Computational Synthesis of Multi-Domain Systems

Zhun Fan*, Kisung Seo*, Ronald C. Rosenberg+, Jianjun Hu*, Erik D. Goodman*

*Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University
+Department of Mechanical Engineering, Michigan State University

Abstract

Several challenging issues have to be addressed for automated synthesis of multi-domain systems. First, design of interdisciplinary (multi-domain) engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behavior. Second, a mechanism is needed to automatically select useful elements from the building block repertoire, construct them into a system, evaluate the system and then reconfigure the system structure to achieve better performance. Dynamic system models based on diverse branches of engineering science can be expressed using the notation of bond graphs, based on energy and information flow. One may construct models of electrical, mechanical, magnetic, hydraulic, pneumatic, thermal, and other systems using only a rather small set of ideal elements as building blocks. Another useful tool, genetic programming, is a powerful method for creating and evolving novel design structures in an open-ended manner. Through definition of a set of constructor functions, a genotype tree is created for each individual in each generation. The process of evaluating the genotype tree maps the genotype into a phenotype -- i.e., to the abstract topological description of the design of a multi-domain system, using a bond graph along with parameters for each component, if needed. Finally, physical realization is carried out to relate each abstract element of the bond graph to corresponding components in various physical domains. To implement the above GPBG approach in a specific application domain, cautious steps have to be taken to make the evolved design represented by bond graphs realizable and manufacturable. To achieve this, one important step is to define appropriate building blocks of the design space and carefully design a realizable function set in genetic programming. We are going to illustrate this in an example of behavioral synthesis of a RF MEM circuit – a micro-mechanical band pass filter design. Finally, we have some discussions on how to extend the above approach to an integrated evolutionary synthesis environment for MEMS across a variety of design layers.

1. Introduction

Design automation is undoubtedly a very difficult task. However, we have some very successful application examples. Much research has been done on design automation of single domain systems using evolutionary computation approach. For example, automated design of analog circuits has attracted much attention in recent years (Grimbleby 2000; Lohn 1999; Koza *et al* 1999; Fan *et al* 2001). They could be classified into two categories: GA-based and GP-based. Most GA-based approaches realize topology optimization via a GA and parameter optimization with numerical optimization methods (Grimbleby 2000). Some GA approaches also evolve both topology and component parameters; however, they typically allow only a limited amount of components to be evolved (Lohn 1999). Although their works basically achieve good results in analog circuit design, they are not easily extendable to interdisciplinary systems like mechatronic systems.

Several challenging issues have to be addressed for automated synthesis of multi-domain systems. First, design of interdisciplinary (multi-domain) engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behavior. Second, a mechanism is needed to automatically select useful elements from the building block repertoire, construct them into a system, evaluate the system and then reconfigure the system structure to achieve better performance. It is a remarkable fact that models based on apparently diverse branches of engineering science can be expressed using the notation of bond graphs, based on energy and information flow. Using that language, one may construct models of electrical, mechanical, magnetic, hydraulic, pneumatic, thermal, and other systems using only a rather small set of ideal elements as building blocks. As a special form of evolutionary computation, genetic programming is a powerful approach to creating and evolving novel design structures in an open-ended manner. Through definition of a set of constructor functions, a genotype tree is created for each individual in each generation. The process of evaluating the genotype tree maps the

genotype into a phenotype -- i.e., to the abstract topological description of the design of a multi-domain system, using a bond graph along with parameters for each component, if needed. Finally, physical realization is carried out to relate each abstract element of the bond graph to corresponding components in various physical domains. The above approach, combining bond graphs and genetic programming, has led to several successful design results by computational synthesis. The first is a domain-independent eigenvalue placement design problem that is tested for some sample target sets of eigenvalues (Seo *et al* 2001). The second is in the electrical domain – design of analog filters to achieve specified performance over a given frequency range (Fan *et al* 2001). The third is in the electromechanical domain – redesign of a printer drive system to obtain desirable damping of the position of a rotational load (Fan *et al* 2002).

We are going to extend our approach to synthesize MEMS (Micro Electro Mechanical Systems). Due to their multi-domain and intrinsically three-dimensional nature, design and analysis of MEMS is very complicated and requires access to simulation tools with finite element analysis capability, like Conventorware and ANSYS. Computation cost is typically very high. A common representation that encompasses multiple energy domains is thus needed for modeling of the whole system. We need a system-level model that reduces the number of degrees of freedom from the hundreds and thousands of degrees of freedom characterizing the meshed 3-D model to as few as possible. The bond graph, based on power flow, provides a unified model representation across inter-disciplinary system domains and is also compatible with 3-D numerical simulation and experimental results in describing the macro behavior of the system, so long as suitable lumping of components can be done to obtain lumped-parameter models. It can be used to represent the behavior of a subsystem within one energy domain, or the interaction of multiple domains. Therefore, the first important step in our method of MEMS synthesis is to develop a strategy to automatically generate bond graph models to meet particular design specifications on system level behaviors.

For system-level design, hand calculation is still the most popular method in current design practice. This is for two reasons: 1) The MEMS systems we are considering, or designing, are relatively simple in dynamic behaviors -- especially the mechanical parts -- largely due to limitation in fabrication capability. 2) There is no powerful and widely accepted synthesis approach to design multi-domain systems automatically.

The GP/BG approach, which combines the capability of genetic programming to search in an open-ended design space, and the merits of bond graphs for representing and modeling multi-domain systems elegantly and effectively, proves to be a promising method to do system-level synthesis of multi-domain dynamical systems, including MEMS. In the first or higher level of system synthesis, our GPBG approach can help to obtain a high-level description of a system that assembles the system from a library of existing components in an automatic manner to meet a predefined design specification. Then in the second or lower level, other numerical optimization approaches (Zhou 1998), as well as evolutionary computation, may be used to synthesize custom components from a functionality specification. It is worthwhile to point out that for the system designer, the goal of synthesis is not necessarily to design the optimum device, but to take advantage of rapid prototyping and "design reuse" through component libraries; while for the custom component designer, the goal may be maximum performance. These two goals may lead to different synthesis pathways.

However, in trying to establish an automated synthesis approach for MEMS, we should take cautious steps. Due to the limitation of fabrication technology, there are many constraints in design of MEMS. Unlike in VLSI, which can draw on extensive sets of design rules and programs that automatically test for design-rule violations, the MEMS field lacks design verification tools at this time. This means that no design automation tools are available at this stage capable of designing and verifying any kind of geometrical shapes of MEMS devices. Thus, automated MEMS synthesis tools must solve sub-problems of MEMS design in particular application domains for which a small set of predefined and widely used basic electromechanical elements are available, to cover a moderately large functional design space.

Automated synthesis of a RF MEM device, namely, micro-mechanical band pass filter is taken as an instance in this paper. As designing and micromachining of more complex structures is a definite trend, and research into micro-assembly is already on its way, the GP/BG approach is believed to have many potential applications. More work to extend the above approach to an integrated evolutionary synthesis environment for MEMS across a variety of design layers is also discussed in the end.

2. Design Methodology

2.1 Bond Graphs

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic components, etc. (Karnopp *et al* 2000). It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design search. For notation details and methods of system analysis related to the bond graph representation see Karnopp et al. and Rosenberg (Rosenberg *et a* 1993). Much recent research has explored the bond graph as a tool for design (Youcef-Toumi 1999).

Bond graphs have four embedded strengths for design applications, namely, the wide scope of systems that can be created because of the multi- and inter-domain nature of bond graphs, the efficiency of evaluation of design alternatives, the natural combinatorial features of bond and node components for generation of design alternatives, and ease of mapping to the engineering design process. Those attributes make bond graphs an excellent candidate for modeling and design of a multi-domain system.

2.2 Bond Graph and Genetic Programming

Genetic programming is an extension of the genetic algorithm, using evolution to optimize actual computer programs or algorithms to solve some task (Holland 1975, Goldberg 1989), typically involving a graph-type (or other variable-length) representation. The most common form of genetic programming (Koza *et al*, 1994) uses trees to represent the entities to be evolved. Genetic programming can manipulate variable-sized strings and can be used to "grow" trees that specify increasingly complex bond graph models. The tree representation on GP chromosomes, as compared with the string representation typically used in GA, gives GP more flexibility to encode solution representations for many real-world design applications. The bond graph, which can contain cycles, is not represented directly on the GP tree—instead, the function set (nodes of the tree) encodes a constructor for a bond graph.

Defining of a proper function set is one of the most significant steps in preparing a genetic programming run. It may affect both the search efficiency of genetic programming and validity of evolved results and is closely related to the selection of building blocks for the system being designed. In this research, a basic function set and modular function set are presented and listed in table 1 and table 2. Operators in the basic function set basically aim to construct primitive building blocks for the system, while operators in the modular function set purport to construct relatively modular and predefined building blocks composed of primitive building blocks. Notice that numeric functions are included in both function sets, as they are needed in both cases. In other research, we hypothesize that usage of modular operators in genetic programming has some implications in improving its search efficiency. However, in this paper, we concentrate on another issue, proposing the concept of a realizable function set. By using only operators in a realizable function set, we seek to guarantee that the evolved design is physically realizable and has the potential to be manufactured. This concept of realizability may include stringent fabrication constraints to be fulfilled in some specific application domains. This idea is to be illustrated in the design example of an RF MEM device, namely, a micro-mechanical band pass filter.

Examples of modular operators, namely insert_BU and insert_CU operators, are illustrated in figure 1 and figure 2. Examples of basic operators are available in our earlier work (Seo *et al* 2001).

Basic Function Set	
add_C	Add a C element to a junction
add_I	Add a I element to a junction
add_R	Add a R element to a junction
insert_J0	Insert a 0-junction in a bond
insert_J1	Insert a 1-junction in a bond
replace_C	Replace the current element with a C
replace_I	Replace the current element with a I
replace_R	Replace the current element with a R
+	Add two ERCs
-	Substract two ERCs
enda	End terminal for add functions
endi	End terminal for insert functions
endr	End terminal for replace functions
erc	Ephemeral Random Constant (ERC)

Table 1. Operators in Basic Function Set

Modular Function Set	
insert_RU	Insert a Resonant Unit
insert_CU	Insert a Coupling Unit
insert_BU	Insert a Bridging Unit
add_RU	Add a Resonant Unit
insert_J01	Insert a 0-1-junction compound
insert_CIR	Insert a special CIR compound
insert_CR	Insert a special CR compound
Add_J	Add a junction compound
+	Add two ERCs
-	Substract two ERCs
endn	End terminal for add functions
endb	End terminal for insert functions
endr	End terminal for replace functions
erc	Ephemeral Random Constant (ERC)

Table 2. Operators in Modular Function Set

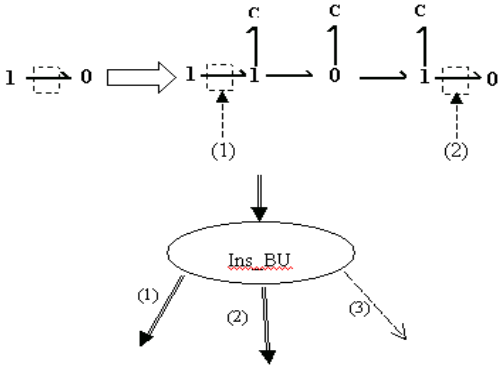

Figure 1. Operator to Insert Bridging Unit

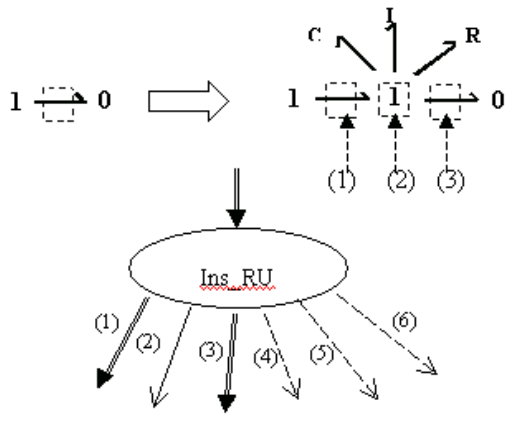

Figure 2. Operator to Insert Resonant Unit

As illustrated in figure 1, a resonant unit (RU) that composes of one I, R, and C component all attached to an 1-junction, is inserted to an original bond with modifiable site through the insert_RU function. After insert_RU function is executed, a new RU is created and one additional modifiable site, namely bond (3), appears in the resulting phenotype of bond graph along with the original modifiable site bond (1). The new added 1-junction also has an additional modifiable site (2). As component C, I, R all have parameters to be evolved, insert_RU function has three corresponding arity (4) (5) (6) for numerical evolution of parameters.

Figure 2 explains how insert_BU function works. Bridging unit (BU) is a subsystem that composes of three capacitors with the same parameters attached together with a 0-junction in the center and two 1-junctions at the left and at the right respectively. After execution of the insert_BU function, an additional modifiable site (2) appears at the rightmost newly created bond. The reason why RU and BU looks in that way is given in the next case study section.

3. Case Study

3.1 Problem Formulation

Automated synthesis of a RF MEM device, micro-mechanical band pass filters is used as an example in this paper (Wang and Nguyen 1999). Through analyzing two popular topologies used in surface micromachining of micro-mechanical filters, we found that they are topologically composed of a series of RUs and Bridging Units (BUs) or RUs and Coupling Units (CUs) concatenated together. Figure 3, 4, 5 illustrates

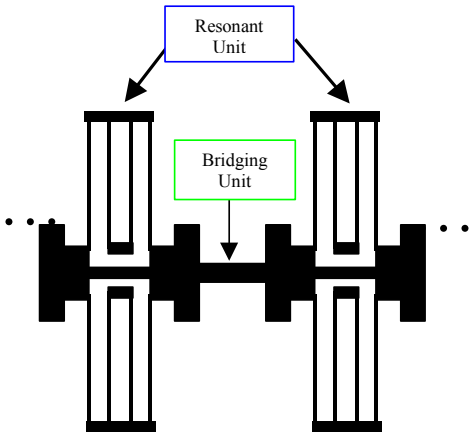

Figure 3. Layout of Filter Topology I:
Filter is composed of a series of Resonator Units (RUs) connected by Bridging Units (BUs).

the layouts and bond graph representations of filter topology I and II.

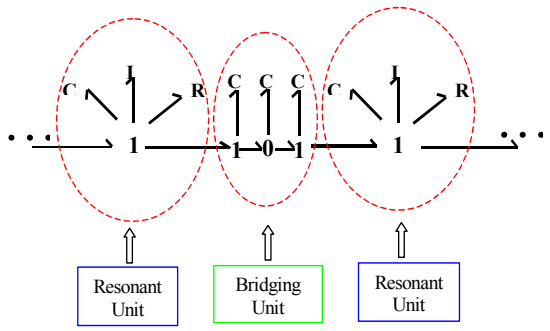

Figure 4. Bond Graph Representation of Filter Topology I

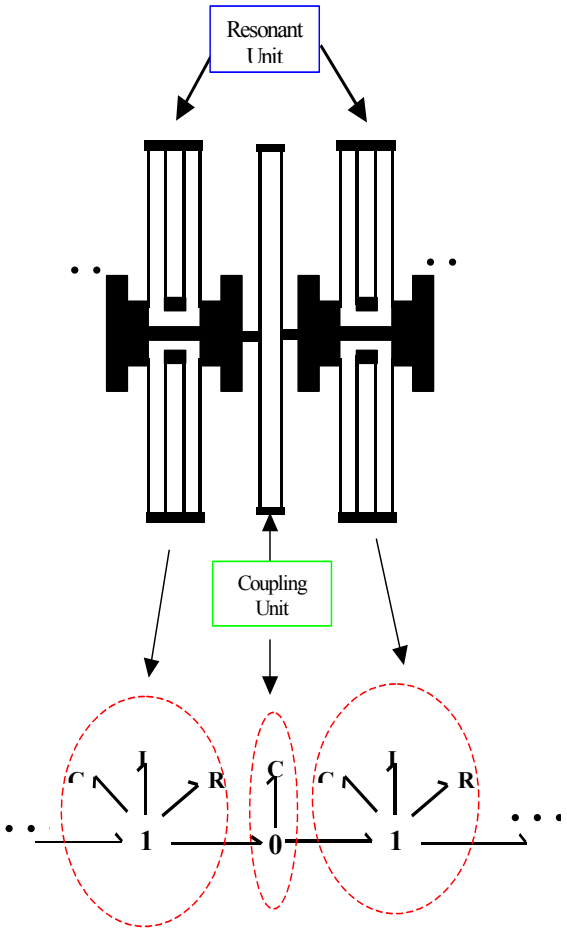

Figure 5. Layout of Filter Topology II: Filter is composed of a series of Resonator Units coupled by Coupling Units. Its corresponding bond graph representation is also shown.

3.2 Design Embryo

All individual genetic programming trees create bond graphs from an embryo. Selection of the embryo is also an important topic in system design, especially for multi-port systems. In our filter design problems, we use the following bond graph as our embryo, as shown in Figure 6.

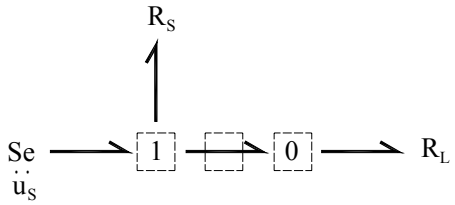

Figure 6. Embryo of Design

3.3 Function Set

GPBG is a quite general approach to automate synthesis of multidisciplinary systems. Using a basic set of building blocks, we can actually try to construct any kind of systems without constraints. However, engineering systems in the real world are confined by various constraints. So if we implement GPBG to synthesis real world engineering systems, we have to take care that those constraints can be enforced within the bounds of the approach.

Unlike our previous designs with basic function sets, which impose fewer topological constraints on design, MEMS design features relatively few devices in the component library. These devices are typically more complex in structure than those primitive building blocks used in the basic function set. Only evolved designs represented by bond graphs matching the dynamic behavior of those devices belong to the component library are expected to be manufacturable under current or anticipated technology. Thus, an important and special step in MEMS synthesis with the GPBG approach is to define a *realizable* function set that, throughout execution, can always produce phenotypes that can be built using existing or expected technology.

By analyzing the system of MEM filters from a bond graph viewpoint, we know that it is basically composed of Resonator Units (RUs) and Coupling Units (CUs). Another popular MEM filter topology includes Resonator Units and Bridging Units (BUs). It turns out that a realizable function set for these design topologies often includes functions from both the basic set and modular set. In many cases, multiple realizable function sets, rather than only one, can be used to evolve

realizable structures of MEMS. In this research, we used the following function set, along with traditional numeric functions and end operators for creating filter topologies with coupling units and resonant units.

$$\Re = \{f\_tree, f\_insert\_J1, f\_insert\_RU, f\_insert\_CU, f\_add\_C, f\_add\_R, f\_add\_I\}$$

3.4 Fitness Function

The fitness function is defined as follows.
Within the frequency range of interest, uniformly sample 100 points. Compare the magnitudes of the frequency response at the sample points with target magnitudes, which is one within the pass frequency range of [316, 1000] Hz, and zero otherwise between 0.1 and 100KHz. Compute their differences and get a sum of squared differences as raw fitness, defined as $Fitness_{raw}$. Then normalized fitness is calculated according to:

$$Fitness_{norm} = 0.5 + Norm / (Norm + Fitness_{raw})$$

3.5 Experimental Setup

We used a strongly-typed version [Luke, 1997] of lilgp [Zongker and Punch, 1996] to generate bond graph models. The major GP parameters were as shown below:

Population size: 500 in each of thirteen subpopulations
Initial population: half_and_half
Initial depth: 4-6
Max depth: 50 Max_nodes 5000
Selection: Tournament (size=7)
Crossover: 0.9 Mutation: 0.3

Three major code modules were created in our work. The algorithm kernel of HFC-GP was a modified version of an open software package developed in our research group -- lilgp. A bond graph class was implemented in C++. The fitness evaluation package is C++ code converted from Matlab code, with hand-coded functions used to interface with the other modules of the project. The commercial software package 20Sim was used to verify the dynamic characteristics of the evolved design. The GP program obtains satisfactory results on a Pentium-IV 1GHz in 1000~1250 minutes.

3.6 Experiment Results

Experiment results show strong topological search capability of genetic programming and feasibility of our GPBG approach for finding realizable design for micro-mechanical filters. Although significant fabrication difficulty is currently presented when fabricating a micro-mechanical filter with more than 3 resonators, it does not invalidate our research and topological search capability of the GPBG approach, considering its potential in exploring more complicated topologies of future MEMS design and the ever-progressing technology frontiers of MEMS fabrication.

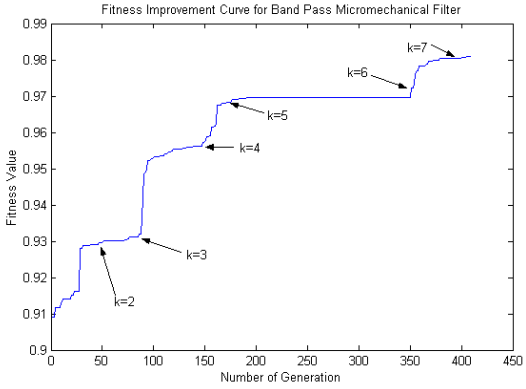

Figure 7. Fitness Improvement Curve

In figure 7 above, we define K = number of resonant units used in the filter topology. It is very obvious from the fitness improvement curve that as evolution goes on, fitness value undergoes continual improvement (Hu *et al* 2002). It is also an interesting observation that, as fitness improves, the value of K also becomes larger. This observation is supported by the fact that a higher-order system with more resonator units has the potential of better system performance than its low-order counterpart. Table 2 shows the values of K and the numbers of the generations at which K changes.

# of generations	64	98	158	164	364	409
K	2	3	4	5	6	7

Table 2. Number of Generations vs. Number of Resonator Units

The plot of corresponding system frequency responses at generations 98, 164, 364 and 409 are shown in Figure 8.

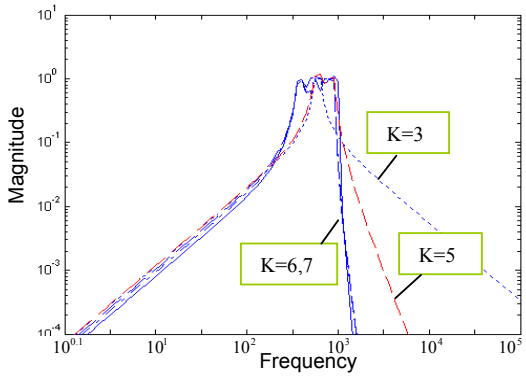

Figure 8. Plot of frequency responses of design candidates with different number of resonator units. All results are from one genetic programming run of GPBG approach.

A layout of a design candidate with three resonators and its bond graph representation are shown below in figure 9. Notice that the geometry of resonators may not show the real sizes and shapes of a physical resonator and the layout figure only serves as a topological illustration.

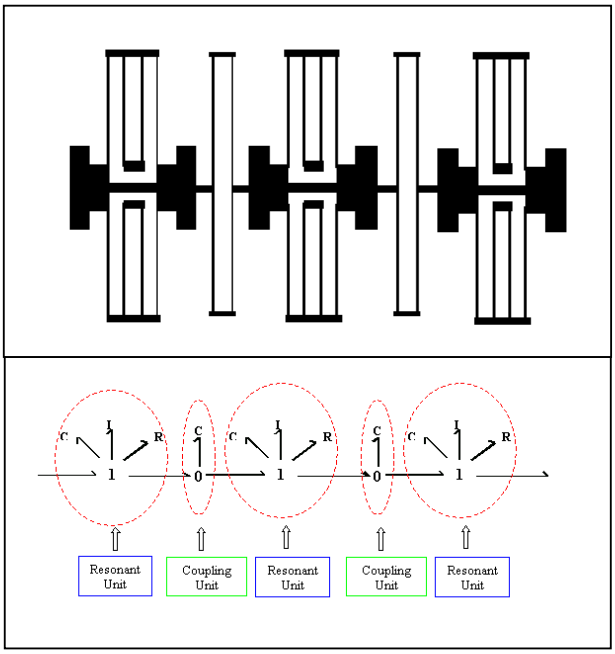

Figure 9. Layout and bond graph representation of a design candidate from the experiment with three resonator units coupled with two coupling units.

4. Extensions

In MEMS, there are two or three levels of designs that need to be synthesized. Usually the design process starts with basic capture of the schematic of the overall system, then goes on through layout and construction of a 3-D solid model. So the first design level is the system level, which includes selection and configuration of a repertoire of planar devices or subsystems. The second level is 2-D layout of basic structures like beams to form the elementary planar devices. In some cases, if the MEMS is basically a result of a surface-micro machining process and no significant 3-D features are present, design of this level will end one cycle of design. More generally, modeling and analysis of a 3-D solid model for MEMS is necessary.

For the second level -- two-dimensional layout designs of cell elements -- layout synthesis usually takes into consideration a large variety of design variables and design constraints. The most popular synthesis method seems to be based on conventional numerical optimization methods. The design problem is often first formulated as a nonlinear constrained optimization problem and then solved using an optimization software package (Zhou 1998). Geometric programming, one special type of convex optimization method, is reported to synthesize a CMOS op-amp. The method is claimed to be both globally optimal and extremely fast. The only disadvantage and limitation is that the design problem has to be carefully formatted first to make it suitable for the treatment of the geometric programming algorithm. However, all the above approaches are based on the assumption that the structures of the cell elements are relatively fixed and subject to no radical topology changes (Hershenson *et al* 2001). A multi-objective evolutionary algorithm approach is reported for automatic synthesis of topology and sizing of a MEMS 2-D meandering spring structure with desired stiffnesses in certain directions (Zhou *et al* 2001).

The third level design calls for FEA (Finite Element Analysis). FEA is a computational method used for analyzing mechanical, thermal, electrical behavior of complex structures. The underlying idea of FEA is to split structures into small pieces and determine behaviors of each piece. It is used for verifying results of hand calculations for simple model, but more importantly, for predicting behavior of complex models where 1st order hand calculations are not available or insufficient. It is especially well suited for *iterative* design. As a result, it is quite possible that we can use an evolutionary computation approach to evolve a design using evaluation by means of FEA to assign fitness. Much work in this area has already been reported and it should also be an ideal analysis tool for use in the synthesis loop for final 3-D structures of MEMS. However, even if we have obtained an optimized 3-D device shape, it is still very difficult to produce a proper mask layout and correct fabricate procedures. Automated mask layout and process synthesis tools will be very helpful to relieve the designers from considering the fabrication details and focus on the functional design of the device and system instead (Ma L., Antonsson E. K. 2000)

Our long time task of research is to include computational synthesis for different design levels, and to provide support for design engineers in the whole MEMS design process.

5. Conclusions

This paper has suggested a design methodology for automatically synthesizing system-level designs for MEMS. For design of systems like the MEM filter problem, with strong topology constraints and fewer topology variations allowed, the challenge is to define a realizable function set that assures the evolved design is physically realizable and can be built using existing or

anticipated technologies. Experiments show that a mixture of functions from both a modular function set and a basic function set form a realizable function set, and that the GPBG algorithm evolves a variety of designs with different levels of topological complexity that satisfy design specifications.

Acknowledgement

The authors gratefully acknowledge the support of the National Science Foundation through grant DMI 0084934.

References

Fan Z., Hu J., Seo K., Goodman E., Rosenberg R., and Zhang B., 2001. Bond Graph Representation and GP for Automated Analog Filter Design. *Genetic and Evolutionary Computation Conference* Late-Breaking Papers, San Francisco,: 81-86.

Fan Z., Seo K., Rosenberg R. C., Hu J., Goodman E. D., 2002. Exploring Multiple Design Topologies using Genetic Programming and Bond Graphs. *Proceedings of the Genetic and Evolutionary Computation Conference*, *GECCO-2002*, New York : 1073-1080.

Goldberg D., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley.

Grimbleby J. B. 2000. Automatic analogue circuit synthesis using genetic algoriths. *IEE Proc. – Circuits Devices Syst.* : 319-323. #

Hershenson M. M, Boyd, S. P., and Lee T.H. 2001. Optimal Design of a CMOS Op-Amp via Geometric Programming. *Computer-Aided Design of Integrated Circuits and Systems.vol* 20(1): 1-21

Holland J. H. 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Hu J., Goodman E. D., 2002. Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms. *CEC 2002*, Honolulu, Hawaii, May,

Karnopp D. C., Margolis D. L. and Rosenberg R. C. 2000. *System Dynamics: Modelling and Simulation of Mechatronic Systems. Third Edition*. New York: John Wiley & Sons, Inc.

Koza J. R., Bennett III F. H., Andre D. and Keane M. A. 1999b. The design of analogue circuits by means of genetic programming. In P. J. Bentley (ed.), *Evolutionary Design by Computers*, 365-385. London: John Wiley & Sons Ltd.

Koza J. R., 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs*, The MIT Press

Lohn J. D., Colombano S. P. 1999. A circuit representation techniques for automated circuit design. *IEEE Transactions on Evolutionary Computation*: 205-219. #

Luke S., 1997. Strongly-Typed, Multithreaded C Genetic Programming Kernel, http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.

Ma L. and Antonsson E.K. 2000. Automated Mask-Layout and Process Synthesis for MEMS. *Technical Proceedings of the 2000 International Conference on Modeling and Simulation of Microsystems* : 20-23

Paynter H. M. 1991. An epistemic prehistory of bond graphs. In P. C. Breedveld and G. Dauphin-Tanguy (ed.), *Bond Graphs for Engineers,* 3-17. Amsterdam, The Netherlands: Elsevier Science Publishers.

Rosenberg R. C., 1993. Reflections on Engineering Systems and Bond Graphs, *Trans. ASME J. Dynamic Systems, Measurements and Control*, 115: 242-251

Seo K., Goodman E., and Rosenberg R., 2001, First Steps toward Automated Design of Systems Using Bond Graphs and Genetic Programming, *Proc. Genetic and Evolutionary Computation Conference*, San Francisco, p. 189 (1-page abstract) and poster.

Wang K. and Nguyen C. T. C. 1999. *Journal of Microelectomechanical Systems*. 8(4): 534-556

Youcef-Toumi K., 1996. Modeling, Design, and Control Integration: A necessary Step in Mechatronics. *IEEE/ASME Trans. Mechatronics*, 1(1): 29-38

Zonker D., Punch W.F., 1998. lil-gp 1.1 User's Manual. GARAGe, College of Engineering, Michigan State University.

Zhou N., Zhu B., Agogino A., Pister K. 2001. Evolutionary Synthesis of MEMS design. *ANNIE 2001, IEEE Neural Networks Council and Smart Engineering System Design conference, St. Louis, MO, Nov 4-7, 2001*.

Zhou Y. Layout Synthesis of Accelerometers. 1998. *Thesis for Master of Science*. Department of Electrical and Computer Engineering, Carnegie Mellon University.