one might argue that experts with MEMS design experience are better suited to rate MEMS design candidates than non-experts, we note that many of the potential design flaws identified can be observed by basic human visual recognition without necessarily requiring extensive training and experience in the field of MEMS. Thus this raises the question of whether non-experts might be trained to recognize good and bad resonator features and perform comparably to experienced MEMS designers in this process.

4.4.2. Experimental Results

Subjective tests and comparisons were performed on eleven engineering graduate students from the University of California at Berkeley. The number of the best scores, *'5'*, given to the designs in the Pareto set of the final generation of the MOGA-only and the MOGA+IEC tests are presented in Table 4.2. Here '+1', '0', and '-1' in the sign column mean that the number of *5*'s in the MOGA+IEC Pareto set is more, equivalent, or less than that of MOGA-only, respectively.

An evaluation of the results in Table 4.2 shows that the MOGA+IEC is significantly better than the MOGA-alone using both the *sign* test [53] ($p < 0.02$), and the *Wilcoxon Matched-Pairs Signed-Ranks* [53] ($p < 0.01$).

Table 4.2: Test Results for the User Evaluations of MEMS Designs with MOGA-only and MOGA+IEC.

		MOGA+IEC	MOGA	
User #	Expert?	# of 5's	# of 5's	sign
1	Y	7	9	-1
2	Y	12	6	1
3	Y	7	3	1
4	N	6	2	1
5	Y	4	4	0
6	Y	11	9	1
7	N	8	7	1
8	Y	1	0	1
9	N	6	3	1
10	N	12	7	1
11	N	9	2	1

Figure 4.4a shows an example of a design in the Pareto set returned by the MOGA that was given a bad score of ‘1’ by one of the users. In this case the user penalized the presence of a sharp corner in the meandering springs due to the potential of the associated stress concentration to lead to premature stress failure. Figure 4.4b shows an example design generated by the same user; the performance numbers predicted by the simulator software are similar, but the user’s interaction has produced a design without potential operational flaws.

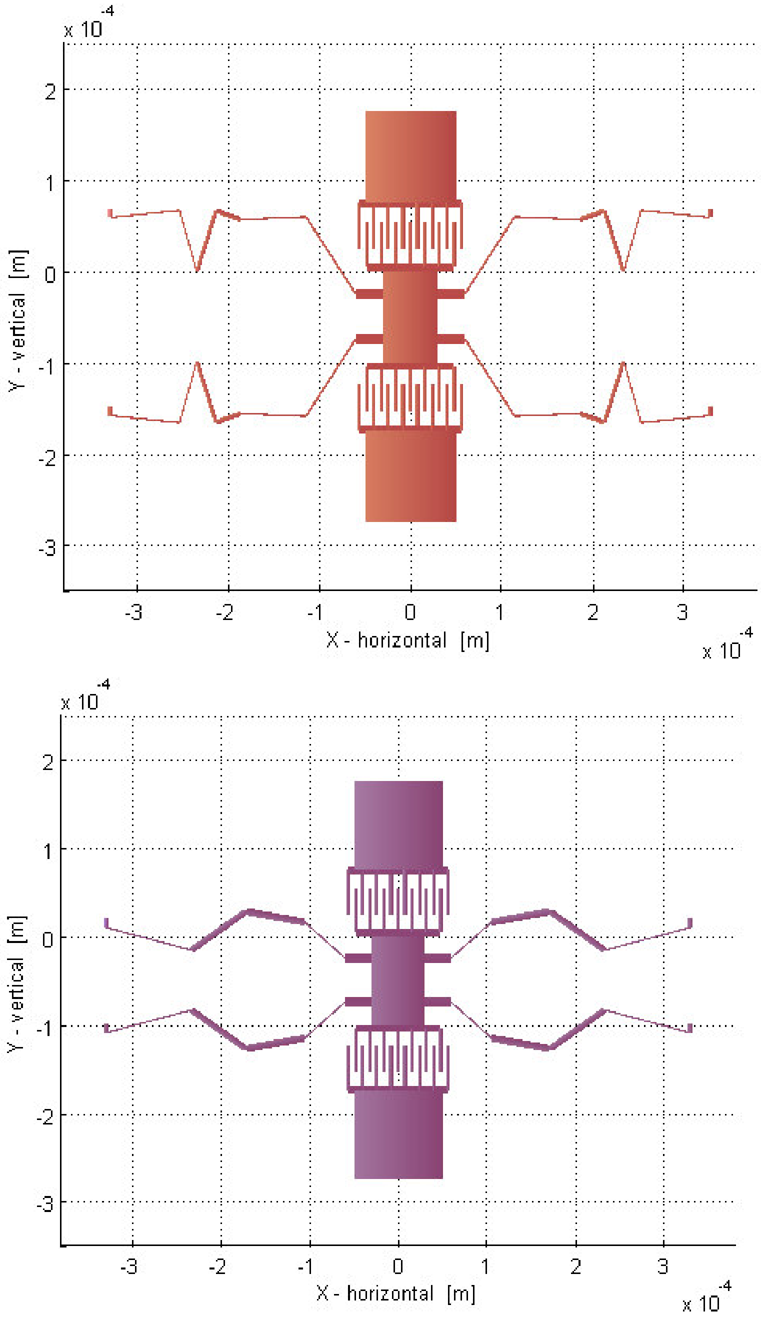


Figure 4.4: (a) Top: MEMS Resonator Design Produced by MOGA. Given a poor score by a user due to potential stress concentrations in the legs. (b) Bottom: High Scoring MOGA+IEC Design Generated by the Same user.

Unfortunately, with only six experts participating, we were not able to draw any statistically relevant conclusions about whether MEMS expertise has any impact on the success of the MOGA+IEC framework. As we refine our work in future experiments, we hope to involve more MEMS experts in the testing.

4.5. Discussion and Conclusions

The results of user testing on the MEMS resonator synthesis example show that using IEC to further evolve designs generated by MOGA can produce better results than those using MOGA alone. Use of MOGA alone does not include critical factors that are difficult and/or costly to simulate or may have been overlooked or omitted from the constraints and objectives. As these factors can have a major impact on the effectiveness of a design when fabricated, MOGA with IEC is able to outperform MOGA alone by incorporating human domain knowledge to produce top ranking designs that are more suitable to the user‘s judgment of well performing designs.

During the course of human user tests, we also made some interesting observations of the impact of human interaction on the synthesis process. It was our observation that humans tended to give low scores to designs that had particular features the user deemed undesirable. For example, as above when the users had a high dislike of sharp corners, they would highly penalize design shapes containing those corners. This causes families of related designs having those features to quickly die off, thus blocking a road of evolution that the user believes is not worth pursuing. The end results show an implicit constraints or penalties concerning the undesirable features. This phenomenon can be used to overcome insufficiencies in the GA formulation, whether due to improper constraint encoding or simulator limitations.

On a much smaller scale we also note that users gave higher scores to designs with ‘interesting’ shapes despite the fact that those designs had poorer performance. This led to those designs propagating into the next generation, being mutated or used for crossover, producing many variants of the ‘interesting’ shapes, eventually yielding designs with the ‘interesting’ features but also better performance according to the simulator. This phenomenon could be described as ‘positive punctuated evolution’; a new shape can quickly flourish and dominate the population if it contains a key feature that the user finds ‘advantageous’. This phenomenon has the potential to accelerate the GA’s search for good designs and helps embed expert knowledge into designs returned by IEC.

Although these observations are not comprehensive enough to make definitive conclusions, it appears that humans are well suited to identifying what they believe to be ‘problem designs’ and ‘killing off’ these designs. In most cases the user made this judgment quickly based on the shape, before even looking at the performance numbers returned by the simulator.

One could describe this human interaction as applying another type of constraint to the evolution process. A human can eliminate designs outside their range of acceptability just as the automated GA presented in the previous section eliminates designs with beams that cross or a resonant frequency in the wrong direction.

Chapter 5:
Evolutionary Synthesis of Resonating Mass

This chapter presents an explanation of the encoding, objectives and constraints used to evolve resonating mass MEMS structures. This type of structure is used as the primary example for evolutionary synthesis throughout this thesis. As shown in Figure 5.1, despite changes in the center mass and leg constraints, the configurations used for different implementations of this example are fundamentally the same – a fixed-size, symmetric center mass connected to four legs, each leg consisting of multiple beam segments connected in a linear chain, each terminating with an anchor. However, specific parameters, such as the number of legs or the center mass configuration are easily modified.

5.1. Encoding

The resonating mass synthesis designs evolved using simulated annealing and genetic algorithms presented in this thesis follow the encoding scheme developed by Zhou [19]. Rather than describe the design based on the coordinates of the intersections between beam segments, Zhou describes the structure as a list of beam dimensions, which is better suited for SUGAR's netlist format. As the center mass is fixed during synthesis, the geometry of the center mass is not encoded in the data structure, but rather stored in the synthesis routine that translates an individual design's leg configuration into a SUGAR netlist (therefore all designs receive the same center mass).

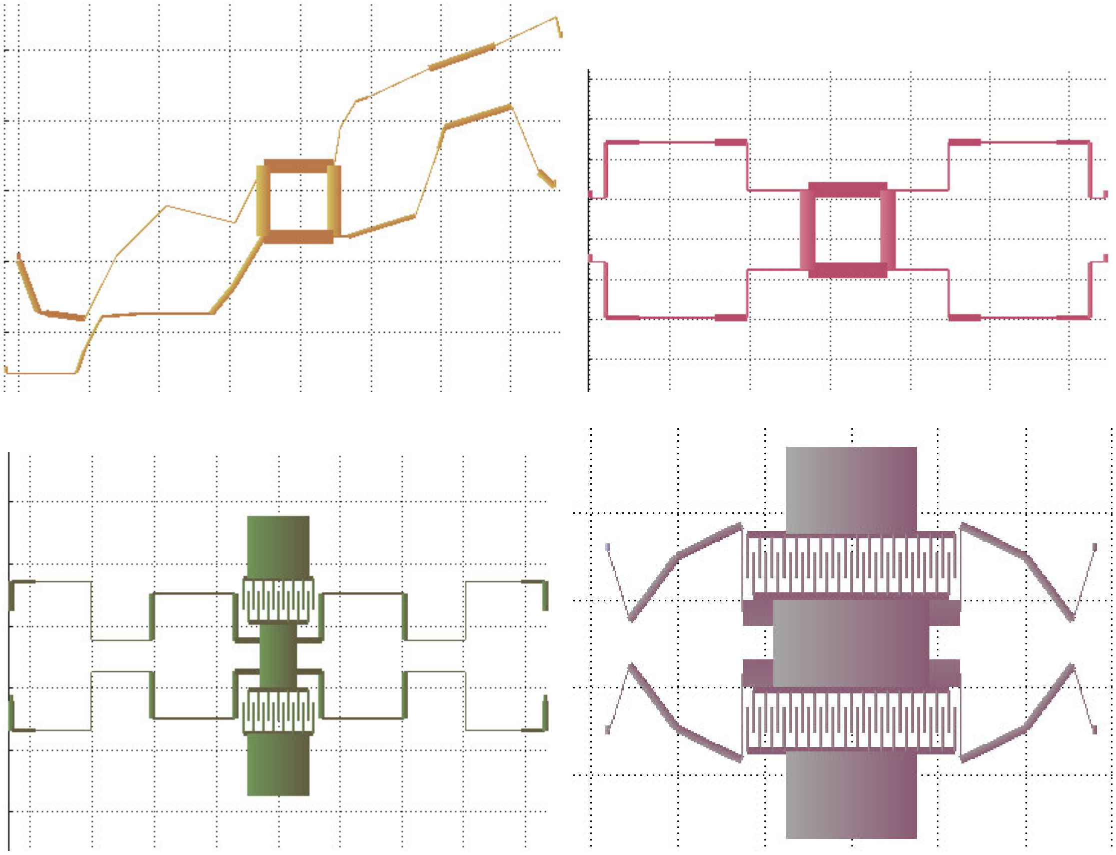
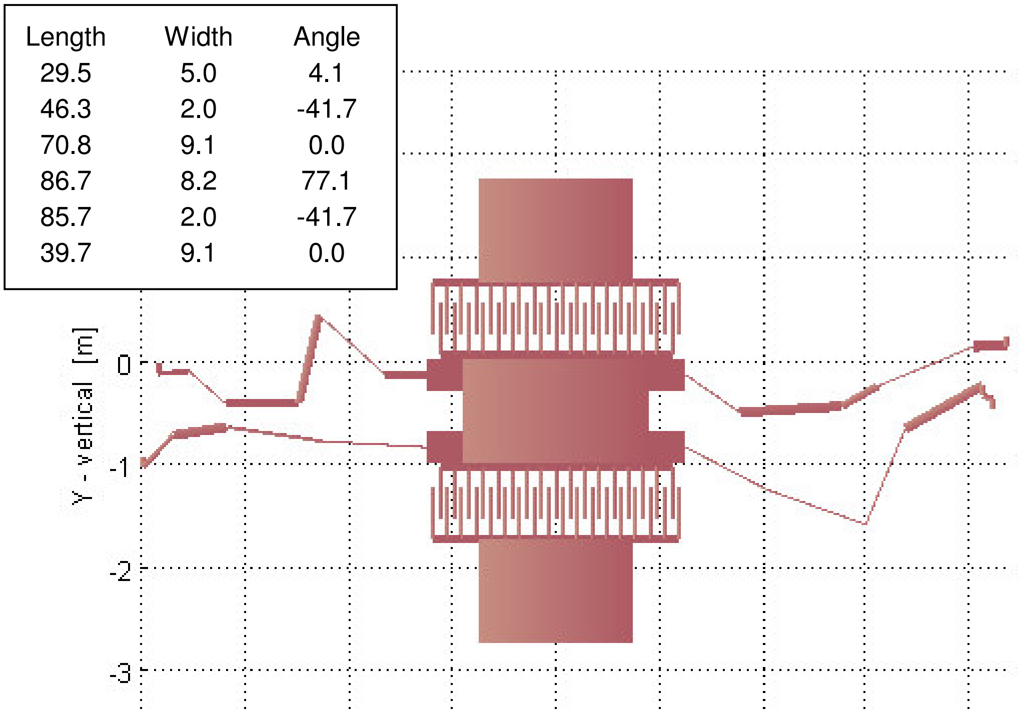


Figure 5.1: Various Examples of Resonating Masses Generated By Different Versions of Our Encoding Scheme and Synthesis Approach.

In the asymmetric leg case, each of the legs is described by an n x 3 matrix, where n is the number of beam segments for that leg. The three columns correspond to the beam segment length, width and rotation angle (in global coordinates). Figure 5.2 shows an example of a design and the matrix describing one of its legs.



Length	Width	Angle
29.5	5.0	4.1
46.3	2.0	-41.7
70.8	9.1	0.0
86.7	8.2	77.1
85.7	2.0	-41.7
39.7	9.1	0.0

Figure 5.2: SUGAR Representation of a Resonating Mass with the Array of Length, Width and Angle Values Representing the Top Left Leg. The first row corresponds with the outmost segment and moves inwards from there.

The genotype for each individual design is comprised of several fields. The leg geometry matrices are included as well as performance and objective function vectors and fitness ranking. In the cases presented here we have chosen four legs per resonator, but this could be any number, the genotype encoding is not limited, although physically there may be practical limits on leg placement around a center mass in the phenotype representation. Each individual is included as an element in an array equal to the size of the population (Figure 5.3).

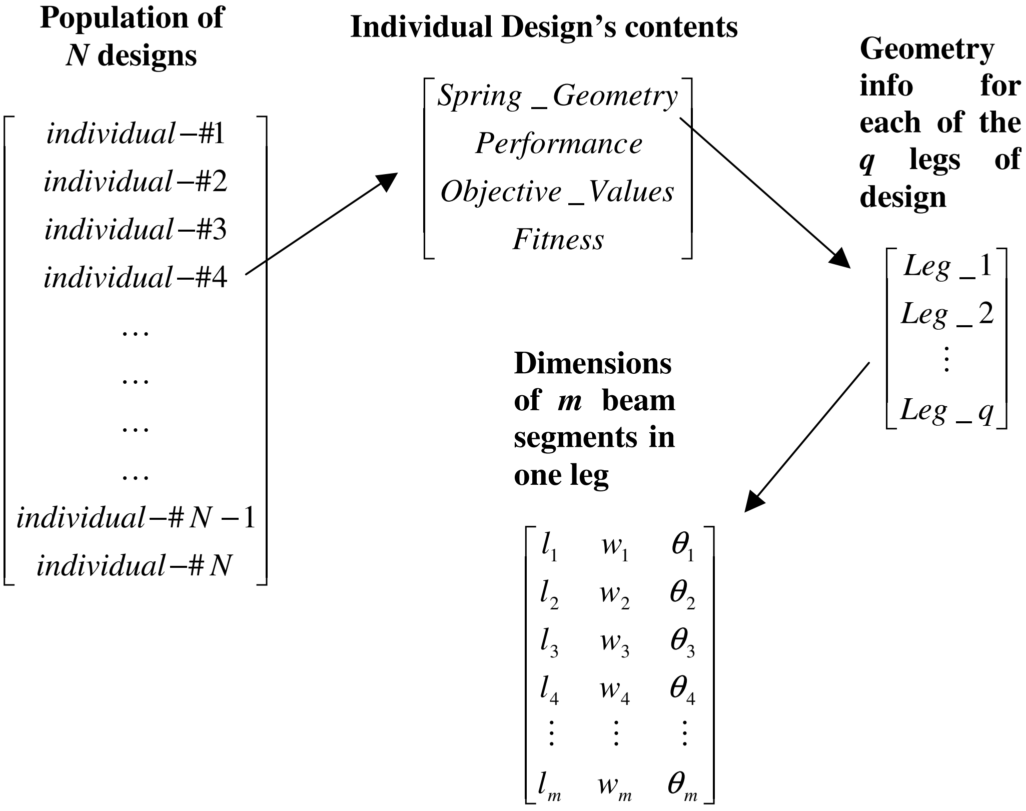


Figure 5.3: Illustration of Population and Geometry Encoding Method for Multi-Leg Resonating Mass Example.

Special cases of this data structure were created for alternate constraint cases, such as the symmetric version of the resonating mass, which has one n x 3 matrix that describes all 4 legs. In the same way, this data structure is capable of describing designs with more than 4 legs or even have the number of legs to be a variable that can change between individuals in the same population.

5.2. Design Objectives

As the resonating mass is our most established evolutionary MEMS example, we have used several sets of objectives to synthesis designs. Zhou's original implementation synthesized for three performance goals, designed to match the performance of a resonating mass presented in [40] in resonant frequency, suspension stiffness in the lateral direction and the vertical (transverse) direction.

We transformed this into a four objective optimization problem by adding design area as an additional objective. This forth objective has a goal of zero area, which makes it an optimization problem rather than simply a constrained synthesis problem. Chapter 7 presents examples of designs generated for these objectives that were later fabricated and tested.

We have also implemented a two objective version, this formulation minimized area while aiming for a certain resonant frequency. In this case a stiffness constraint was added, to ensure that the lateral stiffness was greater than the vertical stiffness, so that the resonant mode that was being optimized was in the correct direction. This implementation was used to study the impact of constraints on synthesis, and is discussed more in depth in the next section.

5.3. Role of Constraints in MEMS Synthesis

Until now the resonator synthesis examples have consisted of a fixed center mass (either with or without electrostatic comb drives) connected to four legs, each made up of multiple beam segments (Figure 5.4). We have run our MOGA synthesis program for several sets of performance objectives, all calculated using the SUGAR simulation program. As we are designing resonators, the most significant performance objective for all structures is the in-plane resonant frequency, which is a function of the center mass and the leg stiffness (and to a lesser extent, the leg mass). Other performance objectives we have used for synthesis include the stiffness of the structure in the x or y-direction as well as the device area (defined as a bounding rectangle around the comb drives and resonator legs).

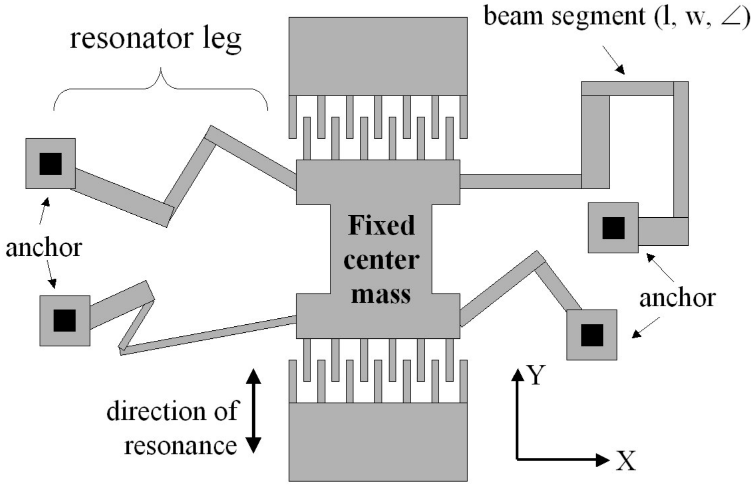


Figure 5.4: Schematic of Resonator Synthesis Example Problem. The geometry of the center mass is fixed, while the number of beam segments per leg and the size and angle of each segment is variable

It is a common misconception amongst non-MEMS designers that designs with Manhattan (or 90 degree) corners are better from a manufacturability point of view then irregular intersections, as can be the case in the macro world. For example, milling a piece of steel with non-orthogonal sides can be time consuming and expensive. It is much more expensive to create a corner at 91 degrees instead of 90. But in the case of MEMS, fabricated through a lithography process that is similar in many ways to developing a photograph, any intersection between straight-line segments is equally easy to fabricate.

The content of the photo has no impact on the difficulty or cost to develop it. Any structure that can be described by MEMS industry standard CAD formats such as GDS or CIF can be transferred into a lithography mask and fabricated. The only real limitation is on the resolution of the lithography, impacting the cost to generate the mask and minimum size of features that can be fabricated.

When faced with multiple complex, competing objectives, there is no reason to believe that the optimal configuration for a given resonator specification must have symmetric legs or Manhattan angles. In order to explore the role of symmetry further, we chose constraint cases with double axis/single axis/no symmetry and with/without Manhattan geometry and other constraint variations.

We conducted the study using a variant of Zhou's original resonator synthesis example as a case study. Zhou's implementation had the following geometric constraints: sizing limits (maximum and minimum length and width on the beam segments that make up each leg) and maximum and minimum angle limits for each element (each element must radiate away from the center mass). Table 5.1 for constraints on the parameter values . Simple beam-crossing detection constraints were also imposed to avoid synthesizing physically meaningless designs consisting of beams crossing other beams, etc [39].

For convex optimization problems, constraints only serve to limit the design space and, if applied artificially, could produce suboptimal designs. One might argue that this is also the case for non-convex spaces if the optimization algorithm has the ability to jump out of local minima and has no computational time limits. The reality with stochastic optimization techniques, such as simulated annealing or GAs, is that the time required to search the entire design space can be astronomical and careful restrictions of the space can lead to better results when there is a limitation on the number of iterations. Thus it is possible that a specific design problem, such as our resonator example, could benefit from the reduction of variables and the reduced design space formed when Manhattan angle constraints or symmetry is enforced. On the other hand, these restrictions could severely limit the exploration of more compact design concepts that could exploit a broader range of angles.

Table 5.1: Design Parameters/Constraints Used for MEMS Resonator Case Study

Parameter	**Value**
Center mass dimensions	100µm x 100µm plate w/ 4 30µm x 50µm beams attached. Plus mass of the comb drives (11 fingers, 50µm long by 4µm thick with 3µm gap, plus a 4µm thick spine)
Max number of beams per leg	7
Min number of beams per leg	1
Max beam length	100 µm
Min beam length	10 µm
Max beam width	10 µm
Min beam width	2 µm

Forcing leg symmetry has benefits as well as drawbacks. Devices with asymmetric legs may not travel in orthogonal directions when displaced making them incompatible with comb drives and causing potential short-circuits between the inter-digitated fingers. On the other hand, as with the angle constraint, it is possible that optimal configurations may be excluded by forcing symmetry. A symmetry requirement could limit concepts that try to take advantage of other asymmetric aspects of the problems, such as asymmetric X and Y suspension stiffness goals. A solution consisting of two short, stiff legs and two longer more flexible legs may have the goal resonant frequency or stiffness but still occupy a smaller area then an equivalent design with all four legs symmetric.

One obvious benefit of constraints is that they can limit the number of variables involved in the search space, leading to faster synthesis of good designs. For example, each leg of the meandering resonators consists of up to seven beam segments, each beam segment has three variables associated with it: length, width and angle. Forcing symmetry between the four legs reduces the optimization problem with up to 84 variables into, at most, a 21 variable problem (the number of beam segments per leg can also range from a maximum of seven to a minimum of one). Likewise enforcing Manhattan angles would change a third of these variables into four discrete values (0, 90, 180, 270 degrees) rather than an infinite space of continuous values. This could lead to faster convergence or a larger number of good configurations evolved by the n^{th} generation.

In the next section we present the resonator design case study, with five constraint settings and discuss the quality of the results returned by the MOGA.

5.3.1. MEMS Resonator Synthesis with and without Angle and Symmetry Constraints

For this case study we adapted our existing MOGA program to synthesize designs with two objectives: (1) minimize error from the target resonant frequency of 10,000 Hz and (2) minimize the device area (as defined by the bounding rectangle). The same beam length and width constraints were applied from Table 5.1. Furthermore we constrained the resonance to be in the Y-direction; this was done by highly penalizing any design that had a lower stiffness in the X-direction than the Y-direction. Resonance in the X-direction would be impossible to excite with the comb drives aligned to the Y-direction, as they can only perturb the center mass in the Y-direction.

Initially the MOGA was run for a population size of 400 for 50 generations. After 50 generations the designs within the Pareto set with a resonant frequency within 5% of the goal frequency were recorded. Each configuration was run 25 times, each with a randomly generated starting population.

Based on the performance results, we chose to perform a second set of MOGA runs, this time for a population of 400 over 500 generations (10x the original setting). This second set of MOGA results allowed us to observe the MOGA searching wider sections of the search space, which we expected would allow it to find a larger number of good solutions. But we were also cognizant when given this much time to search, the MOGA may find and exploit gaps in our geometry constraint functions to find optimum solutions that are not fabricatable. Each constraint configuration was run 10 times for each of the five cases, also with a randomly generated starting population.

We applied five constraint cases to the resonator synthesis problem:

Case 1. Asymmetric legs, unconstrained beam angle
(up to 84 continuous variables)
Case 2. Asymmetric legs, constrained beam angle
(up to 84 continuous variables)
Case 3. Y-Symmetric legs, unconstrained beam angle
(up to 42 continuous variables)
Case 4. Symmetric legs, unconstrained beam angle
(up to 21 continuous variables)
Case 5. Symmetric legs, Manhattan beam angle
(up to 14 continuous variables; up to 7 base-4 integers)

Other than the beam-crossing constraint, cases 1, 3 and 4 were performed with no angle constraints, meaning a beam segment could run in any direction. Case 3 is the same as Case 4, except only Y-symmetry is enforced, not X-symmetry. The idea with case 3 is to explore the biological analog in symmetry. Case 2 uses the same angle constraint used by Zhou [35]: beam segments can only radiate away or parallel to the center mass. Case 5 has symmetric legs and also enforces Manhattan angles on all beam segments: all beams are orthogonal to the axes. Figure 5.5 presents examples of each constraint type. These are actual designs output by the synthesis software for each constraint setting (although not optimal solutions.) Note that the design space decreases as we apply successively more restrictive constraints in moving from Case 1 to Case 5.

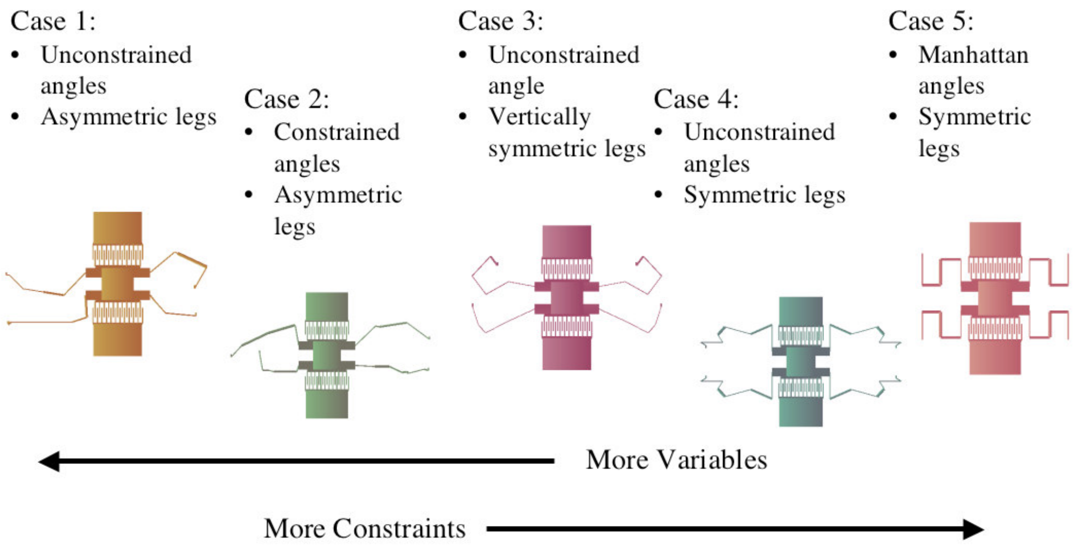


Figure 5.5: Example Designs For Each of The Five Test Cases.

Table 5.2: Best Resonator Performance Evolved Using Five Different Constraint Settings After 50 Generations. The objectives were reaching a resonant frequency of 10 kHz +/- 5% and minimizing device area. No designs were synthesized in case 1 with suitable resononant frequency performance.

Constraint Case:	Best Design within 5% of 10,000 Hz after 50 generations	
	Resonant Frequency (Hz)	**Area (m^2)**
1	*	*
2	10,327	2.105E-07
3	10,325	1.711E-07
4	10,463	1.55E-07
5	10,380	1.703E-07

Table 5.3: Best Resonator Performance Evolved Using Five Different Constraint Settings After 500 Generations. The objectives were reaching a resonant frequency of 10 kHz +/- 5% and minimizing device area.

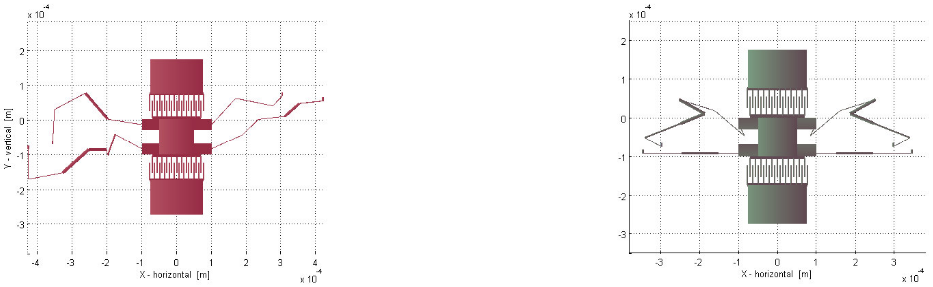
Constraint Case:	Best Design within 5% of 10,000 Hz after 500 generations	
	Resonant Frequency (Hz)	Area (m^2)
1	10,445	2.073E-07
2	10,469	1.698E-07
3	10,237	1.713E-07
4	9,987	1.485E-07
5	10,199	1.455E-07

The results of the 50-generation tests can be seen in Table 5.2 and Table 5.3. Case 1, with minimal constraints, was unable to produce any designs within the 5% frequency threshold range, but cases 2, 3, 4 and 5 were able to produce 22, 3, 69, and 38 threshold designs, respectively. Case 2, which only limits the beam angles to radiate away from the center mass, is able to produce designs with the desired resonant frequency, but with the worst minimum (Table 5.2) and average (Table 6.1) area of all of the cases. This case has more variables to optimize than all of the other cases that produced results, and the MOGA may not have been able to effectively search the design space in only 50 generations. It is also likely that the outward radiating angle constraint eliminates space efficient designs that have beams turning back towards the center mass.

An analysis of the threshold designs shows there is a significant difference between the cases with respect to minimizing device area. An analysis of variance (ANOVA) test of the designs supports the hypothesis that different constraint cases produce significantly different populations with respect to device area ($F = 34.6$, $p < 0.01$) [54]. But as we only looked at a specific range of frequencies (10kHz +/-5%), ANOVA does not show a significant difference in frequencies among the threshold designs (F=0.93, p=0.43), meaning all constraint cases were able to produce designs with similar resonant frequencies.

Case 4 performed slightly better than all of the other cases in minimizing device area, but the average over all of the threshold designs was higher than cases 2 and 3, suggesting that its slightly better performance here was “an outlier” to a pattern. Another interesting observation is that Case 4 – with full symmetry but no angle constraints – produced the largest number of threshold designs, approximately 80% more than case 5, over 3 times as many as case 2 and an order of magnitude more than case 3. Cases 1 and 2 had the largest number of degrees of freedom, yet produced inferior or unacceptable designs in 50 generations; the probability of geometrical violations (beams crossing each other, etc) is much higher in these cases, in effect causing a large portion of the population to be wasted on invalid or designs that are impossible to fabricate.

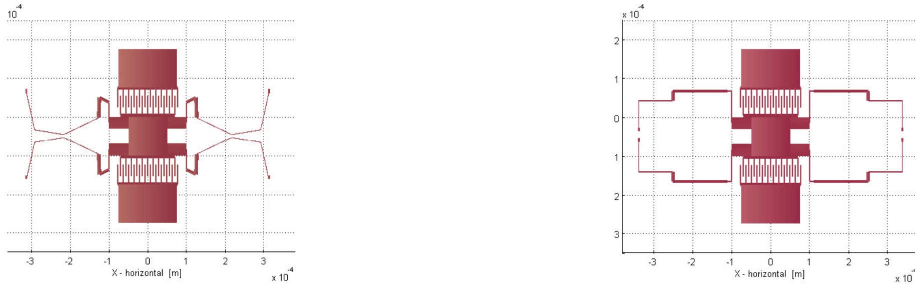
Case 2 **Case 3**



ω_r = 10,327 Hz;
Area = 2.105E-07 m^2

ω_r = 10,325 Hz;
Area = 1.711E-07 m^2

Case 4 **Case 5**



ω_r = 10,463 Hz;
Area = 1.55E-07 m^2

ω_r = 10,380 Hz;
Area = 1.703E-07 m^2

Figure 5.6: Smallest Threshold Design Synthesized in 50 Generations. Case 2 constraints (asymmetric beams, angular constraints away from center mass); case 3 constraint (vertical symmetry on legs); case 4 constraint (full symmetry on legs); and case 5 constraints (symmetric legs and Manhattan geometry).

The fact that cases 2 and 3 produced the least number of threshold designs indicates that the lack of full symmetry, or at least the larger design space no or limited symmetry allows, plays a major role in the MOGA's ability to optimize over the resulting design space within 50 generations. We note that case 3 – with only Y-symmetry – outperformed case 2 in minimizing area and distance from the target frequency. Figure 4 shows the best design from each case for their set of threshold designs. Note that while the best designs from cases 3, and 5 were nearly identical in the objective space, their configurations are not similar in the design space.

Table 5.4: Average Statistics for All Designs Generated by 50 Generation MOGA That Were Within 5% of 10,000 Hz (i.e., within 500 Hz of the goal) Over 25 Runs.

Case	Frequency difference from goal of designs within 5% of 10 kHz	Area of designs within 5% of 10 kHz	Number of designs in Pareto set returned by GA	Number of Pareto set designs within 5% of 10 kHz per run
1	*	*	23.6	0
2	265	3.2260E-07	31.3	0.88
3	190	1.956E-07	23.4	0.12
4	156	2.046E-07	19.6	2.76
5	133	1.975E-07	22.8	1.52

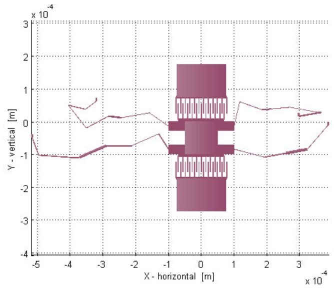
Table 5.5: Average Statistics for All Designs Generated by 500 Generation MOGA That Were Within 5% of 10,000 Hz (i.e., within 500 Hz of the goal) over 10 Runs.

Case	Frequency difference from goal of designs within 5% of 10 kHz	Area of designs within 5% of 10 kHz	Number of designs in Pareto set returned by GA	Number of Pareto set designs within 5% of 10 kHz per run
1	135	2.526E-07	29.6	1.00
2	155	2.217E-07	32.1	7.90
3	80	1.916E-07	26.3	2.70
4	25	1.814E-07	21.8	1.70
5	42	1.595E-07	27.6	3.00

Looking at the 500-generation tests in Tables 4.2b and 4.3b helps further clarify the effects of the constraints on performance. In general, we can see that equal or better performance was reached compared to the 50-generation tests (Figure 5.6). Furthermore, because of the increased number of generations, the Pareto sets returned by the MOGA are populated with more and better threshold designs. In case 1, by searching the design space further, the 500-generation MOGA was able to successfully generate better designs. However of the 10 runs performed, only three were able to generate successful valid designs within the specified frequency range. At least 250 generations were required to generate one design in this range. This illustrates that given enough computation our MOGA was successfully able to find acceptable solutions, albeit less optimum ones, even in this much larger, unconstrained design space.

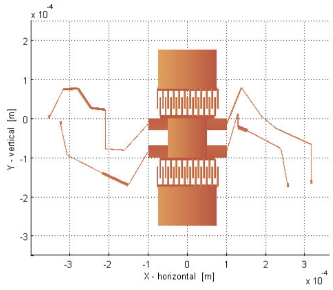
Case 2 for 500-generations returns interesting results, as mentioned previously, the outward radiating angle constraint limits the area minimization, but as the design space has been limited, the number of threshold designs generated is much greater than case 1. The number of threshold designs generated is also affected by our constraint implementation, which will be discussed further below. The case 3 results standout, as the best design evolved within 500 generations was slightly larger than the best design evolved in the 50-generation test. The limited sample size could explain this variation. But the number of successful designs generated per run and their average performance does increase as expected with more generations.

Case 1



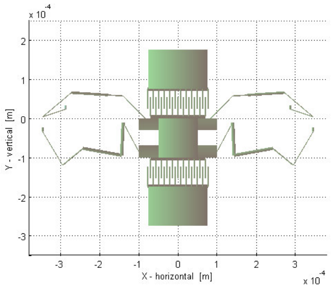
ω_r = 10445 Hz;
Area = 2.073E-07 m^2

Case 2



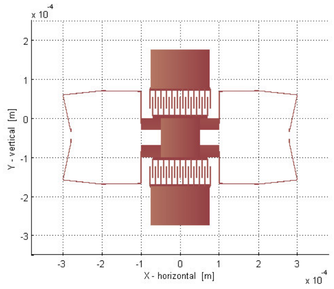
ω_r = 10469 Hz;
Area = 1.698E-07m^2

Case 3



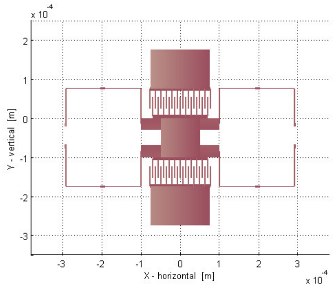
ω_r = 10237 Hz;
Area = 1.713E-07 m^2

Case 4



ω_r = 9987 Hz;
Area = 1.485E-07 m^2

Case 5



ω_r = 10199 Hz;
Area = 1.455E-07 m^2

Figure 5.7: 500 generation Cases: Smallest threshold design synthesized by case 1 constraints (asymmetric beams, no angular constraints); case 2 constraints (asymmetric beams, angular constraints away from center mass); case 3 constraint (vertical symmetry on legs); case 4 constraint (full symmetry on legs); and case 5 constraints (symmetric legs and Manhattan geometry).

Cases 4 and 5 for 500 generations produced designs with smaller areas and better average performance than the 50-generation case, as one would expect. However an interesting feature to observe is the much lower number of threshold designs generated for these cases compared to case 2 (Table 5.5), which is different than the 50-generation case (Table 5.4). This is caused by the large number of the designs generated by the MOGA that performing quite well in the objective space, but were non-fabricatable. In most cases this was due to the MOGA generating two types of flawed designs – designs that contained parallel overlap between legs and designs where the legs wrapped around the comb drives through the top or bottom bonding pads.

Both of these flaws demonstrate limitations of our node-based simulation environment. Two parallel or collinear beams will not violate our beam intersection constraint, which calculates the cross product of their centerlines to detect beam crossing. Even if the parallel centerlines are non-collinear, the beams may overlap, as they have finite width. Furthermore as SUGAR defines an anchor (used in this case as a bonding pad for our comb drives) as a single node, our beam crossing detection implementation has no ability to detect beams crossing the bonding pads. An example of a threshold design generated using constraint case 4 is shown in figure 6. In the case of the 50-generation MOGA tests, these two flaws rarely occurred and made up only a small number of the Pareto set designs, however given an order of magnitude more evolution, the 500-generation MOGA often discovered designs that took advantage of these two situations to produce designs that better satisfied the design constraints, filling up the Pareto frontier with designs that our beam overlap constraint function did not penalize.

Chapter 6: Advanced MEMS Synthesis Applications

To demonstrate the breadth of our MEMS synthesis approach, this chapter presents two examples of more advanced MEMS devices that can be synthesized using evolutionary methods, an ADXL type accelerometer and a vibratory rate gyro. Extensive study of the best-suited genetic operations and design constraints is not included, as the focus of this chapter is on the encoding and instantiation to apply evolutionary synthesis to new classes of design problems rather than the details of algorithm implementation. The configurations presented in this chapter are only example implementations of accelerometers or gyros, but by no means the only. The same implementations used for the types presented here can easily be adapted for other types of designs however.

6.1. ADXL-type Accelerometer

To demonstrate the applicability of evolutionary synthesis for MEMS applications we chose a well-known MEMS design application, the synthesis of an ADXL type accelerometer by Analog Devices. The ADXL-type accelerometers were one of the first commercial applications of MEMS and are used in applications such as automotive air bags, laptop drop sensors and other inertial measurement situations [55]. The numerous versions (there are currently 10 variations being produced by Analog Devices) vary in performance but share a similar configuration of the electromechanical portion of the design.

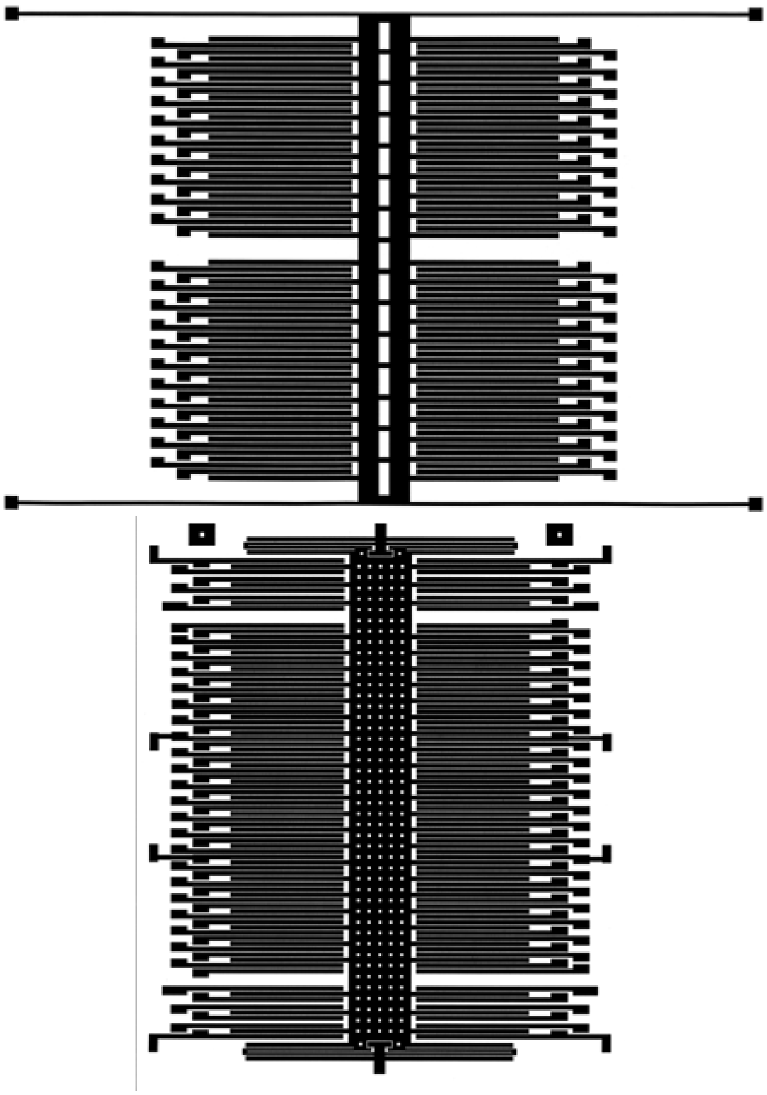


Figure 6.1: Analog Devices' Single Axis Accelerometers - top: ADXL 50, bottom: ADXL 150 [from 55].

As Analog Devices' layouts shown in Figure 6.1 demonstrate, designs are comprised of a suspended center mass, with sense fingers along both sides. Pairs of fixed fingers anchored to the substrate are interdigitated with the moving fingers on the center mass. The change in capacitance between the moveable finger and the stationary ones can be measured to calculate the acceleration-induced motion of the center mass.

6.1.1. Encoding

For this application we did two types of design encoding to describe the geometry of an ADXL-type accelerometer. The two methods share the same encoding for the center mass and sense fingers, and differ in the geometry of the suspension. One method incorporates the serpentine spring element (similar to the serpentine present in the ADXL design in Figure 6.1) and the 2nd uses regular beam segments connected in linear chains. An example of both types can be seen Figure 6.2. We also tried both versions with and without suspension symmetry to compare the output and demonstrate the flexibility of the encoding structure.

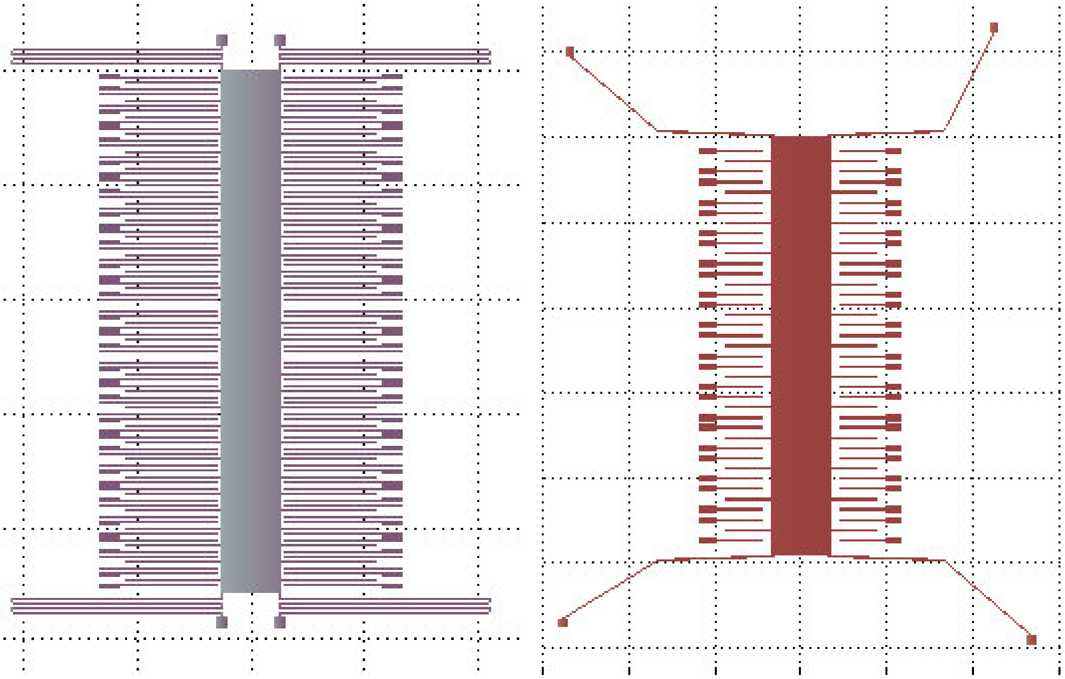


Figure 6.2: Example Instantiations Of Two Types Of ADXL-Type Accelerometers, With Serpentine Springs And Free-Form Multi-Segment Suspension Beams.

Each individual's geometry is divided into three geometry fields: *springs, spine* (center mass) and *sense fingers*. Table 6.1 describes the contents of each of these fields. The proper number of sense fingers is generated with the specified spacing along both sides of the spine when the individual design is converted to a SUGAR netlist.

Table 6.1: Encoding Structure for ADXL-Type Accelerometer Example. Two versions were created, one with regular meandering springs and one with free-form legs comprised of multiple beam segments.

Geometry field name	**Variables**	**Sample Min/Max Constraint**
Spine	Spine Length Width	50-500μm 20-100μm
Sense Fingers	Finger Length Finger Width Finger Gap	20-100μm 2-10μm 2-10μm
Springs: *Version 1: Serpentine springs*	Crenulation Length Crenulation Width Number of Crenulations	50-200μm 20-10μm 1-4
Springs: *Version 2: Multi-beam segment legs:* Four *n* x 3 matrices. Each matrix row represents a beam segment.	Number of beam segments per leg: *n* $\begin{bmatrix} l_1 & w_1 & \Delta\theta_1 \\ l_2 & w_2 & \Delta\theta_2 \\ l_3 & w_3 & \Delta\theta_3 \\ l_4 & w_4 & \Delta\theta_4 \\ \vdots & \vdots & \vdots \\ l_m & w_m & \Delta\theta_m \end{bmatrix}$	1-4 length: 10-150μm width: 2-10μm Δ angle: -90º, +90º

In the second method, the approach to encoding the geometry of the springs is similar to that of the resonating mass. A key difference is that in this case the angle representation is a delta angle, rather than an absolute global orientation. This difference is significant; it allows more control over the types of joint intersection evolved. The importance of this is discussed in more detail in section 7.4.

6.1.2. Design Objectives

Two problem formulations were tested for this application, the first was a three objective synthesis problem where there was a goal value set for resonant frequency, a goal value for transverse sensitivity (defined as the ratio of $K_{lateral} / K_{vertical}$). The third objective is to minimize area (defined as an orthonormal bounding box around the entire structure). In this case the sense fingers and center mass spine sizes were fixed.

The second synthesis problem was a four objective problem, three objectives were the same and the fourth objective added was to maximize the capacitive sensitivity. As mentioned in section 2.1, the relationship between finger geometry and sensitivity is monotonic, and we can easily predict in advance what geometry will maximize this particular objective. But in this case we are interested in the balance between maximizing capacitive sensitivity and the other three objectives (particularly design area, which is most directly affected by the size of the sense fingers), looking to the GA to synthesis interesting combinations of all four objectives. The user may choose a configuration with less than maximum sensitivity if it performs particularly well in a different competing objective. Sample output from these runs will be presented in the following section.

6.1.3. Sample Output

Figure 6.3 through Figure 6.6 demonstrate example output from several versions of the accelerometer synthesis GA. Figure 6.3 represents the output and performance values for the 3 objective case with asymmetric legs. In this case none of the objectives were a function of the capacitive sensitivity, so the sense finger sizing is arbitrary (except for its impact of their mass on the resonant frequency). Figure 6.4 and Figure 6.5 represent the output from the four objective implementation with serpentine spring legs. Both the asymmetric and symmetric cases are represented, respectively. Finally Figure 6.6 represents an output from the four objective problem with asymmetric mutli-segment legs - clearly this particular design represents a configuration where high capacitive sensitivity is chosen over minimized area.

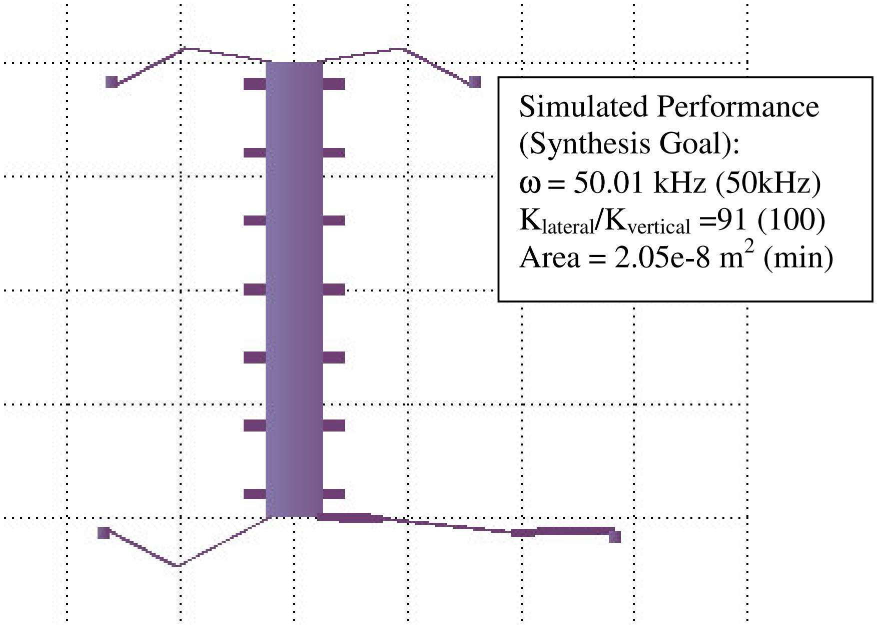
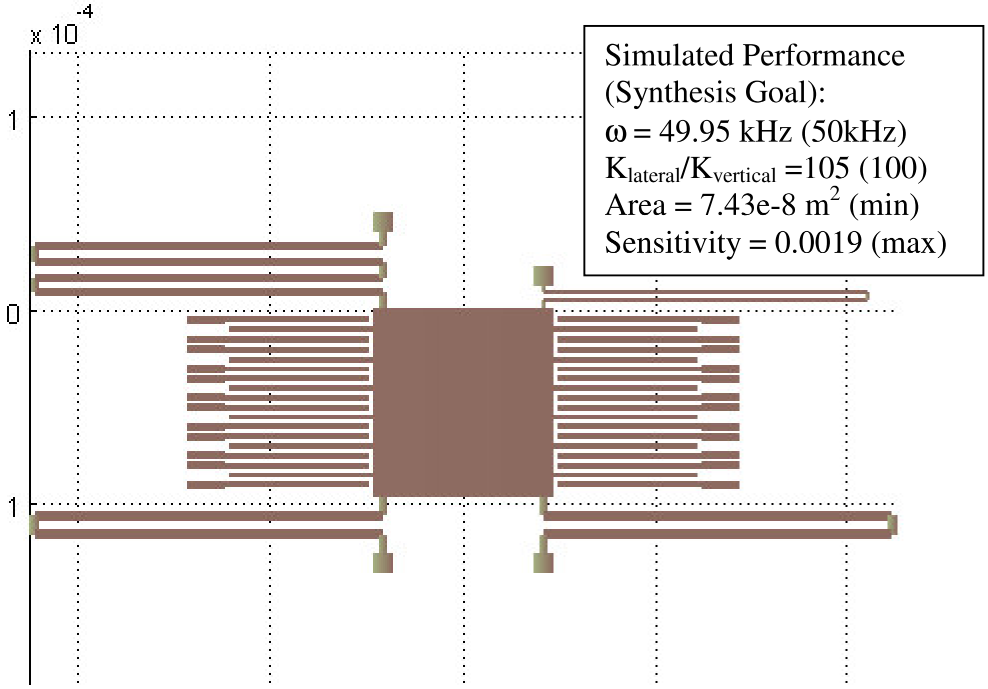
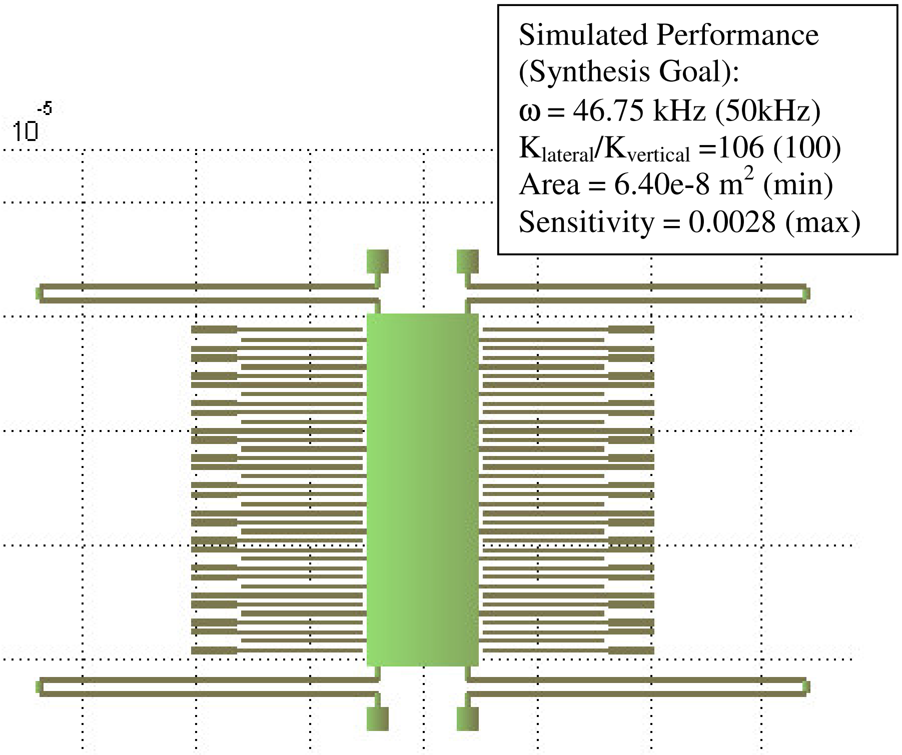

Simulated Performance
(Synthesis Goal):
ω = 50.01 kHz (50kHz)
$K_{lateral}/K_{vertical}$ =91 (100)
Area = 2.05e-8 m^2 (min)

Figure 6.3: 3-Objective Accelerometer Synthesis Output, with Symmetric Suspension Constraint Removed and up to 3 Free-Form Beams Per Leg Allowed. (Note: stationary sense fingers not shown).



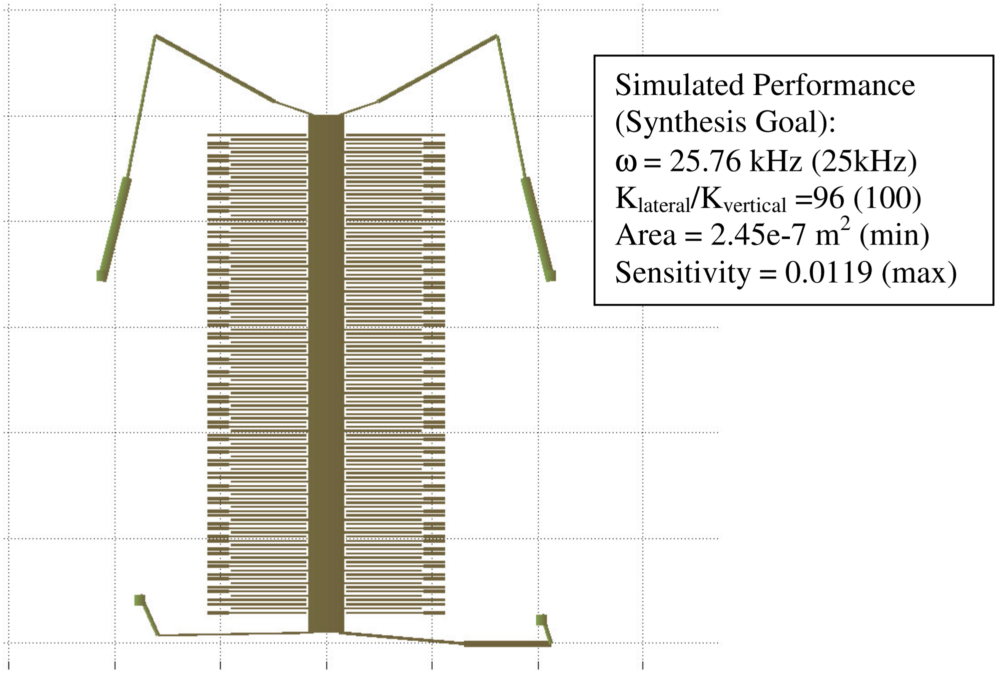
Simulated Performance
(Synthesis Goal):
ω = 49.95 kHz (50kHz)
$K_{lateral}/K_{vertical}$ =105 (100)
Area = 7.43e-8 m^2 (min)
Sensitivity = 0.0019 (max)

Figure 6.4: 4-Objective Serpentine Suspension Accelerometer Synthesis Output, with Symmetric Suspension Constraint Removed.



Simulated Performance
(Synthesis Goal):
$\omega = 46.75$ kHz (50kHz)
$K_{lateral}/K_{vertical} = 106$ (100)
Area = 6.40e-8 m^2 (min)
Sensitivity = 0.0028 (max)

Figure 6.5: 4-Objective Serpentine Suspension Accelerometer Synthesis Output, with Symmetric Suspension Constraint Activated.



Simulated Performance
(Synthesis Goal):
$\omega = 25.76$ kHz (25kHz)
$K_{lateral}/K_{vertical} = 96$ (100)
Area = 2.45e-7 m^2 (min)
Sensitivity = 0.0119 (max)

Figure 6.6: 4-Objective Accelerometer Synthesis Output, without Symmetric Suspension Constraints.

Note again that these examples do not represent a comprehensive search of the design space or best set of design objectives or constraints for a GA, but rather demonstrate the ability of GAs, and the flexibility of this data structure, to generate suitable output for a relevant set of objectives.

6.2. MEMS Vibratory Rate Gyroscope

Micromachined gyroscopes can be used to measure rotation rates for a variety of applications in navigation and guidance systems, automotive safety and consumer electronics. This section presents a MEMS gyroscope encoding scheme along with example output from an instantiation and naïve and Simulated Annealing optimization routines. A full GA implementation has not been performed, but this section demonstrates the applicability of the evolutionary approach to this type of MEMS application through the use of a simulated annealing synthesis.

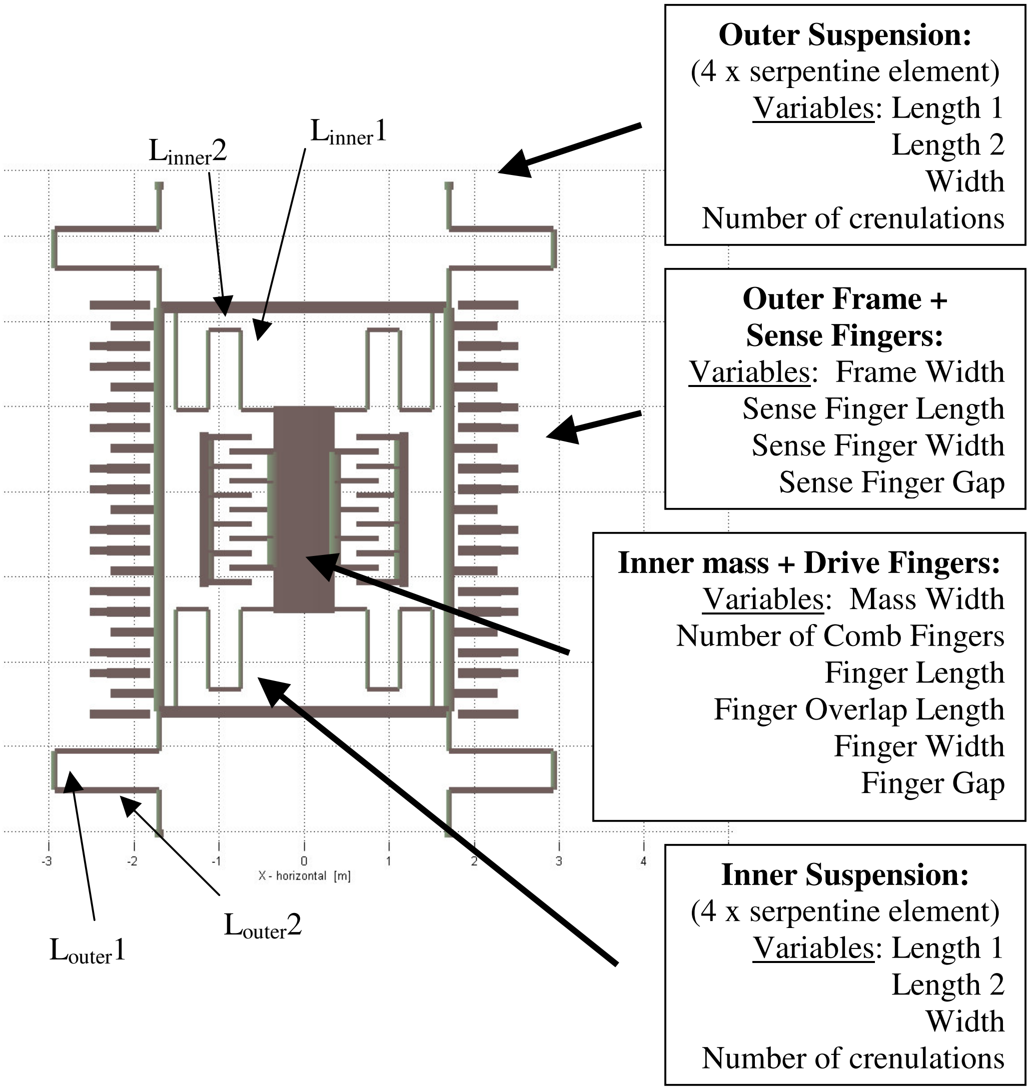
Vibratory rate gyroscopes (VRGs) use Coriolis acceleration to measure rotation rate. A mass is driven in vibration in the x-direction; rotation about the z-axis (out of plane) induces Coriolis acceleration in the y-direction, which is then capacitively sensed. Clark, Howe and Horowitz [56,57] presented a surface micromachined VRG that used resonance mode matching to maximize sensitivity. Acar and Shkel [58] have developed a VRG that decouples the vibrating masses into two independently oscillating proof masses. This version is designed not to be driven at resonance, and therefore is less sensitive to fabrication fluctuation.

We have chosen a VRG design similar to both Clark's and Acar's for this application example; the encoding scheme presented here could easily be adjusted to match either version directly if desired. A center mass is driven in the x-direction by two comb drives and connected to an outer frame by a set of four serpentine springs, the outer frame contains sense fingers which can measure the Coriolis induced displacement in the y-direction.

6.2.1. Encoding

We used a simple encoding scheme that uses parametric sizing within a fixed configuration (Figure 6.7). In this case the only variability in the number of discrete elements comes from varying the number of crenulations in a serpentine subnet and the number of drive and sense fingers, the remainder of the variables are continuous lengths and widths. Substituting the serpentine springs with crab-leg, double folded flexures or free form suspensions as implemented in the resonating mass example could be done easily with this flexible encoding structure.

The key design elements that are most important to the performance of this design are the relative spring constants in the x and y-directions for both the inner and outer suspensions (as Acar and Shkel refer to their design as a "Four degree-of-freedom Gyroscope"[58]) and the masses of the inner mass and outer frame. Figure 6.8 shows several more examples of instantiations using this data structure.



Outer Suspension:
(4 x serpentine element)
Variables: Length 1
Length 2
Width
Number of crenulations

Outer Frame + Sense Fingers:
Variables: Frame Width
Sense Finger Length
Sense Finger Width
Sense Finger Gap

Inner mass + Drive Fingers:
Variables: Mass Width
Number of Comb Fingers
Finger Length
Finger Overlap Length
Finger Width
Finger Gap

Inner Suspension:
(4 x serpentine element)
Variables: Length 1
Length 2
Width
Number of crenulations

Figure 6.7: Description of Vibratory Rate Gyro Encoding Structure – each design has 4 fields that describe the geometry of the device; each of these fields contains several parameters that can be altered by an optimization routine

Like the accelerometer presented in the previous section, the number of sense fingers is not an explicit variable, rather the number of sense fingers is dependent on the size of the outer frame, the sense finger width and gap. An alternative encoding scheme where the number of fingers is set and the length of the outer frame is adjusted accordingly could also have been implemented.

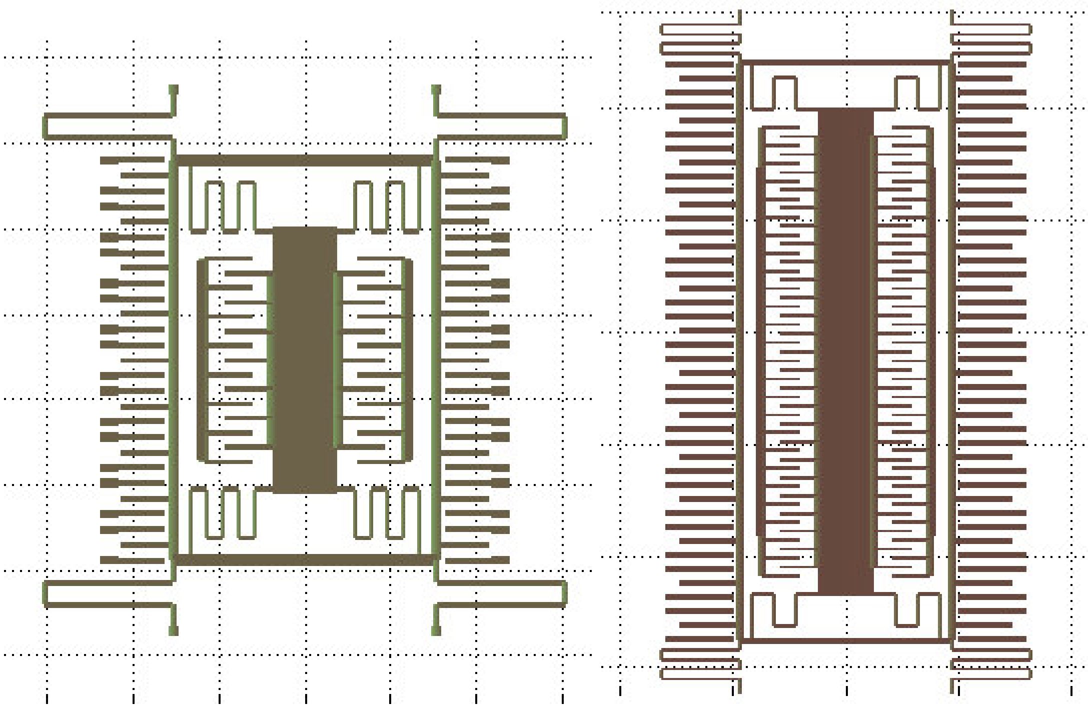


Figure 6.8: Example Instantiations Of Gyro Design Encoding.

6.2.2. Design Objectives

As we are dealing with a more complex device than presented in the previous sections, the number of objectives of interest to a designer has increased. As we are dealing with vibratory devices, resonant frequency is especially critical, it is important to ensure that the minimum resonant frequency is well above the bandwidth of extraneous accelerations the device might be subjected to, that the Coriolis accelerations are distinct from noise. Synthesizing for a desired resonant frequency with a GA or SA can be achieved in the same manner as shown in the ADXL and resonating mass example in this thesis.

Clark style VRGs are particularly interested in very close mode matching between resonance of the inner mass and the outer mass, providing maximum rotational rate sensitivity. Optimization of the mode matching can be easily achieved as a minimization of the difference between the two resonance frequencies. Although the previous synthesis examples focused on only the lowest resonant frequency, SUGAR has the ability to simulate the higher order modes as well. Using the mode shape information encapsulated in the eigenvector of the system, the direction of oscillation of a mode can be identified. The two frequencies corresponding to x-direction resonance and y-direction resonance can be extracted and the result supplied to the objective function.

Cenk style VRGs do not operate at resonance, but rather at a frequency between the inner and outer masses' resonant frequencies. In this case both resonant frequencies can be set as synthesis goals to ensure their proper placement. Other objectives of interest are similar to the previous examples in this chapter, i.e. capacitive sensitivity of the sense fingers, suspension stiffness, design area, etc.

It should be noted that if the number of crenulations of the serpentine spring is fixed, this problem could be solved analytically using gradient methods, but as we have shown in this section, there are a number of competing objectives which make it well suited for a multi-objective approach. Likewise, stochastic methods make alteration of the design configuration relatively quick and easy, not requiring the recalculation of the analytical equations to describe the new design.

6.2.3. Sample Output

While we have developed a data structure to describe this type of MEMS structure, we currently have not implemented a full genetic algorithm based synthesis for it. We have, however, written the instantiation routine to generate a random configuration as well as a mutation function to stochastically perturb a design. Using these functions, we first were able to create a naïve search implementation for a single resonant frequency goal. Figure 6.9 shows the results from a naïve search for a lowest resonant frequency of 50 kHz. Within 25 perturbations a solution within 300 Hz of the goal is found. Figure 6.10 is the SUGAR representation of this design.

This simple naïve search, purely focused on lowest resonant frequency is severely limited however, as can be seen by looking at the mode shape. Figure 6.11 shows that the lowest mode is in the sense direction, not the drive direction, as is desired, furthermore the lateral stiffness of the outer suspension relative to the inner suspension is too low this gyro application. Clearly synthesis with additional constraints and objectives is better suited to this application example.

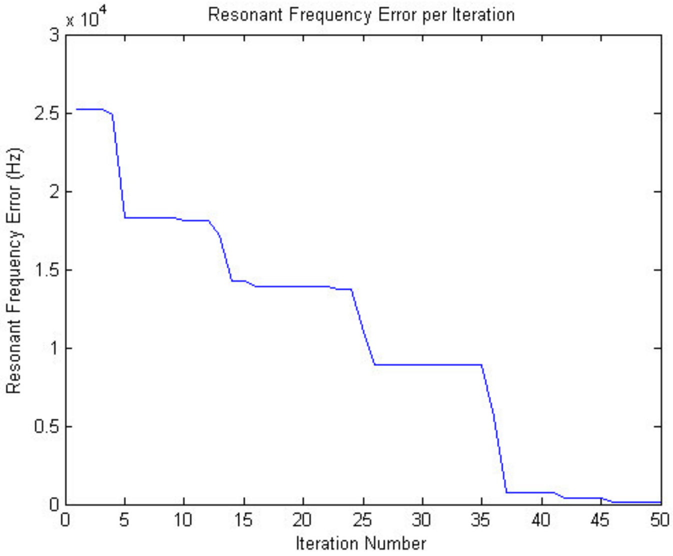


Figure 6.9: Resonant Frequency Error per iteration for simple naïve search synthesis of 50k resonant frequency gyro.

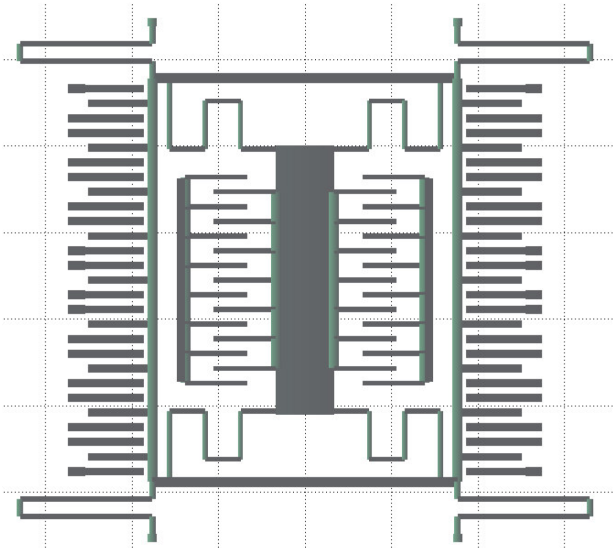


Figure 6.10: Layout of Synthesized Gyroscope with Lowest Resonant Frequency of 50kHz.

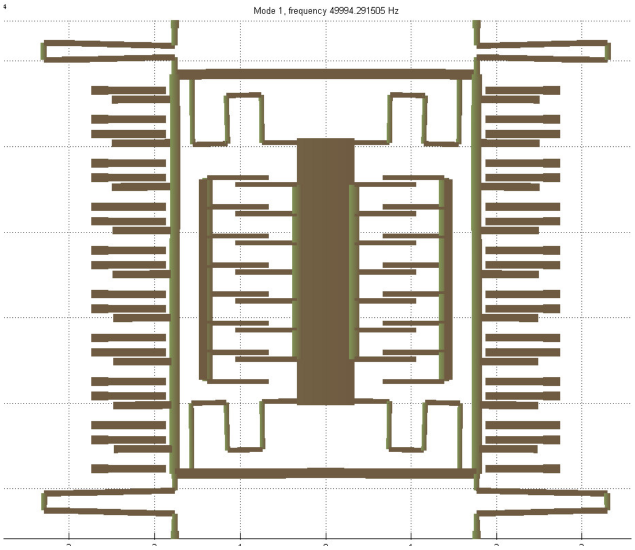


Figure 6.11: First Mode Shape for Design Presented in Figure 6.10. Oscillation is in the vertical direction.

To solve the issue of synthesizing the incorrect mode shape, we add a constraint to the system. When SUGAR is called to provide the resonant modes of the device, the eigenvectors provided with each mode are also inspected to decide if the mode shape associated with a resonant frequency is in the desired direction - in this case, the lateral direction for the lowest frequency and the vertical direction for the 2nd lowest frequency.

We implemented this constraint as a penalty function, designs in violation of the desired mode shapes were given extremely poor fitness values. In this case a naïve search was able to devices with oscillation at the lowest resonant frequency in the appropriate direction.

We applied a simulated annealing algorithm with a more realistic objective function to demonstrate the ability of this data structure to synthesize a meaningful design using stochastic optimization. We chose a Clark-style VRG design application, where we desire close matching between the lowest two resonant frequencies ($\omega_{r\text{-}lowest}$, $\omega_{r\text{-}2nd\_lowest}$) close to a frequency goal (ω_{goal}). The objective function becomes:

$$F(x) = \left|\omega_{r-lowest} - \omega_{goal}\right| + k\left|\omega_{r-lowest} - \omega_{r-2nd\_lowest}\right|$$

where k is a scaling factor, chosen to balance the importance of frequency difference to the matching frequency goal (the values of k =10 and 100 were used).

Figure 6.12 shows the objective function (dot-dashed line) along with the lowest and 2nd lowest resonant frequencies (solid and dotted lines respectively). We desire both the frequencies to be as close as possible and to approach our goal, in this case 10 kHz. In this case a k scaling factor of 100 is used. The final frequencies output after 150 perturbations was 8.38 kHz and 8.40 kHz. A lower k factor produces designs closer to 10 kHz but with a larger difference in frequencies. Figure 6.13 shows the mode shapes for the final design at 150 perturbations. Note that the absolute displacement in the mode shape plots is arbitrary (as it is taken from the eigenvector). Only the orientation and relative magnitude of the node displacement has relevance. The constraints have successfully forced the lowest resonant frequency to be in the lateral (drive) direction and the 2nd mode is the vertical (sense) direction.

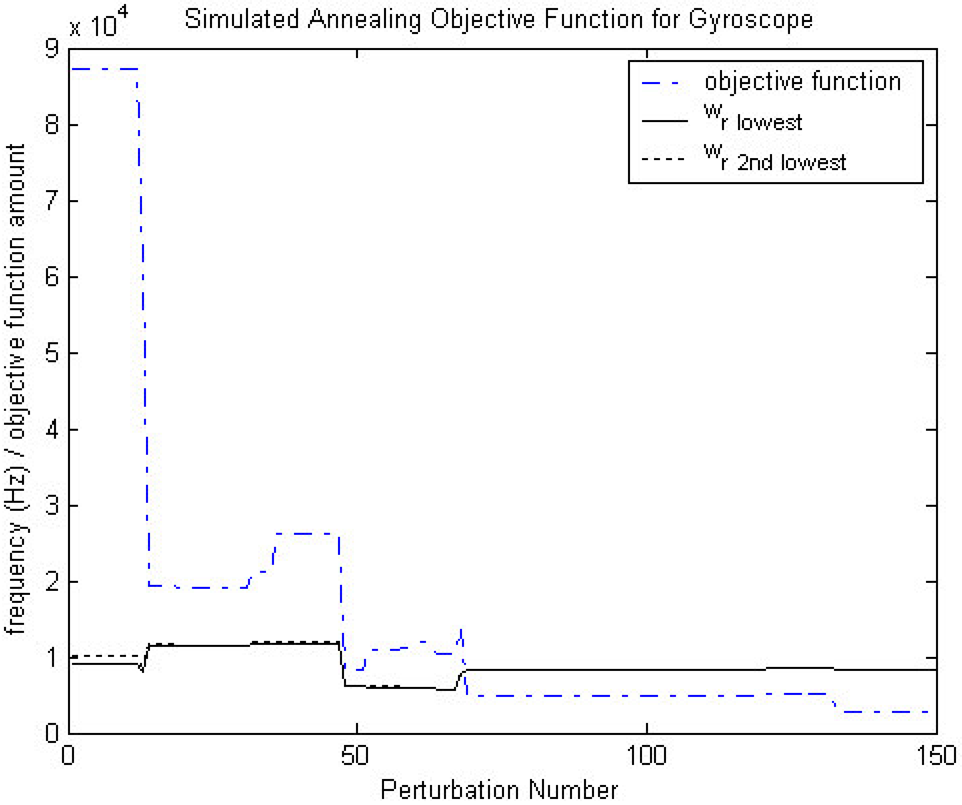


Figure 6.12: Simulated Annealing of Gyroscope. Objective function evaluation and design resonant frequencies per perturbation. Objective function is minimized, forcing the lowest resonant frequency and 2nd lowest resonant frequency close together and close to the goal of 10 kHz.

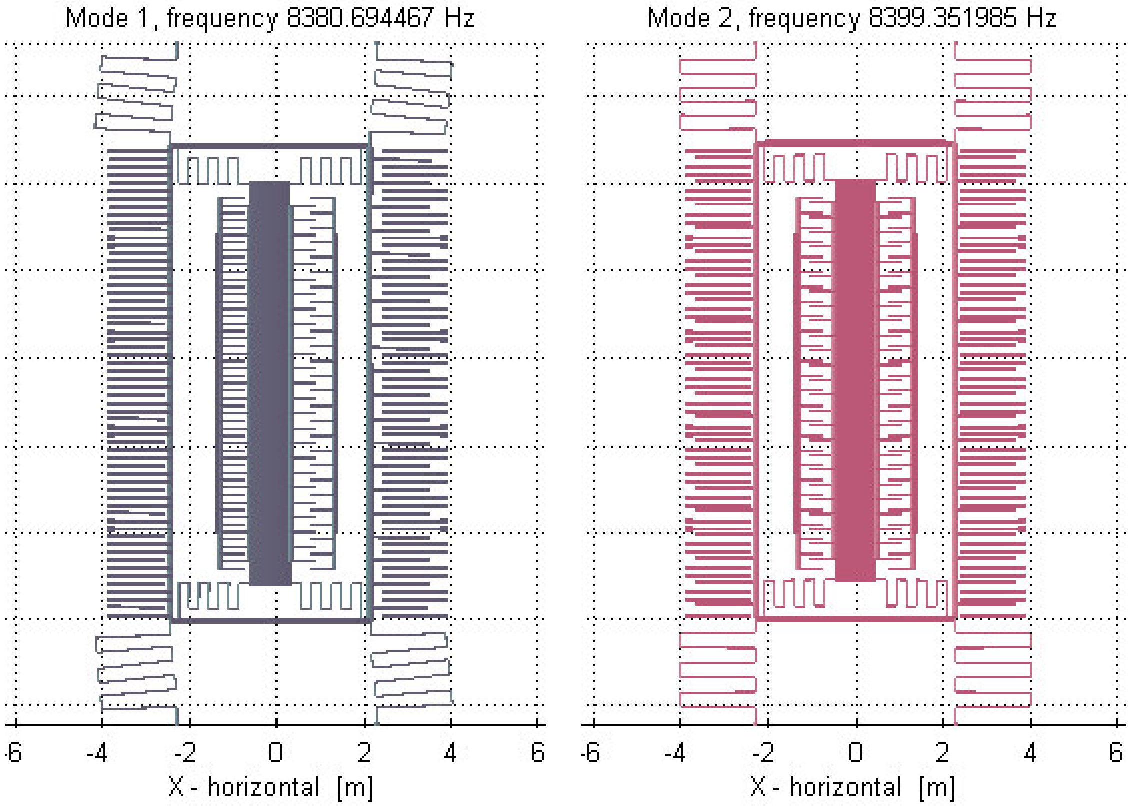

Mode 1, frequency 8380.694467 Hz
Mode 2, frequency 8399.351985 Hz
-6
-4
-2
0
2
4
6
X - horizontal [m]
X - horizontal [m]

Figure 6.13: Mode Shapes for Simulated Annealing Example. Lowest mode is forced to be in lateral direction and 2nd lowest is in vertical direction. Note that magnitude of displacement is arbitrary, only the orientation and relative magnitude of node movement has significance.

Chapter 7:
Fabrication and Characterization of Synthesis Output

7.1. Introduction

We fabricated a wide range of MOGA generated MEMS resonating masses to validate our MOGA based MEMS synthesis approach, to validate our use of SUGAR as an effective evaluation engine, and to better understand the constraints and objectives necessary for accurate synthesis. The designs were created with a variety of different GA settings and constraints. We compare the predicted results with measured performance from the fabricated structures; additionally the results are compared with finite element modeler’s simulation results to help better understand what factors contribute to discrepancies in certain types of geometries.

By tackling a wide range of devices generated by different GA settings, including inward turning suspensions, sharp joint angles and asymmetry, rather than limiting ourselves to typical symmetric, Manhattan suspensions, we are attempting to identify whether these constraints are truly necessary for MEMS structures, or are simply an artifact of engineers limiting themselves to ‘traditional’ methods of macro machine design.

In the macro-world, the development of sophisticated Computer Numerical Controlled (CNC) milling machines has allowed for macro designers to break free of the cost and practicality limitations that previously forced them to adhere to 90-degree corners. In the same way, the use of sophisticated and efficient simulation tools has allowed designers to break free from simple geometries they are able to analyze by hand. Likewise in MEMS design, particularly using a genetic algorithm to evolve layouts, helps to also explore non-traditional portions of the design space.

Characterization of the fabrication process and material properties is a critical component of the MOGA approach. Advance knowledge of Young's modulus of materials once they are fabricated is critical for stiffness calculations. Material density is also required for mass calculations. Both of these properties impact the resonant frequency of the structures we are generating. Additionally fabrication variations, such as line overetch, and non-vertical sidewalls have an impact on the performance of a fabricated device and therefore must be characterized.

The information learned in this study can be used to improve our MEMS synthesis software. By studying the types of features that occur in MOGA generated designs whose measured performance differs from their FEA or SUGAR predicted performance, we can identify fabrication or simulation issues that need to be addressed in our MOGA formulation to ensure better accuracy in future MOGA runs. This can take the form of added design constraints, or a change in the design objectives of the genetic algorithm.

7.2. Method

7.2.1. Resonator Test Case:

For our fabrication and characterization example, we chose the MEMS resonator test case of the type presented in chapter 5. We synthesized devices for four objectives:

1. Resonant Frequency (ω_r)
2. $K_{lateral}$
3. $K_{vertical}$
4. Area

Note: $K_{vertical}$ is the y-direction when looking at the layout, not to be confused with the out of plane, or z-direction.

Our fabrication test devices were designed with two sets of goals:

Set 1:

1. Resonant Frequency (ω_r) = 10,000 Hz
2. $K_{lateral}$ = 100 N/m
3. $K_{vertical}$ = 0.5 N/m
4. Area = Minimize (goal = 0 Area)

Set 2:

1. Resonant Frequency (ω_r) = 15,000 Hz
2. $K_{lateral}$ = 100 N/m
3. $K_{vertical}$ = 1.0 N/m
4. Area = Minimize (goal = 0 Area)

As our fourth objective has a goal of zero area, which is impossible to physically achieve, it is impossible to synthesize a design that meets all four goals. Our primary interest was objective 1, the resonant frequency goal, with a minimal area (objective 4). $K_{lateral}$ and $K_{vertical}$ were specified less as specific goals, but more to ensure that the suspension evolved would be less stiff in the vertical direction so that resonance in the lowest natural frequency was also in that direction.

Note that depending on the film thickness, the lowest resonant frequency may be out of the plane (if $K_{out\text{-}of\text{-}plane} < K_{lateral}, K_{vertical}$). In this research we limited ourselves (in both simulation and characterization) to only in-plane resonance modes. We constrained our simulation to only look at in-plane resonance and as we ran our characterizations in the presence of air (where squeeze-film damping will damp out of plane motion), and our actuators only excited motion in the plane – we were only measuring in-plane resonance.

7.2.2. Fabrication of Synthesis Output

Each MOGA synthesis run outputs a set of non-dominated Pareto solutions. As mentioned above, no designs were able to meet all design objective goals, therefore designs that met our goal values in resonant frequency within approximately 10-20%, that also had higher lateral stiffness ($K_{lateral} >> K_{vertical}$) and a shape deemed 'of interest' (for either its unique shape or its predicted performance) were chosen for fabrication. One of our goals was to study the impact of different constraints and objectives of the MOGA on the viability of the output created, therefore we tried to select a diverse group of different designs, rather than just fabricating multiple versions of a few shapes.

SUGAR netlists were created for each of the solutions chosen for fabrication. A program was written to translate the netlists directly into parametrized cells (known as P-cells), allowing for layout in Cadence. The translation program also automatically included proper layout for bond pads attached to the four leg anchors and comb drives (Figure 7.1). Furthermore, the program automatically corrected joint gaps (Figure 7.2) at points where beam segments met. By connecting the vertices of connecting beams, we attempt to avoid potential abnormalities, such as stress concentration points that could cause the fabricated device to behave differently from the simulation software's prediction.

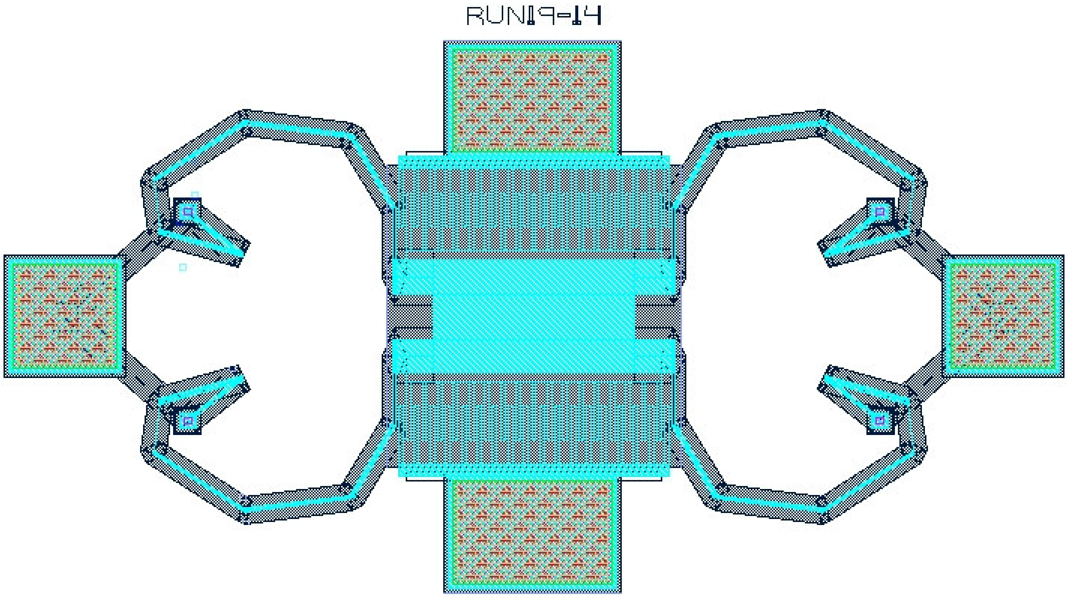


Figure 7.1: Example Of Synthesized Resonator Design Once Converted Into Cadence P-Cell. Bond pads and anchors are automatically generated.

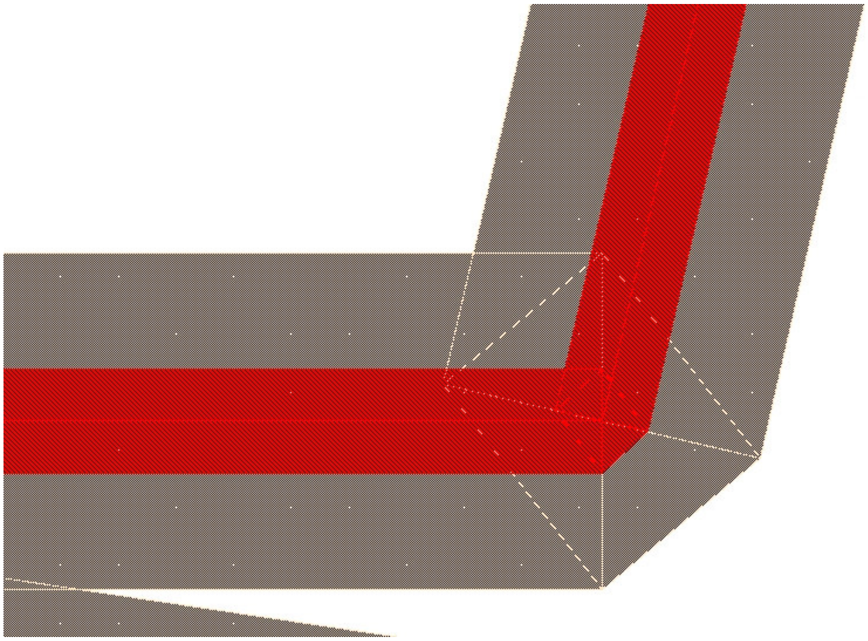


Figure 7.2: Gaps Between Two Rectangular Beam Segments Are Automatically Filled When Converted Into P-Cells.

The layout file generated contained two copies each of several test structures and 38 unique resonator designs generated by both MOGA and MOGA+IEC, for the constraint settings shown in Table 7.1.

Layout from Cadence was then submitted for fabrication to MEMSCAP's MUMPs fabrication house (formerly MCNC). We chose to evolve for a standardized commercially available fabrication process, the PolyMUMPs process [8]. Use of a standardized process is not only faster and more cost effective, it also reduces the likelihood of process variation or errors that would negatively impact our ability to produce consistent, characterizable devices.

Table 7.1: List Of Constraint Settings Used to Develop Designs Selected for Fabrication. Asymmetric, Symmetric and Symmetric with Manhattan Angles only were chosen for the two sets of objectives:

Run name	Symmetric Legs?	Manhattan Angles Only?	Max Number of Beams per Leg	Angle Range	Max Beam Length
Objective Set 1: ω_r = 10kHz, K_{lat} = 100 N/m, K_{vert} = 0.5 N/m, Area = min.					
Run 1	Y	N	7	$\pm\pi$	100μm
Run 4 / IECrun 4	Y	N	14	$\pm\pi$	100μm
Run 5 / IECrun 5	Y	Y	7	$\pm\pi$	100μm
Run 6	Y	Y	4	$\pm\pi$	150μm
Run 16	N	N	5	$\pm\pi / 3$	150μm
Objective Set 2: ω_r = 15kHz, K_{lat} = 100 N/m, K_{vert} = 1.0 N/m, $Area$ = min.					
Run 17	Y	N	9	$\pm\pi / 2$	150μm
Run 19 / IECrun 19	Y	N	7	$\pm\pi$	100μm
Run 20	Y	Y	5	$\pm\pi$	100μm
Run 21	N	N	7	$\pm\pi / 2$	100μm

Numerous works have been performed to characterize the material properties of the polysilicon MUMPs process since it was first introduced in 1992. The results of these characterizations are compiled in [59]. While there is variation between runs, the numbers for Young's Modulus, Poisson's ratio and material density are well known within our expected range of accuracy. We used 165 GPa, 0.3 and 2330 kg/m^3 respectively for the three values. Our suspensions were fabricated in the poly1 layer with a nominal film thickness of 2 um, with a poly0 backing below the structures.

The PolyMUMPs process uses a 0.25 um mask pixel size, meaning our feature resolution is 0.25um, with a minimum feature size of 2.0 um. We did not limit the beam width dimensions during simulation, so we expected a small amount of additional error between the fabricated (discrete beam widths) and simulated (continuous beam widths).

7.2.3. Characterization of Overetch

Upon receipt of the chips from the fabrication house, the structures were released and characterization began. The first step was to quantify the fabrication variation between the fabricated design geometry and the originally synthesized design geometry.

A critical issue with the fabrication of a CAD layout is overetch – the difference between the location of a line as laid out in CAD and the actual location on the fabricated device (Figure 7.3). In the case of beams in a MEMS suspension, a given beam's width will vary by twice the line overetch amount. Literature [60] cites a line width variation of 0.05-0.10 um, which would result in a beam overetch of 0.10-0.20um.

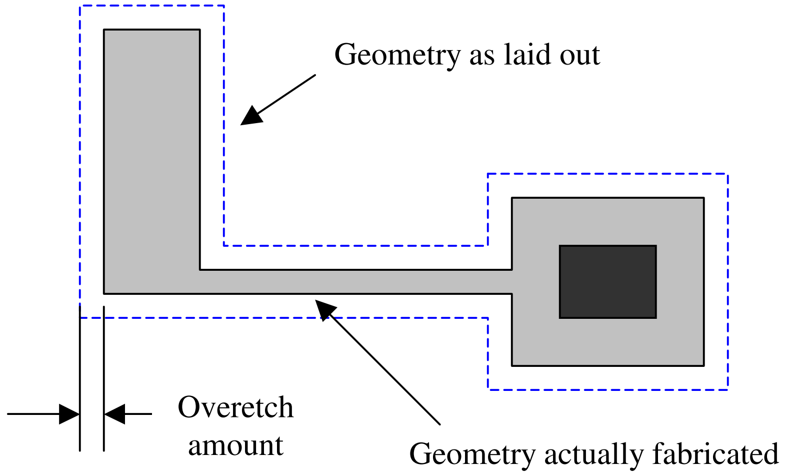


Figure 7.3: Illustration of Overetch in MEMS Layout

In anticipation of this overetch, we could have added an amount from 0.10-0.20um to all of our beam widths in order to account for the expected overetch. As this was our first attempt at characterization, we chose not to pre-alter our layouts for the expected overetch, rather we decided to submit our layout with the beam widths as evolved. To account for overetch more accurately, we chose to do measurements of beam width on the fabricated devices to compare against the layout widths to find the average beam overetch rate across all the structures.

7.2.4. Geometry Correction of Simulation Models

Once the average overetch is found, this amount can be subtracted from all the designs' simulation models and their performance re-simulated. This allows us to estimate the 'as-fabricated' performance of the structure. One of our primary foci for this research, is validating the accuracy of our simulator, but when overetch is accounted for (pre-fabrication or post-fabrication) is not of critical importance to this goal.

If the overetch rate is found to be consistent across fabrication runs, it can be added to beam widths in future layouts so that the 'as-fabricated' geometry (and performance) more closely matches the 'as synthesized' geometry (and performance). As this overetch causes a significant shift in the resonant frequency, with each unique design shifting a different amount, we framed our comparison between simulated and fabricated variation as a percent error, rather then an absolute difference in frequency, expressed in hertz.

7.2.5. Resonant Frequency Measurement

The fabricated designs were excited to resonance electrostatically via probe tips on a probe station. A DC bias voltage from a power supply was applied to one of the two leg bond pads and an AC voltage was applied to one of the comb drive pads. The other comb drive pad was grounded (see circuit diagram Figure 7.4). The AC signal was generated by an HP33120A function generator. The resonant frequency was detected manually by visual observation through the probe station's microscope CCD camera. Frequency was swept upwards until the amplitude of oscillations reached a peak. This was verified multiple times. We were able to achieve resolutions of approximately +/- 100 Hz in our resonant frequency measurements (based on multiple measurements of the same structures across different days).

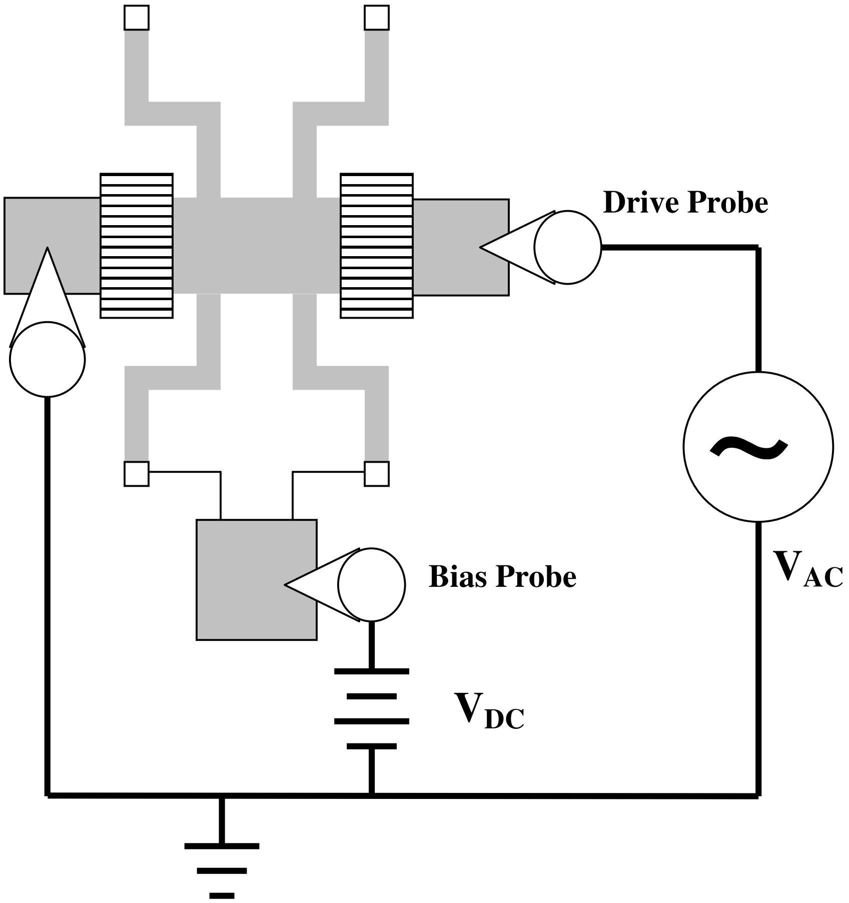


Figure 7.4: Electrical Schematic For MEMS Resonance Characterization. Probe tips are connected to both comb drive pads and bias probe is connected to leg anchors.

Multiple copies of the fabricated designs were measured across three of the MUMPS fabricated chips. For each device, the resonant frequencies measured were then averaged to give a mean performance for that design. This average number is used to compare against simulation.

7.2.6. Comparison with Simulation

The percentage difference between each device's average measured resonant frequency performance and its simulated performance is used as a benchmark for the accuracy of the simulator. In this case, we compared the measurements with both the predictions of SUGAR and the commercial FEA package ANSYS.

The first step for both simulations was to recreate a given design's SUGAR netlist with the average overetch subtracted from the beam width. Once this 'as fabricated' netlist was created, a SUGAR modal analysis was performed to find the 'as fabricated' resonant frequency (which has been shifted downwards from the original 'as synthesized' due to thinner, more compliant beams).

ANYSYS simulations were created from the same netlist used in SUGAR. Using the SUGAR function '*cho2cif*', the SUGAR layout is converted to the Caltech Interchange Format (CIF), which can then be brought into ANSYS using the built-in CIF importer. This imports the 2-D topology of the design into ANSYS; the layout was then merged into a single contiguous area through a Boolean union operation and meshed. It is important to note that the geometries in ANSYS did not include the joint gap filling that was included in the Cadence P-cell conversion. It was found that the minor geometric details in the joints between two beam segments did not have a significant impact on the ANSYS-simulation performance. As including these geometric details only varied the ANSYS predicted resonant frequency by a few 10's of Hertz (while our measurement resolution was in the 100's of Hertz), justifying the omission of these details.

It was found that 5um mesh spacing provided a sufficiently accurate lowest resonant frequency. Further refinement of the mesh size did not significantly alter the frequency. A 2D planar 8-node element with thickness was used for FEA, as more complex 3D element models were not necessary to find the in-plane resonant frequency.

The boundary conditions at the anchors were simulated by applying a zero displacement constraint to the mesh nodes at the center of the anchors – approximating the fabricated configuration where the inner 10 um square of 14 x 14 um anchor is affixed to the substrate. Due to the scale and stiffness of polysilicon, we found this approximation of the anchor sufficient for our level of accuracy.

A modal analysis was performed using the 'Subset' method (Block Lanczos also returns identical answers). A frequency sweep from 0 to 100 kHz was performed to identify up to the first five modes, although the measurements we were comparing to were only interested in the first (lowest) mode.

It is important to reiterate the significant difference between how a geometry is described and simulated in SUGAR and how a geometry is described and simulated in an FEA package like ANSYS. As mentioned previously, SUGAR uses nodal analysis, and looks at discrete elements joined at unique nodes. The ANSYS simulation, as we performed it, much more closely matches the real situation on a fabricated design, where two overlapping beam portions become a contiguous element. The impact of the difference between these two simulators is discussed in the next section.

7.3. Results

7.3.1. Overetch Measurements

After the chips were returned from MUMPS and released, we measured 55 features with help of a scanning electron microscope (SEM). The features measured were beam and comb drive finger widths at different places on the chip. The average amount of line overetch was 0.15 μm with a standard deviation of 0.06μm. We then modified the beam widths in our simulator models by two times this average value in order to compensate for the impact of overetch on design geometry (Table 7.2)

Table 7.2: Example from Synthesized Design "IECrun5-05", Comparing Original Beam Dimensions as Synthesized with SEM Measured Beam Widths and Beam Widths With Average Overetch Subtracted (Used in 'As Fabricated' Simulations). Difference in simulated performance for these three configurations is presented at the bottom.

Beam #	Original width as synthesized (μm)	Width of actual beam as measured in SEM (μm)	Width with average overetch subtracted (μm)
1	9.6	9.14	9.3
2	2.7	2.42	2.4
3	2.3	2.17	2.0
4	6.7	6.17	6.4
5	6.5	6.17	6.2
6	2.2	1.91	1.9
7	2.6	2.41	2.3
Simulated Performance (kHz)	10.9	9.00	9.10

Table 7.2 also indicates the impact of overetch on a sample design's performance. In the case of design "IECrun5-05" the resonant frequency of the design was shifted from 10.9 kHz to 9.10 kHz when all beam widths are reduced by 0.30 um, a shift of 17%

downwards. The table further shows that applying the average overetch (9.10 kHz) to all beams can provide a close approximation to the actual geometry (9.0 kHz), where the amount of overetch varies. Across all the devices fabricated, the average shift in resonant frequency after incorporating the overetch in the simulation model was approximately 17%.

7.3.2. Fabricated Device Resonant Frequency Measurements

180 separate measurements of resonant frequency were performed in all with 3-5 measurements for each design and the results averaged. For all devices the error between measured performance and SUGAR predicted performance ranged from -45 % to 87%, with an average of -8% and a standard deviation of 20%. In comparing ANSYS simulations with SUGAR for all devices simulated, the performance ranged from -5% to 88% with an average of 6% and a standard deviation of 16%. In comparing ANSYS simulations with measured resonant frequencies, the results ranged from –45% to 5%, with an average of -13% and a standard deviation of 10%.

As we fabricated a wide variety of unconventional designs (38 total), including designs that contain features that may not simulate properly in SUGAR or ANSYS or may not fabricate properly, we expected to find a wide variation in the performance, as represented by the large standard deviations. We need to segregate the different types of designs to better understand the factors causing this variation.

We found these three sets of results (SUGAR simulation, ANSYS simulation and measured devices) could generally be classified into three categories:

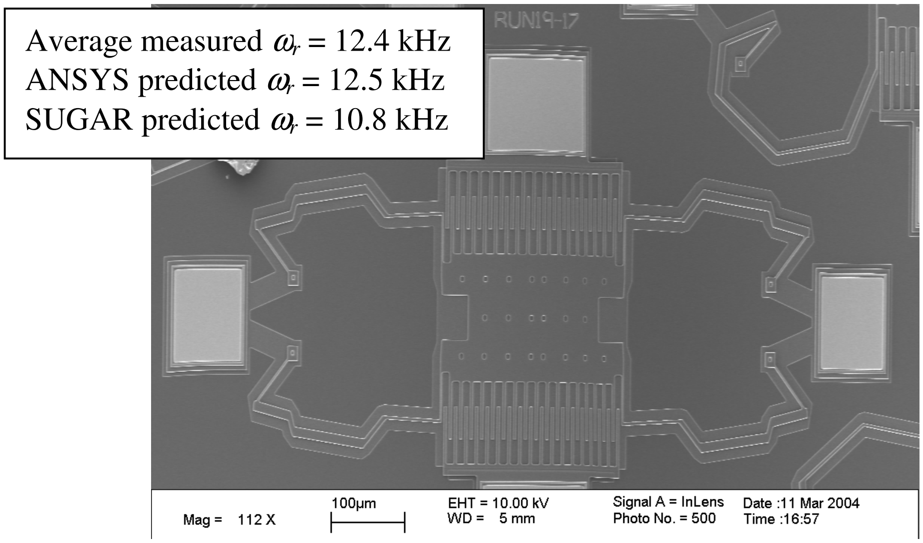
1) Designs where SUGAR and ANSYS and measured values generally agreed (13 designs).
2) Designs where ANSYS generally agreed with measured value, but SUGAR differed significantly (16 designs)
3) Designs where measured values different significantly from ANSYS and SUGAR predictions (9 designs).

By studying the differences between designs in the three groups, we are able to infer information about the impact that different design features have on simulated or fabricated performance. In the following section we will discuss the factors that caused the significant differences.

7.3.3. Description of Performance Groups

7.3.3.1. Group 2

Group 2 designs represent cases where SUGAR and ANSYS disagreed, and yet the ANSYS results are supported by the measured value, therefore the natural suspect is SUGAR's simulation method for causing the discrepancy. Inspection of designs that fell into this group show that error primarily arises due to the overlapping beam issue mentioned in the previous section: SUGAR models beams as discrete elements connected explicitly at a node point and does not account for beam segments overlapping. (Figure 7.5)


Average measured ω_r = 12.4 kHz
ANSYS predicted ω_r = 12.5 kHz
SUGAR predicted ω_r = 10.8 kHz

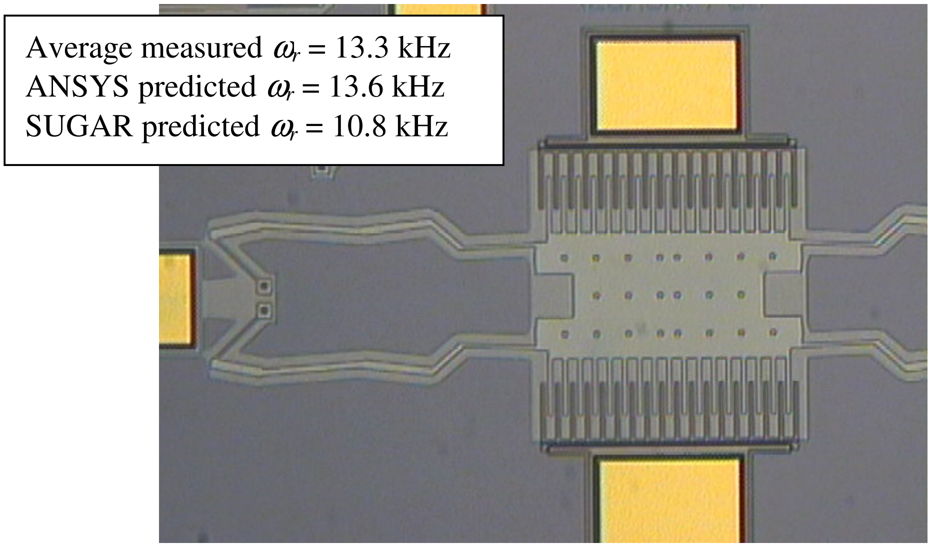

Average measured ω_r = 13.3 kHz
ANSYS predicted ω_r = 13.6 kHz
SUGAR predicted ω_r = 10.8 kHz

Figure 7.5: Examples Of Group 2 Designs. ANSYS generally agreed with measured values, but SUGAR predictions differed significantly.

In exchange for much faster simulation speed, SUGAR does not include the actual conditions at the end of the beam – it assumes a fixed connection exactly at the node point. In reality, when two beams connect at an angle, especially an acute angle, there is some overlap in their areas. Therefore the SUGAR simulation is based on the length of the beam as specified in the netlist, whereas ‘effective length’ of the compliant portion of the beam changes in both the ANSYS simulation and the fabricated design. ANSYS joins all the beam polygons into a single polygon through a union operation, overlapping areas are subsumed, as they are in Cadence layout that is used to create the fabricated design.

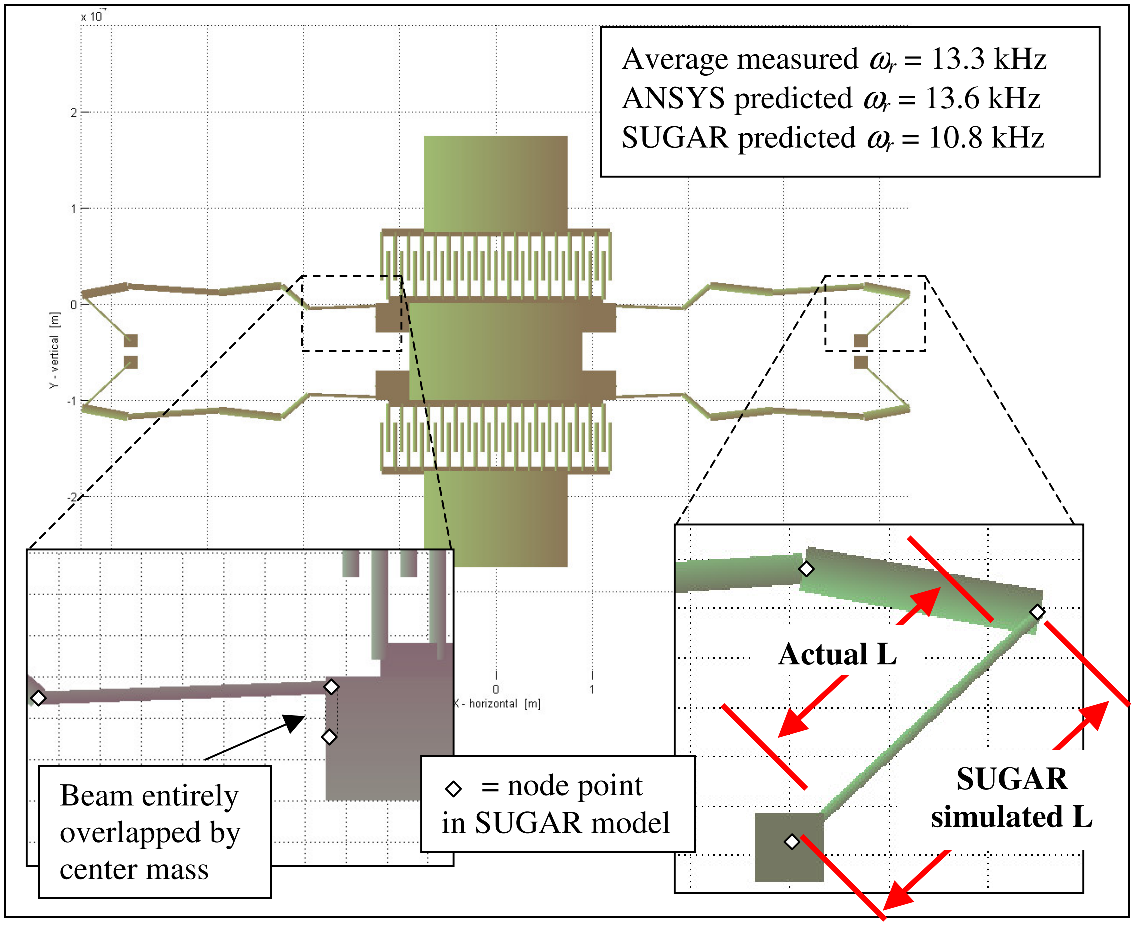

Average measured ω_r = 13.3 kHz
ANSYS predicted ω_r = 13.6 kHz
SUGAR predicted ω_r = 10.8 kHz
x 10
2
1
0
-1
-2
Y - vertical [m]
0
1
X - horizontal [m]
Actual L
SUGAR simulated L
Beam entirely overlapped by center mass
◇ = node point in SUGAR model

Figure 7.6: Example of Symmetric Design “IECrun19-03” (see Figure 7.5 - Bottom) Where Beam Overlap Caused Significant Disagreement Between ANSYS and SUGAR Simulated Performance. ANSYS and the measured value on the fabricated device did correspond. Significant overlap occurs at two points, where the legs connects to the center mass and the outer most beam segments (see insets).

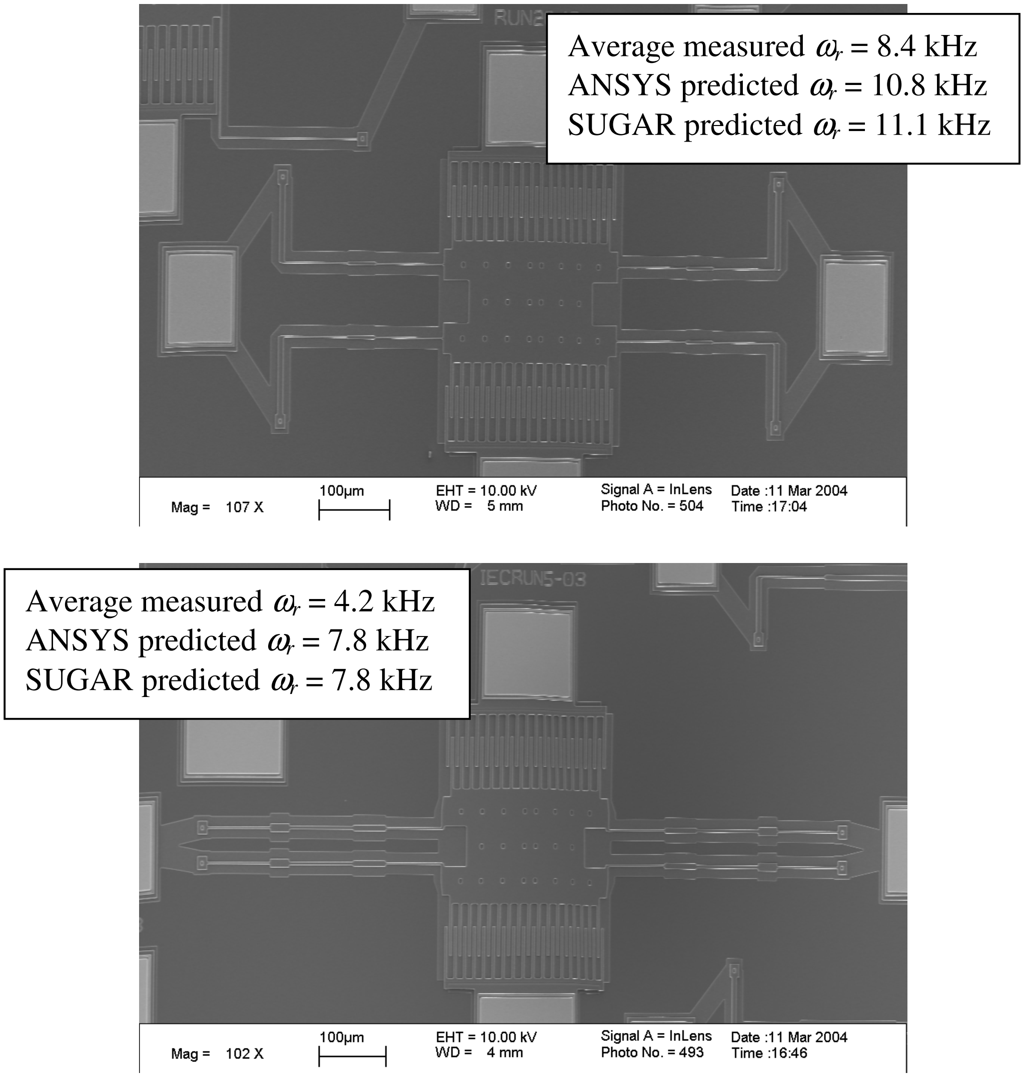
The SUGAR model for the lower design in Figure 7.5 is shown in Figure 7.6. This is a prime example of how overlap can cause a mismatch between ANSYS/fabricated and

SUGAR. In this symmetric design, significant overlap occurs at two regions, at the inner most beams that connect to the center mass and the outer most beams that connect to the anchors. SUGAR's simulation predicts a resonant frequency of 10.8 kHz, whereas ANSYS predicts 13.6 kHz, and the average measured value was 13.3 kHz.

The impact of these overlaps can clearly be seen by adjusting the SUGAR model to approximate the 'as fabricated' geometry. By removing the inner most beams (which are completely subsumed by the center mass in the fabricated device) from the netlist, the SUGAR predicted resonant frequency shifts upwards from 10.8 kHz to 13.1 kHz. Furthermore, removing 10 μm of length from the outer most beams to approximate the overlap of the anchor and the sharp joint angle makes the frequency 13.2 kHz.

7.3.3.2. Group 3

Group 3 designs represent cases where neither SUGAR nor ANSYS agreed with the measurements made from the fabricated designs. Due to the existence of groups 1 and 2, we believe that this group exists due not to a general error in our simulations or approach to accounting for overetch, but rather due to a discrepancy between the environment we are simulating and what exists in reality on the chip. It should be noted that all the designs that fall into group 3 are cases of negative error, in other words, the measured frequency is lower than the simulated frequency in all group 3 cases. (Figure 7.7)



Average measured ω_r = 8.4 kHz
ANSYS predicted ω_r = 10.8 kHz
SUGAR predicted ω_r = 11.1 kHz

Average measured ω_r = 4.2 kHz
ANSYS predicted ω_r = 7.8 kHz
SUGAR predicted ω_r = 7.8 kHz

Figure 7.7: Two Examples of Group 3 Designs. SUGAR and ANSYS simulations match but fabrication issues cause the measured value to shift downwards.

MEMSCAP includes limited characterization for each of their production runs, in the case of our chips, they list a compressive residual stress in the poly 1 layer (our designs' structural layer) of 5.0 MPa [61]. We believe this residual stress is responsible for the downward frequency shift.

Analytically, the relationship between residual stress and resonant frequency for a fixed-fixed beam can be expressed with the following equation:

$$\omega_{r'} = \omega_r \sqrt{1 - \frac{\sigma_{resid}}{\sigma_{crit}}} \qquad [62]$$

where σ_{resid} is the residual stress in the beam and σ_{crit} is the critical stress at which buckling occurs:

$$\sigma_{crit} = \frac{4\pi EI}{AL^2} \qquad [63]$$

where *A* is beam cross-sectional area, *I* is second moment of area, *E* is Young's Modulus for the beam and *L* is the length of the beam. As $\sigma_{resid}/\sigma_{crit}$ gets closer to unity, the design will undergo a larger shift in resonant frequency. In our case residual stress is nominally identical for all structures, but the critical buckling stress varies from design to design. From the above equations, its clear that designs with longer lengths and/or smaller cross sectional areas will have a lower critical stress and therefore be more susceptible to frequency shift.

We verified this phenomenon by analyzing a simple test structure ('test 1-1') comprised of 4 single fixed-fixed beam legs (Figure 7.8). This design, as fabricated, has 3.6 um x 2um x 520 um legs. Which makes gives a σ_{crit} of approximately 8 MPa. Using the MEMSCAP supplied σ_{resid} of 5 MPa, this gives a frequency shift of approximately 39% ($\omega_{r'} = 0.61\omega_r$).

In the case of this test structure, SUGAR and ANSYS predict a resonant frequency of 8.6 kHz and 8.4 kHz respectively, but the measured value was 6.3 kHz – approximately a 30% difference. The analytical equation does not include the true beam end conditions at the anchors or other extenuating factors (which would lead to a higher sigma critical, and therefore a smaller shift), so we take 30% to be close enough to the analytical prediction for this verification.

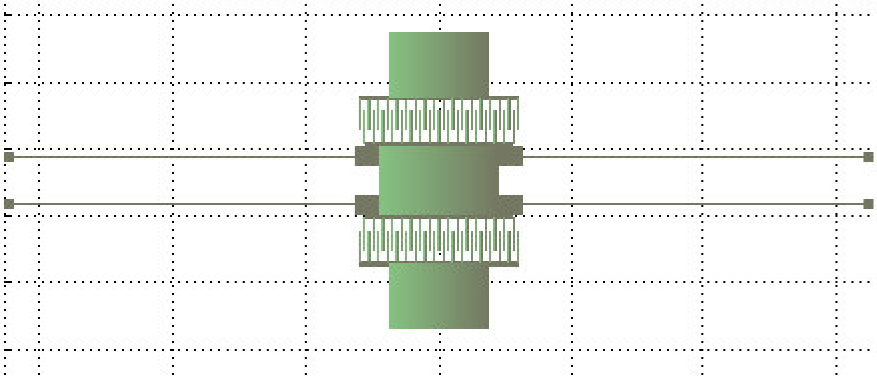
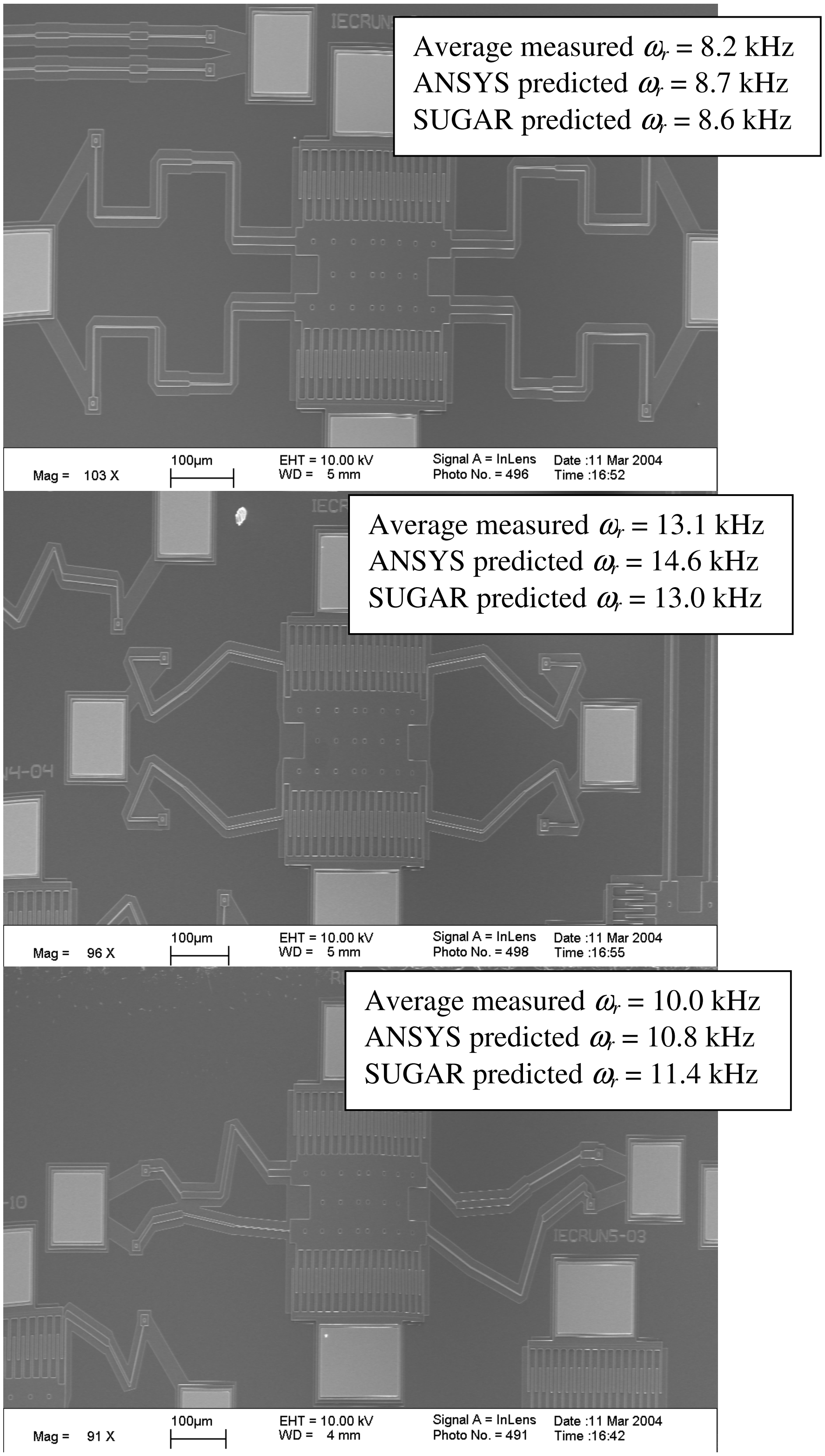


Figure 7.8: Simple Test Structure 'Test 1-1' Used to Verify Effect of Residual Stress on Resonant Frequency.

ANSYS can predict the effect of residual stress through a two-step simulation, which first performs a static analysis using a uniform pre-stress load and then performs modal analysis based on the pre-stressed mesh. Using this functionality, we repeat the simulation of the 'test 1-1' with residual stress included. For a residual stress of 5 MPa we find the predicted resonant frequency shift from 8.4 kHz to 7.7 kHz. A residual stress of 10 MPa results in a resonant frequency of 6.9 kHz. Based on this measurement we suspect the true residual stress to be higher than the 5 MPa reported by MEMSCAP. The ANSYS simulation does not include the 3D realities of the anchor geometry, etc; therefore we cannot make a definitive judgment about actual amount of residual stress.



Average measured ω_r = 8.2 kHz
ANSYS predicted ω_r = 8.7 kHz
SUGAR predicted ω_r = 8.6 kHz

Average measured ω_r = 13.1 kHz
ANSYS predicted ω_r = 14.6 kHz
SUGAR predicted ω_r = 13.0 kHz

Average measured ω_r = 10.0 kHz
ANSYS predicted ω_r = 10.8 kHz
SUGAR predicted ω_r = 11.4 kHz

Figure 7.9: Three Examples of Designs in Group 1: where SUGAR, ANSYS predictions of performance closely matched measured. These examples include a symmetric-Manhattan design, a symmetric unconstrained angle design and an asymmetric design.

In practical terms the residual stress issue means designs that have high axial stiffness and wide anchor spacing will not match simulations. In the section 7.4 we will discuss how this knowledge can be incorporated into the MOGA synthesis formulation to avoid designs with this fabrication issue.

7.3.3.3. Group 1

Group 1 is comprised of the remaining designs that did not suffer from simulation error (due to overlap) or fabrication error (due to the unsimulated residual stress) (Figure 7.9). In this case the simulation in SUGAR corresponded to the simulation in ANSYS and the measured value from the fabricated designs. For the 18 designs that fell into this group, error between ANSYS and SUGAR ranged from –5% to 6%, with a standard deviation of 3%, corresponding to a 95% confidence interval of ± 6% if we assume a normal distribution. This level of correspondence between the reduced order nodal simulation and the FEA simulation for complex shapes serves to validate SUGAR's choice as a viable simulation engine for our evolutionary MEMS design tool.

Comparing the Group 1 designs' SUGAR simulations with the measured values also serves to validate our approach to MEMS design synthesis using SUGAR. We found that the difference between SUGAR and measured values for this group ranged from –18% to 7%, with a standard deviation of 7% and a 95% confidence interval of ± 13%. These designs validate the approach that a SUGAR-based evolutionary synthesis approach can generate viable MEMS configurations when mated to the proper synthesis formulation.

As this was our first synthesis attempt using the PolyMUMPS process, we were uncertain about the material properties on the fabricated chip. This study also shows that the Young's Modulus of 165 GPa, Poisson's ratio of 0.3 and density of 2300 kg/m^3 is sufficiently close to the actual properties to predict performance. The outstanding issue remains beam overetch. Further study across more MUMPS fabrication runs should be performed to characterize the overetch problem, which will allow adding the expected overetch to design models in advance to resolve this issue.

7.4. Discussion

This section discusses the lessons learned from this characterization study and how it can be used to improve the Genetic Algorithm formulation to avoid designs with fabrication or simulation issues, as in the designs found in Groups 2 and 3, and ensure all designs stay within feasible regions of the design space.

A key limitation that shapes the recommendations made in this section is what is physically possible in the SUGAR simulation environment. For example, SUGAR's nodal analysis modeling does not incorporate the entire bulk solid, so detecting where polygons cross each other would require additional programming outside of the SUGAR environment.

Likewise SUGAR does not currently have the ability to simulate residual stress or stress gradients in a structure. Therefore these fabrication related issues cannot be directly included into a synthesis algorithm (neither as a constraint nor as an objective).

7.4.1. Proposed Design Constraints – Group 2

As mentioned in the previous section describing Group 2 designs, the major issue was 'beam overlap.' As shown in the example design in Figure 7.6, this type of issue created simulation mismatches in two distinctive ways.

First, where resonator legs connected to the center mass, if the innermost beam was at parallel to the edge and was short, in the fabricated design, this beam was either partially or entirely subsumed in the center mass and did not flex, whereas SUGAR did not model this properly. Therefore a constraint could be added to force the innermost beams to be normal to the center mass, or limit the minimum allowable length or width of these beams to avoid this beam from being a thin compliant beam with its effective length significantly changed through overlap with the center mass (Figure 7.10).

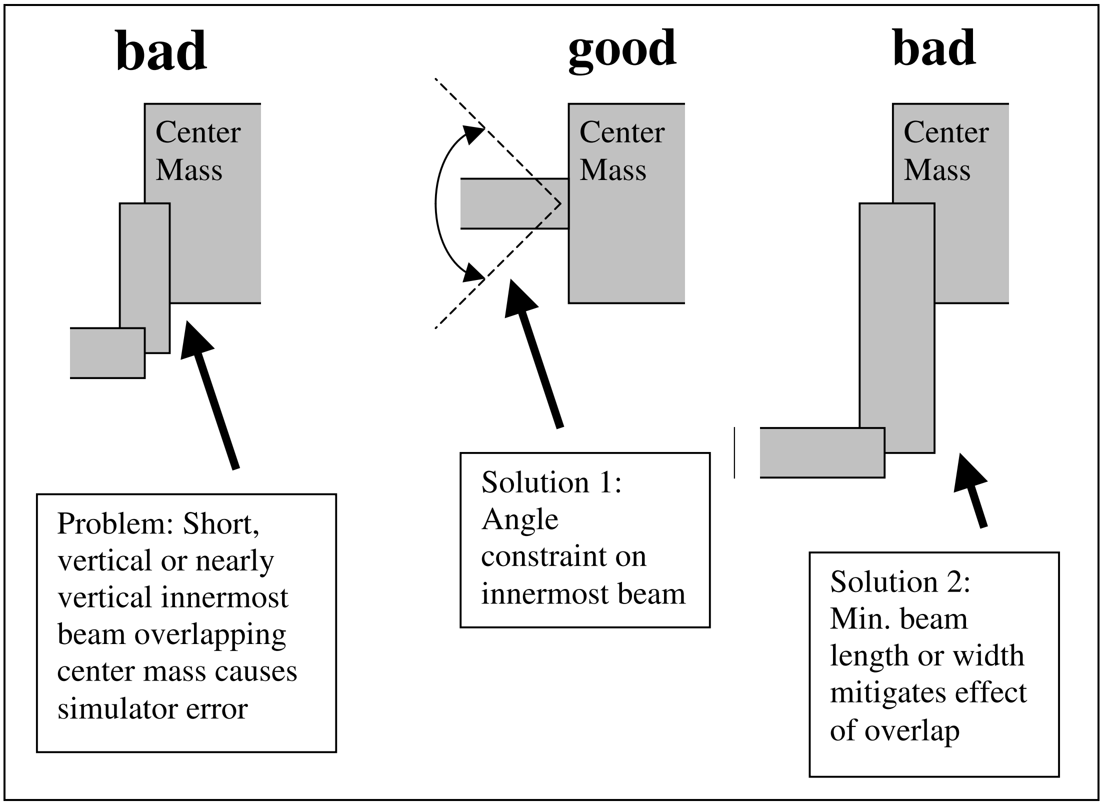
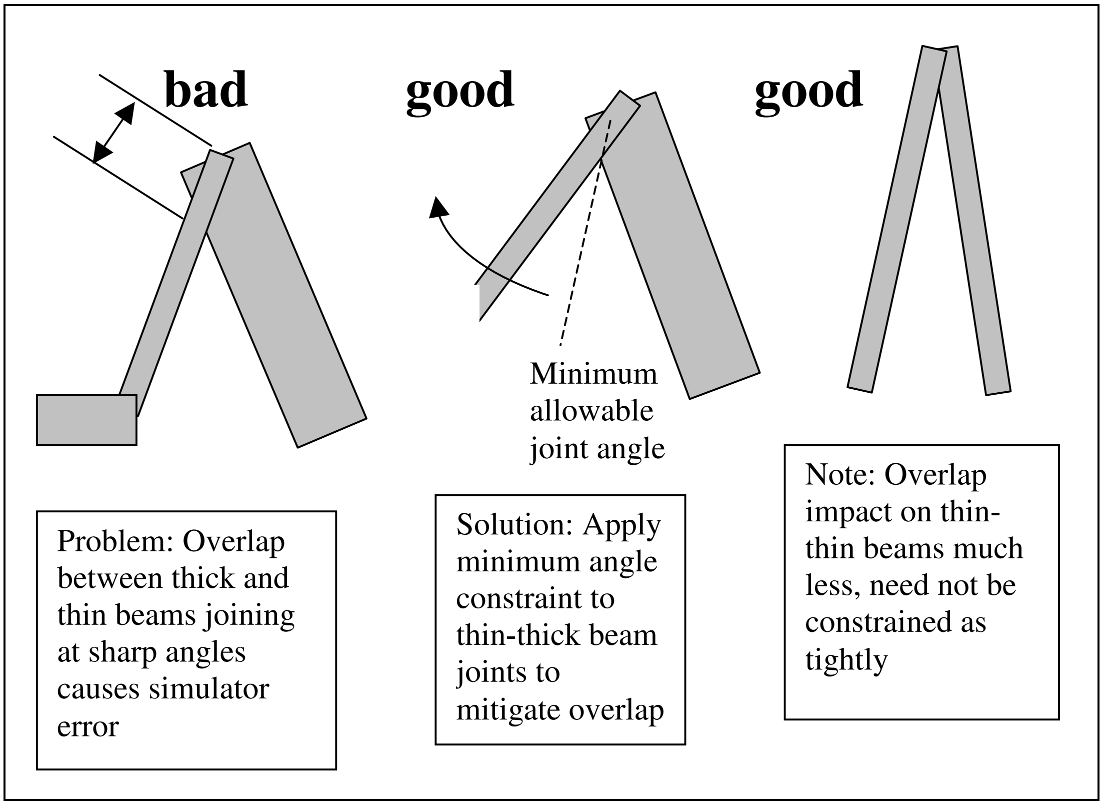

bad
good
bad
Center Mass
Center Mass
Center Mass
Problem: Short, vertical or nearly vertical innermost beam overlapping center mass causes simulator error
Solution 1: Angle constraint on innermost beam
Solution 2: Min. beam length or width mitigates effect of overlap

Figure 7.10: Inner Beam Simulation Problem and Proposed Solutions.

The second type of beam overlap that must be avoided for accurate simulation is where two beams meet at an acute angle and the overlap changes the effective length of the compliant portion of one or both of the beams. The easiest and the most restrictive solution would be to limit sharp angles of intersection between any two beams.

However overlap is only an issue when a significant portion of a beam's length is 'subsumed' by another beam. Therefore this angle constraint can be made more effective (and less restrictive) by only being applied to junctions between thin beams and thick beams. (Figure 7.11)

As an example – the overlap when two beams 75um long and 3 um wide join at a 30 degree angle is not a significant source of error, while the overlap between a 30 um long, 3 um beam joining with a 10 um beam at 30 degrees is.



Problem: Overlap between thick and thin beams joining at sharp angles causes simulator error

Solution: Apply minimum angle constraint to thin-thick beam joints to mitigate overlap

Note: Overlap impact on thin-thin beams much less, need not be constrained as tightly

Figure 7.11: Beam Junction Simulation Problem and Proposed Solution.

7.4.2. Proposed Design Constraints/Objectives – Group 3

As described in the previous section, the measured values for group 3 designs differed significantly from simulation predictions due to residual stress. Residual stress varies greatly between MUMPs runs (anywhere from 1 to 17 MPa) in the last 15 runs, making incorporating residual stress effects on resonant frequency or other objectives inaccurate. Furthermore the additional computation power required to include this in the simulation portion of a synthesis program would slow a synthesis tool significantly (simulation engines such as SUGAR can not currently include residual stress effects at all). Therefore a better choice is to adjust the synthesis objectives to ensure that designs generated do not suffer from residual stress effects.

One simple solution would be to limit the distance between the anchor points, as is done in many conventional suspension designs created by hand. This could be done explicitly by explicitly placing the anchor positions relative to the center mass. Another implementation could place an inequality constraint on the maximum allowable distance between anchor points, or less explicitly create an additional minimization objective of ‘anchor distance’; designs that violate this constraint would be considered invalid in our GA implementation. If implemented as an additional objective, ‘anchor distance’ will be balanced with the other objectives in our multi-objective GA, however, and will not necessarily be present in Pareto designs returned by the GA (Figure 7.12).

Additionally, adding these constraints or objectives would also most likely contribute to smaller design areas. It would drive designs towards more space efficient regions of the design space.

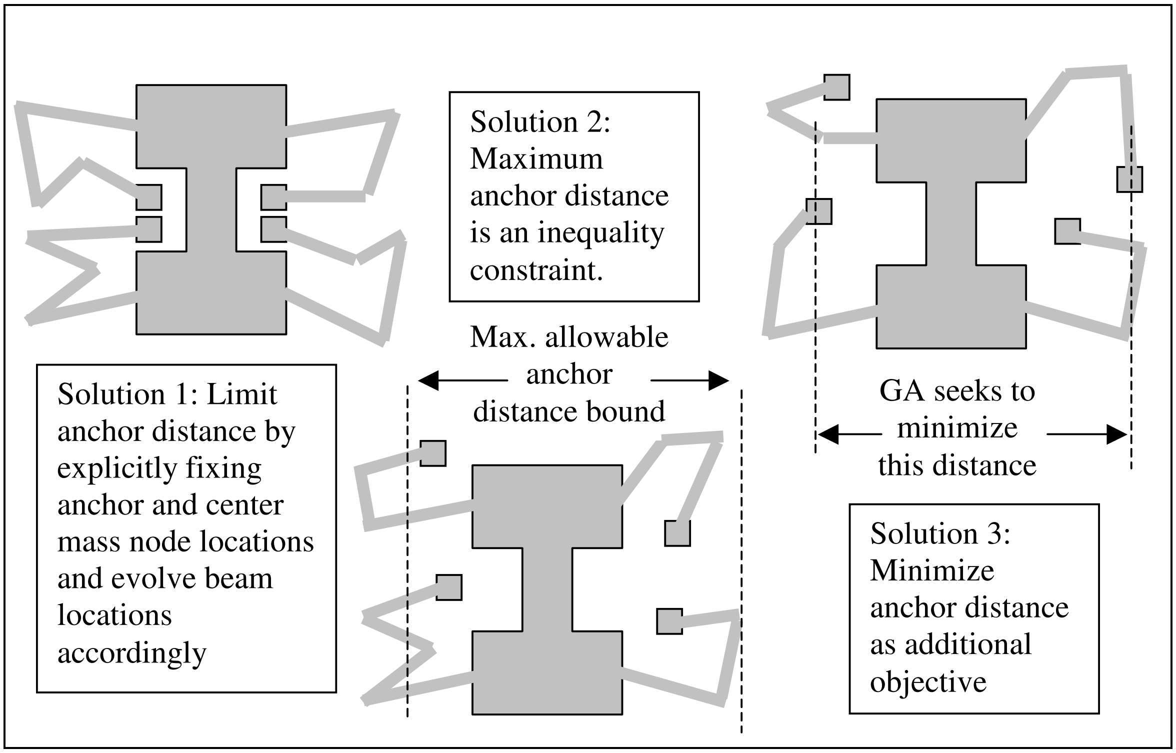

Solution 1: Limit anchor distance by explicitly fixing anchor and center mass node locations and evolve beam locations accordingly
Solution 2: Maximum anchor distance is an inequality constraint.
Max. allowable anchor distance bound
GA seeks to minimize this distance
Solution 3: Minimize anchor distance as additional objective

Figure 7.12: Proposed Anchor Distance Constraint or Objectives.

But as demonstrated by designs in group 1, (Figure 7.9), not all designs with a large anchor distance will have resonant frequency prediction issues when fabrication includes residual stress. A high lateral stiffness must be present. In the top design in Figure 7.9, for example, the residual stress was relaxed by a flexing of the meander, reducing the resonant frequency shift.

When choosing designs to fabricate in this study, we set a relatively high lateral stiffness objective (100 N/m), and only concerned ourselves with ensuring $K_{lateral} >> K_{vertical}$, not with the amount of $K_{lateral}$. Especially in the case of Manhattan angle constraints, this resulted in designs with legs going directly laterally, with little, if any meandering. In the future, $K_{lateral}$ should have a maximum constraint. This will also serve to encourage designs with more meandering, which will result in smaller design area.

Another important factor that impacts the accuracy of the designs produced is sensitivity to fabrication variation. As mentioned previously, overetch amount is not well quantified, and therefore the actual size of fabricated features cannot be precisely predicted. In our GA implementations for this and other design problems, we have set a minimum beam width of 2µm, the minimum feature size of the PolyMUMPs process. However as beam width can change as much as 0.3µm or more with overetch, beams at this minimum feature size are much more susceptible to fabrication variation. Whereas beams with a width of 10µm are much less sensitive to this variation – a 0.3 µm overetch will cause a 3% change in width for 10µm beams versus a 15% for 2µm beams.

To further explain the impact of beam variation sensitivity, we can look at its impact on suspension performance. Stiffness is a function of the second moment of area, I, which for a rectangular beam is

$$I = \frac{hw^3}{12}$$

where *h* is height of beam (in this case thickness of poly 1 layer) and *w* is the beam width. Stiffness of a beam, *k* is

$$k \propto \frac{EI}{L^3}$$

where E is the Young's modulus and L is the length of the beam. As beam width has a cubic contribution to I, which is proportional to k, a 5% reduction in beam width can cause a 15% reduction in beam stiffness.

In a simplified lumped parameter model, resonant frequency is defined as:

$$\omega = \sqrt{\frac{k}{m}}$$

where m is the suspended mass. Using this model we can see that a 5% thinning of beam width on a suspension can shift resonant frequency downwards by 7%.

Fabrication variation sensitivity can be reduced through additional constraints or objectives – increasing the minimum allowable beam width constraint from 2um to 4um for example. This would restrict the design space, requiring longer beams to achieve the same stiffness as a thinner beam could achieve.

Alternatively a more complex overall sensitivity metric can be computed from the beam widths and lengths. In the case of most of the designs evolved for this study, we find that a majority of the compliance in a suspensions comes from only a portion of the beam elements, for example 1-3 beam segments in a 7 beam symmetric resonator leg contribute 90%+ of the compliance (compliance = stiffness^{-1}) in a suspension (beams that are both long and thin). Therefore fabrication variation mitigation should be focused on these important segments. An example variation sensitivity metric could be:

$$S = \sum \frac{1}{(w_i - w_{\min})k_i}$$

where w_i and k_i are the width and stiffness of the i^{th} beam in the suspension, and w_{min} is the minimum allowable beam width. From the above equation you can see that suspensions with beam segments with very low stiffness and a small width will result in a higher S value. This fabrication variation sensitivity metric could also be considered the inverse of 'robustness' or 'fabricatability'. Therefore we could implement a constraint to limit it or an objective to minimize it, resulting in designs that are more robust to fabrication variation.

These are a few examples of ways that lessons learned through this research could be incorporated into additional constraints or objectives. Additionally a user aware of these issues could implicitly apply this information interactively to an interactive evolution synthesis.

7.4.3. Conclusion

This study has validated our approach to MEMS synthesis using multi-objective genetic algorithms using the reduced-order modeling tool SUGAR as a simulation engine. By fabricating devices produced through several types of constraints and comparing simulation predicted performance with measured resonant frequency, we were able to identify different fabrication and simulation problems that impact the success of the synthesis. We were able to classify our results into three groups: 1) designs where the measured performance matched simulation, 2) designs where simulation different from measured due to simulator limitation, and 3) designs where simulations differed from measured due to fabrication variation.

For the successfully generated designs in the first group, we were able to achieve a 95% confidence interval of +/-13%, which given the fabrication variations currently present in MEMS fabrication is reasonable for our first attempt at “closing the loop” between synthesis, simulation, fabrication and characterization.

More importantly, by studying the other two groups, we were able to identify deficiencies in our current synthesis encoding and suggest several additional constraints or objectives that will lead to better designs in the future.

Chapter 8:
Ongoing and Future Work

This chapter integrates the research in this dissertation with parallel research being performed by others, and also makes recommendations for future research directions. We will go over three areas - the expanded object-oriented data structure, advanced human interactive evolutionary synthesis and development of a knowledge base for initial designs. The expanded data structure allows for synthesis of complex MEMS structures and more advanced genetic manipulation than what was presented in chapters 5 and 6, and will facilitate the less restricted instantiation research reviewed in chapter 2. We will present the research plan to continue the human interaction work presented in chapter 4, developing different ways of combining human expertise with computational synthesis, and finally we will discuss the current work being done to develop a knowledge base to draw initial design information for evolutionary synthesis.

8.1. Object-oriented Data Structure

8.1.1. Introduction

The implementations of evolutionary synthesis for MEMS presented in this thesis, particularly the data structure to genetically encode a design and the genetic operations to modify it are case specific and thus limiting for a fundamental framework for a general MEMS synthesis tool.

The focus of this work, developed in cooperation with Sebastian Graf of Aachen University [64], is to create a general and flexible data structure for use with genetic algorithms and other evolutionary synthesis techniques. The hierarchical data structure supports an extendable MEMS design and component library and is able to handle complex connectivity between those blocks as well as design constraints and multiple objective goals.

8.1.2. Implementation

The data structure used is object based; meaning each component of a MEMS design is described and represented as an entity with its own parameters and attachment points for the connectivity with other components. Since the data structure is hierarchical, two types of design elements are defined - Higher level elements, or *clusters*, are composed of other elements, whereas the most basic elements do not contain any further components. For example basic elements include anchors, beams, plates, etc. Clusters are made up combinations of basic elements, for example a folded flexure suspension is made of multiple beam elements and an anchor. A resonating mass cluster might be made up of two folded flexure clusters along with a center mass cluster.

This hierarchical, nodal approach works well with SUGAR, which defines elements in a similar hierarchy of in its netlist, with each lumped parameter element connected at nodes. In addition to the physical attributes of a cluster of elements, the data structure also includes a list with connection points referred to as the block's external nodes. These nodes are used for connectivity with other elements (whether they be basic or clusters).

Since several blocks can share the same external node complex graph-like connectivity is possible. In addition to the attributes list and nodes list clusters provide a list with all their contained blocks. Each block also contains several flag variables, which are used to control the modifications made by the optimization routines. For example a flag can prevent the alteration of the block's attributes.

A key feature of this data structure is that it allows information about the elements and how to manipulate them to be embedded into the object for that element. For example, a beam element not only contains information on its geometry, but also can include instructions on how to mutate this element (e.g. "choose one of the following three actions: change length, change width, change angle").

In contrast the data structure used in chapter 4 and 5 of this thesis only contained geometric information for fixed basic elements, element connectivity was fixed (i.e. only linear chains of beams connected to a fixed center mass). Furthermore the genetic operations had to be specifically written for each design problem.

Using this new object-oriented structure, once a library of basic elements and clusters has been created, at the top level only the choice of what type of elements to start with as an initial design must be made for a given set of objectives. How to interface these elements with the perturbations or genetic operations is already included into the data structure for each element. This allows for much broader application to a wider range of MEMS designs.

8.1.3. Examples

The data-structure has been developed and used with MOGA for a simple two objective optimization problem similar to the work presented in section 5.3. The problem calls for the synthesis of a resonating mass with resonant frequency of 10 kHz with minimum area. In this case the center mass was set as a 100 µm x 100 µm square plate with four leg attachment points. Figure 8.1 shows output from this MOGA, in this case only basic beam elements are allowed, this type of solution is similar to those presented in this thesis, Figure 8.2 demonstrates a cyclic graph configuration not possible with the standard implementation, the addition of cross-leg beams is included as a possible mutation operation for the device as a whole. Figure 8.3 demonstrates the ability to mix different elements – in this case serpentine clusters and beam elements are both allowed in legs. Note that this is similar to the type of structure shown in Figure 2.3, except in this case the encoding structure allows this design to be modified and evolved.

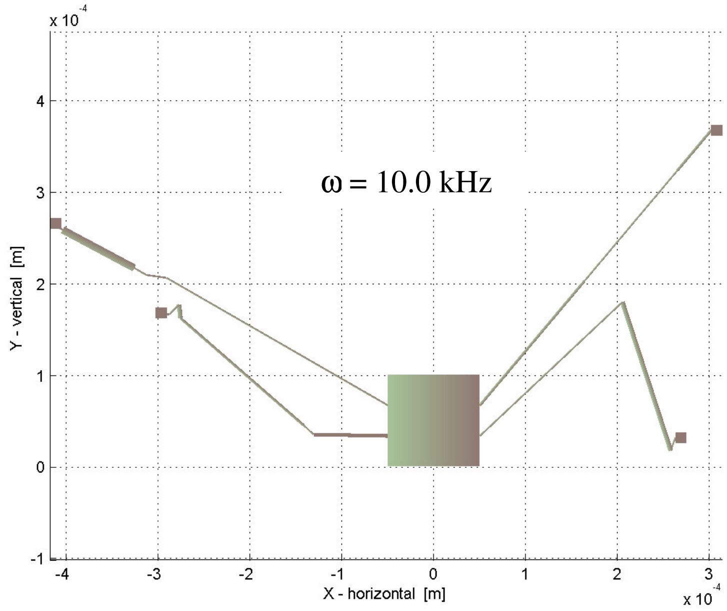


Figure 8.1: Four Leg Resonating Mass Encoded Using Object-Oriented Data Structure.

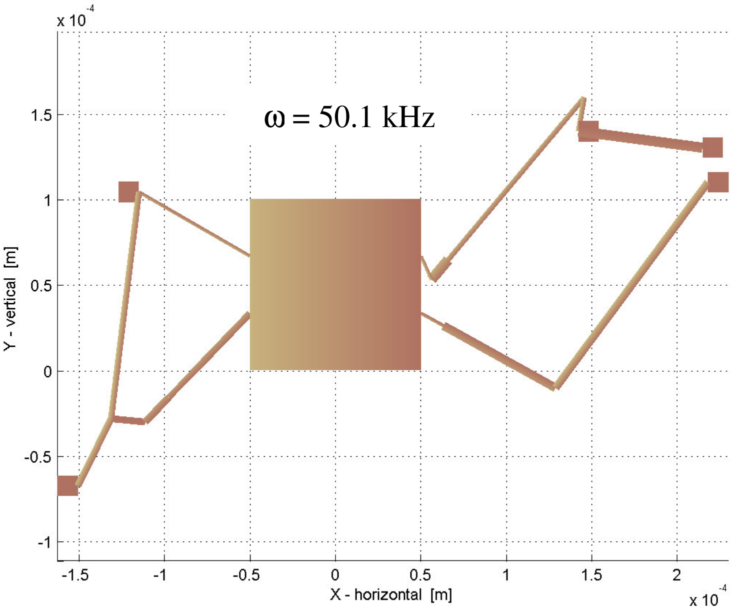


Figure 8.2: Example of Flexibility of Object-Oriented Data Structure. Cyclical graph configurations can be generated and manipulated by MOGA.

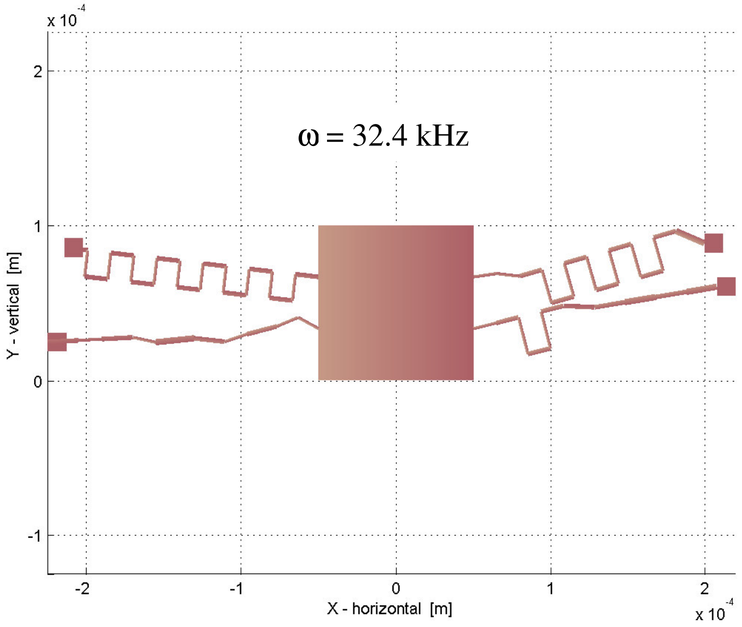


Figure 8.3: Example of Alternate MOGA Implementation Output Using Object Oriented Data Structure - four leg resonating mass where multiple element types are allowed, both serpentine spring clusters and regular beam elements are incorporated into the same leg.

8.2. Expanded IEC development

8.2.1. Alternate IEC Integration

A major thrust of the future work by the author will be in expanding the IEC work presented in chapter 4. The initial IEC presented here used a simple serial relationship between interactive and non-interactive evolution where IEC was a post-process applied after MOGA evolution. We chose the simplest, but also least powerful means of mating the two. As with all IEC work, human fatigue is still a critical factor limiting the process. As a result, we will investigate running a non-interactive evolution with periodic human intervention. This better utilizes the knowledge of the user throughout the evolution process and makes more efficient use of the tirelessness of the non-interactive approach.

We will verify this by performing user tests for both quality of output and execution time, similar to those conducted in chapter 4.

8.2.2. Online Knowledge Embedding

The second goal of our collaboration will be to extend the forefront of IEC research by introducing the concept of "Online Knowledge Embedding" in our design tools. One of the key drawbacks of evolutionary design in the analog integrated circuit field has been the inability to focus the evolution on specific parts of a design while leaving the rest of the design fixed. Takagi has used online knowledge embedding in his work with simple photomontage systems [65,66], which use IEC to evolve a picture of a human face for tasks such as identifying criminals. This allows the user (e.g. a witness to a crime) to fix facial features that they feel are a good match and continue evolving the rest of the face. By reducing the number of variables involved, this allows the evolutionary algorithm to more quickly converge on the best solution. Configurational synthesis problems like MEMS design synthesis are much more complex, for example our resonating mass case study problem, synthesizing a MEMS resonator, involves up to 84 continuous variables, so the ability to reduce variables is highly desirable.

We have split the development of online knowledge embedding in our IEC MEMS synthesis tools into two stages – the first is simpler and works similar to the photomontage system, it allows the human to fix or 'freeze' specific elements of designs. In our case study example this would include the geometry (angles, lengths, and/or widths) of some or all of the beams that make up the suspension of the MEMS resonator.

The human user will be able to focus the IEC's attention on specific parts of the MEMS device, reducing the computational complexity of the problem and allowing it to synthesis better designs faster.

The second stage for online knowledge embedding takes the human interaction one step further. It would allow the human to suggest directly to the IEC how to evolve a given design from one generation to another. In the photomontage example, this would be similar to a witness suggesting to a police sketch artist to lengthen the hair or increase the size of the nose on a sketch. Applying this concept to IEC synthesis would be highly useful not only for MEMS but other IEC applications. Often when using our IEC MEMS tool, a user would evolve a design that was good, but had perhaps one flaw in it. Using this more advanced knowledge embedding, the user could specifically tell the evolutionary algorithm how to improve the design, rather than relying purely on stochastics to randomly happen upon the correct choice. In effect the human user is able to direct the evolution, giving the evolution process a 'push' in a specific direction. This type of online knowledge embedding will have the benefit of being able to more efficiently evolve a more population of designs, as the human has a more hands on role in the evolution of each design throughout the evolution process and can avoid spending resources evolving in less fruitful directions. User tests will be performed to validate the effectiveness of both stages of knowledge embedding IEC through comparison with the previous IEC and non-interactive evolutionary algorithms developed.

8.2.3. Artificial Intelligence Assisted IEC

The third goal is to further build upon the first two goals by incorporating artificial intelligence concepts into IEC. As mentioned previously, human fatigue is the key limitation of IEC. To avoid this, we would like to be able to train a computer program to recreate the decision making process of a human expert. If we were dealing with only one specific synthesis problem using our basic IEC implementation, it is feasible to create an expert system comprised of specific functions to recreate the decision of a human user based on the geometry of a design. But this solution is impractical if the problem formulation is subject to change. A better solution is to create a learning agent. An agent can be trained by observing the human's interaction with the software and learn to reproduce it, eventually taking over a larger and larger share of the decision-making burden from the human. One of the first approaches we will investigate is the use of ensemble learners. We will also study other machine learning techniques, such as support vector machines and neural networks.

8.2.4. Application of IEC to Real-World MEMS problems

For the application aspect of the project, the goal is to make a concrete demonstration of IEC's effectiveness in MEMS design. Several companies developing MEMS have expressed interest in using automated synthesis to improve their products. We would like to use the many aspects of our IEC MEMS software developed throughout the research phase of this project to prove that a user can synthesize better designs than currently produced by traditional design methods. This would entail identifying one or more commercial MEMS products, such as accelerometers, gyroscopes, or micro-relays. Then apply the IEC software to synthesize devices using the same application specifications and design constraints. By comparing several metrics such as sensitivity, device area, power required, measurement range, etc a definitive evaluation can be made about our tools' ability to produce superior examples of 'real-world' devices.

8.3. Case Based Initial Design Generator

Figure 1.1 and Figure 1.2 presented our overall framework for MEMS synthesis, in addition to the components described in this thesis; another part that is currently being developed is a case-based library from which a synthesis algorithm can draw initial designs. The synthesis work presented in this thesis uses randomly generated designs for fixed configurations for all application examples. Essentially a user has to derive their own encoding scheme and general design configuration and randomly generate instances for that configuration. A more efficient method is to use a knowledge base to draw similar examples from for both the encoding and initial configurations.

The ability to draw upon past design knowledge is advantageous to the designer, as it permits them to generate useful MEMS designs without the same depth of knowledge in this domain. Furthermore better configurations can be generated faster as we will not need to revolve 'from scratch' every time. Case-based reasoning can utilize past successful MEMS designs and sub-assemblies as building blocks stored in an indexed library. Reasoning tools will then find cases in the library with solved problems similar to the current design problem and propose an approximate solution.

Chapter 9: Conclusions

In this thesis, several important developments in evolutionary synthesis of MEMS have been presented. Different stochastic and non-stochastic synthesis approaches including genetic algorithms, simulated annealing and gradient methods have been presented and compared. The use of human interactive evolutionary synthesis for MEMS has also been demonstrated; user test results demonstrate that by combining automated and interactive synthesis, better MEMS structures can be generated.

The applicability of our evolutionary synthesis is demonstrated for several example surface micromachined suspension applications, including accelerometers and gyroscopes. Details of instantiation, design objectives, constraints and genetic encoding structure are discussed as well. A case study of the impact of constraints on MEMS resonator synthesis has shown that due to the high dimensionality of the designs space, symmetry and angle constraints may be required to reach better solutions in a reasonable amount of time.

Our approach to the evolutionary synthesis of MEMS has been validated through the characterization of fabricated designs. We were able to fabricate resonating mass structures with measured performance close to our predicted performance. More importantly we developed new design rules that will improve our design constraints and objectives for future synthesis. Fabrication variation and simulator limitations remain to be outstanding issues facing evolutionary synthesis that need to be addressed in future synthesis tools.

Finally we presented ongoing and future directions for research in MEMS design synthesis. An object oriented data structure for describing more complex MEMS topologies for evolutionary optimization is presented. Development of a case-based library to store initial design information is presented as well. A research plan for building on the initial human interactive evolution presented in this thesis is also presented. This work will better utilize the tirelessness of automated synthesis with the expertise of a human designer to generate better MEMS designs.

As mentioned in chapter 1, MEMS based technology promises to bring a revolution to world we live in just as the integrated circuit has done in recent decades; better design tools are critical to this revolution. It is hoped that this thesis presents a more complete picture of the utility of design synthesis for MEMS and can inspire future research into other aspects of the application of design automation to the field of MEMS.

Bibliography:

[1] Lucyszyn, S., “Review of Radio Frequency Microelectromechanical Systems Technology”, *IEEE Proceeding of Sci. Meas. Technol*. vol. 151, no. 2, March 2004.

[2] Walker, S.J., Nagel, D.J., “Optics and MEMS”, Naval Research Lab Report, NRL/MR/6336-99-7975, 1999.

[3] Jacobson, S. A., Epstein, A.H., “An Informal Survey Of Power MEMS”, *The International Symposium on Micro-Mechanical Engineering*, Dec. 1-3, 2003, Tsukuba, Japan

[4] Brown, E.R., “RF-MEMS Switches for Reconfigurable Integrated Circuits”, *IEEE Transactions On Microwave Theory And Techniques,* vol. 46, no. 11, November 1998, pp. 1868-1880.

[5] Howe, R.T., Boser, B.E., Pisano, A. P., “Polysilicon integrated microsystems: technologies and applications,” *Sensors and Actuators A*, 56, 167-177, 1996.

[6] Bustillo, J. M., Fedder, G. K., Nguyen, C. T-C., Howe, R.T., “Process Technology for the Modular Integration of CMOS and Microstructures,” *Microsystem Technology*, vol 1, 1994, pp. 30-41.

[7] Fennimore A.M., Yuzvinsky T.D., Han, W.Q., Fuhrer M.S., Cumings J., Zettl A., “Rotational Actuators Based On Carbon Nanotubes.” *Nature*, vol 424, pp. 408-410, 2003.

[8] MEMSCAP Manufacturing Services, http://www.memscap.com/memsrus/crmumps.html

[9] MEMX, http://www.memx.com/

[10] Senturia, S.D., *Microsystem Design*, Kluwer Academic Publishers, 2001.

[11] Stark, B., “MEMS Reliability Assurance Guidelines for Space Applications,” Jet Propulsion Laboratory Report JPL 99-1, http://parts.jpl.nasa.gov/docs/JPL PUB 99-1.pdf

[12] ANSYS MEMS Initiative, http://www.ansys.com/applications/mems/

[13] ALGOR, MEMS Simulation, http://www.algor.com/products/applications/mems/default.asp

[14] ABAQUS Alliances: Coventor, http://www.hks.com/alliances/alliances_coventor.html

[15] SUGAR, Simulation Research for MEMS, http://www-bsac.eecs.berkeley.edu/cadtools/sugar/sugar/

[16] Bindel, D., Clark, J.V., Zhou N., Bhave, S, Bai Z., Demmel J., Pister, K. S. J., "Sugar: Advancements in a 3D Multi-Domain Simulation Package for MEMS." *Proceedings of the Microscale Systems: Mechanics and Measurements Symposium.* Portland, OR, June 4, 2001.

[17] Nise, N.S., *Control Systems Engineering*, Wiley and Sons, Inc., New York, NY, 1995.

[18] Antonsson, E. K., and J. Cagan, eds., *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge, 2001.

[19] Zhou, N., “Simulation and Synthesis of Microelectromechanical Systems”, Doctoral thesis, UC Berkeley, 2002.

[20] Ghaffarian, R.,“Thermal and Mechanical Reliability of Five COTS MEMS Accelerometers” *NASA Electronic Parts and Packaging Program EEE Links Quarterly*, February 2002.

[21] Stiny, G., “Introduction to Shape and Shape Grammars”, *Environment and Planning B: Planning and Design*, vol. 7, 1980, pp. 343-351.

[22] Koning, H., Eizenberg, J., “The Language of the Prairie: Frank Lloyd Wright’s Prairie Houses,” *Environment and Planning B: Planning and Design*, vol. 8, 1981, pp. 295-323.

[23] Agarwal, M. and Cagan, J., "Systematic form and function design of MEMS resonators using shape grammars." *ICED'99*, Technische Universität München. 1999.

[24] Agarwal, M., Cagan, J., and Stiny, G., "A micro language: generating MEMS resonators using a coupled form-function shape grammar," *Environment and Planning B: Planning and Design*, vol. 27, 2000, pp. 615-626.

[25] Pramod, J., "A Vector Quantization Multistart Method for Global Optimization", Doctoral Thesis, UC Berkeley, 1989.

[26] Weisstein, E.W., "Stochastic Optimization." MathWorld website, http://mathworld.wolfram.com/StochasticOptimization.html

[27] Spall, J.C., "An Overview of the Simultaneous Perturbation Method for Efficient Optimization," *Johns Hopkins APL Technical Digest*, vol. 19, 1998, pp. 482–492.

[28] Deb K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley and Sons, Inc., New York, NY, 2001

[29] van Laarhoven, P. J. M., Aarts, E. H. L., *Simulated Annealing: Theory and Applications*, Reidel Publishing Company, Dordrecht, Holland, 1987.

30] Bock, R.K., The Data Analysis BriefBook, online edition, http://rkb.home.cern.ch/rkb/titleA.html, 1998

[31] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989

[32] Tamaki, H., Kita, H., Kobayashi S., “Multi-Objective Optimization by Genetic Algorithms: A Review”, *Proc. of 1996 IEEE Int. Conf. on Evolutionary Computation (ICEC'96)*, 1996, pp. 517-522.

[33] Coello Coello, C.A., “An Updated Survey of Evolutionary Multiobjective Optimization Techniques : State of the Art and Future Trends,” *1999 Congress on Evolutionary Computation*, Vol. 1, Washington, D.C., July 1999.

[34] Skaar, S. E., Nilssen, R., “Genetic Optimization of Electric Machines, a State of the Art Study”, *Proceedings of Nordic Workshop on Power and Industrial Electronics*, Trondheim, Norway, 14-16 June, 2004.

[35] Zhou, N., Zhu, B., Agogino, A. M., Pister, K. S. J., “Evolutionary Synthesis of MEMS MicroElectronicMechanical Systems Design,” *Proc. of the Artificial Neural Networks in Engineering (ANNIE2001)*, 2001, pp.197-202.

[36] Li, H. and Antonsson, E.K., “Evolutionary Techniques in MEMS Synthesis’’, *Proc. DETC'98, 1998 ASME Design Engineering Technical Conferences*, Atlanta, GA 1998.

[37] Ma, L., Antonsson, E.K., “Robust Mask-Layout Synthesis for MEMS,” *Technical Proceedings of the 2001 International Conference on Modeling and Simulation of Microsystems*, ch 5: Optimization, 2001, pp. 128-131.

[38] Fan, Z. E. Goodman, J. Hu, R.C. Rosenberg and K. Seo, “System-Level Synthesis of MEMS via Genetic Programming and Bond Graphs,” *Proceedings of the 20003 Genetic and Evolutionary Computation Conference, GECCO-2003*, 2003.

[39] Zhou, N., Agogino, A. M., "Automated Design Synthesis for Micro-Electro-Mechanical Systems (MEMS)", *Proceedings of the ASME Design Automation Conference*, Sept. 29-Oct. 2 2002, Montreal, Canada.

[40] Fedder, G.K, Simulations of Microelectromechanical Systems, Doctoral thesis, UC Berkeley, 1994.

[41] Kamalian, R., Zhou, N., Agogino, A.M., "A Comparison of MEMS Synthesis Techniques", *Proceedings of the 1st Pacific Rim Workshop on Transducers and Micro/Nano Technologies*, Xiamen, China, July 22-24, 2002, pp 239-242.

[42] Vale, C.A.W. and K. Shea, “Learning Intelligent Modification Strategies in Design Synthesis”. *Computational Synthesis: From Basic Building Blocks to High Level Functionality, Papers from the 2003 AAAI Symposium*, 24-26 March, Stanford, CA, 2003, pp. 247-254.

[43] Vale, C.A.W. and K. Shea, “MEMS Design Synthesis using a Connected-Node Design Representation”, unpublished report, 2003.

[44] Campbell, M., J. Cagan and K. Kotovsky, "The A-Design Approach to Managing Automated Design Synthesis," *Research in Engineering Design*, vol. 14, no. 1, 2003, pp. 12-24.

[45] Campbell, M.I, “The A-Design Invention Machine: A Means of Automating and Investigating Conceptual Design”, PhD Thesis, Carnegie Melon University, Pittsburgh, PA, 2000, pp. 129-144.

[46] Mukherjee, T., Zhou, Y., Fedder, G. K., “Automated Optimal Synthesis of Microaccelerometers”, *Technical Digest of the 12th IEEE International Conference on Micro Electro Mechanical Systems (MEMS '99)*, 1999, pp. 326-331.

[47] Jing, Q., Luo, H., Mukherjee, T., Carley, L. R., Fedder, G. K., “CMOS Micromechanical Bandpass Filter Design Using a Hierarchical MEMS Circuit Library”, *Proceedings of the 13th IEEE International Conference on Micro Electro Mechanical Systems (MEMS 2000),* 2000, pp. 187-192.

[48] Han, J.S., Ko, J.S., Kim, Y.T., Kwak, B.M., “Parametric Study And Optimization Of A Micro-Optical Switch With A Laterally Driven Electromagnetic Microactuator,” *Journal of Micromechanics and Microengineering*, 2002, pp.939-947

[49] Fedder, G.K., Iyer, S., and Mukherjee, T., "Automated Optimal Synthesis of Microresonators," *9th Int'l Conf. on Solid State Sensors and Actuators (Transducers '97),* vol. 2, June, 1997, pp. 1109-1112.

[50] Sedivec, P., “Robust Optimization in MEMS,” Masters thesis, UC Berkeley, December 2002.

[51] Mathworks MATLAB and SIMULINK for Technical Computing, http://www.mathworks.com/

[52] Takagi, H., “Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation”, *Proceedings of the IEEE,* vol. 89, no. 9, 2001, pp. 1275-1296.

[53] Siegel, S. and Castellan, N.J., *Nonparametric Statistics for the Behavioral Sciences* (2nd ed.), London: McGraw-Hill, 1988.

[54] ANOVA: ‘ANalysis Of VAriance between groups’, http://www.physics.csbsju.edu/stats/anova.html.

[55] Samuels, H., “Single- and Dual-Axis Micromachined Accelerometers”, *Analog-Dialogue*, vol.30, no. 4, 1996.

[56] Clark, W. A., Howe, R. T., Horowitz, R. T. , “Surface Micromachined Z-axis Vibratory Rate Gyroscope,” *7th Solid-State Sensor and Actuator Workshop*, Hilton Head Island, S. C., June 2-6, 1996, pp. 299-302.

[57] Clark, W.A., “Micromachined Vibratory Rate Gyros”, Doctoral thesis, UC Berkeley, 1998.

[58] Acar C., Shkel A. M "Non-Resonant Micromachined Gyroscopes with Structural Mode-Decoupling" *Int. Conf. On Modeling and Simulation of Microsystems (MSM'2003)*, San Francisco, CA, March 2003.

[59] Sharpe, W.N., Yuan, B., Vaidyanathan, R., Edwards, R.L., "Measurements of Young's modulus, Poisson's ratio, and Tensile Strength of Polysilicon", *Proceedings of the Tenth IEEE International Workshop on Microelectromechanical Systems*, Nagoya, Japan 1997, pp. 424-429.

[60] Lin, G., Lawton, R.A., “3D MEMS in Standard Processes: Fabrication, Quality Assurance, and Novel Measurement Microstructures”, JPL Report, http://parts.jpl.nasa.gov/docs/Lin2.pdf.

[61] MEMSCAP, PolyMUMPS Run Data, Run #60, http://www.memscap.com/memsrus/svcsdata.php?RunID=60&CurrentPage=6

[62] Galef, A. E., "Bending Frequencies of Compressed Beams," *J. Acoust. Soc. Am.*, 44(8), 1968

[63] Popov, E.P., *Engineering Mechanics of Solids*, Prentice Hall, New Jersey, 1990.

[64] Graf, Sebastian, “GA Building Blocks and Data Structures for MEMS/NEMS Design Automation and Synthesis,” Masters thesis, Aachen University, Germany, 2004.

[65] Takagi, H. and Kishi, K., "On-line Knowledge Embedding for Interactive EC-based Montage System," *Third Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems (KES'99),* Adelaide, Australia, 1999, pp.280-283.

[66] Takagi, H. "Active User Intervention in an EC Search," *International Conference on Information Sciences (JCIS2000),* Atlantic City, NJ, 2000 pp.995-998.