Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 579–589
Applied Soft Computing
www.elsevier.com/locate/asoc

Structured synthesis of MEMS using evolutionary approaches

Zhun Fan [a,*], Jiachuan Wang [b], Sofiane Achiche [a], Erik Goodman [c], Ronald Rosenberg [d]

[a] Technical University of Denmark, Department of Mechanical Engineering, Lyngby DK-2800, Denmark
[b] Systems Department, United Technologies Research Center, East Hartford, USA
[c] Michigan State University, Department of Electrical and Computer Engineering, East Lansing, USA
[d] Michigan State University, Department of Mechanical Engineering, East Lansing, USA

Received 13 April 2005; received in revised form 19 March 2007; accepted 2 April 2007
Available online 2 June 2007

Abstract

In this paper, we discuss the hierarchy that is involved in a typical MEMS design and how evolutionary approaches can be used to automate the hierarchical synthesis process for MEMS. The paper first introduces the flow of a structured MEMS design process and emphasizes that system-level lumped-parameter model synthesis is the first step of the MEMS synthesis process. At the system level, an approach combining bond graphs and genetic programming can lead to satisfactory design candidates as system-level models that meet the predefined behavioral specifications for designers to trade off. Then at the physical layout synthesis level, the selection of geometric parameters for component devices and other design variables is formulated as a constrained optimization problem and addressed using a constrained genetic algorithm approach. A multiple-resonator microsystem design is used to illustrate the integrated design automation idea using these evolutionary approaches.

Keywords: MEMS synthesis; Genetic programming; Bond graphs; Genetic algorithm

1. Introduction

Due to the complexity and mixed-domain intricacy involved in MEMS design, designing MEMS remains an art in most applications, requiring a large amount of investment of human resources, time and money. Much of the investment is consumed in the iterative trial-and-error design process. Automated design synthesis helps engineers to rapidly develop optimal configurations for a given set of performance and constraint guidelines, and thus to shorten typical development cycles for MEMS (with a given fabrication technology) by a large factor and to enable design of far more complex MEMS than can be handled today. Electronic Design Automation (EDA) has achieved great success in both industry and academia. However, analogous research in design automation for MEMS seems to lag far behind, although considering the close affinity of MEMS and VLSI – MEMS actually evolved from microelectronics and inherited the fabrication techniques of VLSI – the potential for successful applications of design automation to MEMS appears to be promising. Translating the key insights of silicon evolution success into MEMS technologies is a much more challenging task than most people have expected. Major research topics to be addressed include:

1. developing a broad base of building blocks in MEMS technologies so that huge networks of micro-devices can be assembled into arbitrary architectures with desirable functionalities;
2. hierarchically decomposing the MEMS design process to reduce design complexity, making it more amenable to automation;
3. improving models of computation and extending current synthesis methodologies to facilitate generation of viable design candidates and smoother transitions from conceptual and embodied designs to process fabrication;
4. combining MEMS component layout extraction and lumped-parameter simulation and design synthesis to provide MEMS designers with a VLSI-like environment, enabling faster design cycles and improved design productivity.

This paper aims to partially address these challenges. The proposed hierarchical and evolutionary design framework for

* Corresponding author. Tel.: +45 27140868.
E-mail addresses: zf@mek.dtu.dk (Z. Fan), WangJ2@utrc.utc.com (J. Wang), sac@gst.mek.dtu.dk (S. Achiche), goodman@egr.msu.edu (E. Goodman), rosenber@egr.msu.edu (R. Rosenberg).

1568-4946/$ – see front matter
doi:10.1016/j.asoc.2007.04.001

MEMS seeks to eliminate tedious and repetitive design tasks, facilitate hierarchical problem decomposition, and combine the power of multiple evolutionary computation algorithms working sequentially to generate and identify better product designs and process solutions. In particular, we divide design representations of MEMS into two levels, the system-level behavioral macromodel and the detailed-level physical geometric layout model. At the system level, we use a combination of genetic programming and bond graphs to automatically generate and search for viable design candidates represented by behavioral macromodels satisfying high-level design specifications. At the second detailed (layout) level, constrained genetic algorithms are used to optimize the geometric parameters that relate the physical device model to the behavioral macromodel and meet more detailed design objectives.

This paper is organized as follows: Section 2 introduces a structured method of MEMS synthesis presented by Antonsson [1]. Section 3 describes the GPBG approach that combines genetic programming and bond graphs for system-level behavioral synthesis. Section 4 explains how a constrained GA approach can be used to solve the second-level physical layout synthesis problem. Concluding remarks are presented in Section 5.

2. Structured MEMS design methodology

In MEMS, there are a number of levels of designs that need to be synthesized [10]. Usually the design process starts with basic capture of the schematic of the overall system, and 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 a result of a surface-micro machining process (2.5-D geometry) and no significant complex 3-D features are present, design of this level will end one cycle of design. However, more generally, modeling and analysis of a 3-D solid model for the MEMS are necessary. Furthermore, even if an optimized 3-D device shape is obtained, it is still very difficult to produce a proper mask layout and correct fabrication procedures. Some research work addressed this issue; a design tool that calculates the required 2-D mask set producing a given 3-D model by investigating the vertical topology to the model through a trial mask set was proposed in Refs. [29,30]. This work was based on the development proposed in Ref. [4], where a new process planning technique that uses a three-dimensional surface micromachined structure as input is proposed. Furthermore, automated mask layout and process synthesis tools would be very helpful to relieve designers from considering fabrication details and let them focus on the functional design of the device and system [21,23]. After a ‘‘top-down’’ design path, a ‘‘bottom-up’’ verification process is usually followed to guarantee that at each design level the design specifications are met, as defined (see Fig. 1). The ultimate goal is to develop tools for MEMS design to ensure first-pass success by having a well-defined ‘‘top-down’’ design path and ‘‘bottom-up’’ verification path.

For system-level design, hand calculation is still the most popular method in current design practice [31]. This is mainly because no powerful and widely accepted synthesis approach exists to automated design of multi-domain systems in the system level. In addition, most MEMS system-level designs are accomplished by modeling the entire microelectromechanical systems as single behavioral entities having no lower hierarchical levels. If there are any changes in geometric parameters or topology, a whole new model must be created, and this substantially lengthens design cycles. To avoid these problems, the hierarchical design approach combined with evolutionary techniques can be used to deal with only the parts that involve changes rather than the entire model and hence saving design time and energy. Evolutionary computation (EC) is a cluster of computational algorithms based on Darwin’s principles of evolution [12,14]. Over the past two decades, EC have developed from academic curiosities into practical and effective tools for scientists and engineers. Gero, for example, investigates evolutionary systems as computational models of creative design and studies the relationships among genetic engineering, style emergence, and complex evolution [11]. Goodman and co-workers [5] studied evolution of engineering artifacts using heterogeneous parallel genetic algorithms. Koza has applied genetic programming to evolve analog filter circuits and can optimize the topology and sizing parameters of the evolved circuits simultaneously [20]. Other most recent applications of EC in engineering design include [41,42].

3. Genetic programming for system-level sythesis

In this research, genetic programming (GP) is used as a strong search tool to explore the topologically open-ended design space for system-level behavioral models of MEMS. The bond graphs (BG) is also used as a modeling tool to unify representations of mixed energy domains of MEMS. The overall approach is called GPBG approach. The next section gives a brief introduction to bond graph.

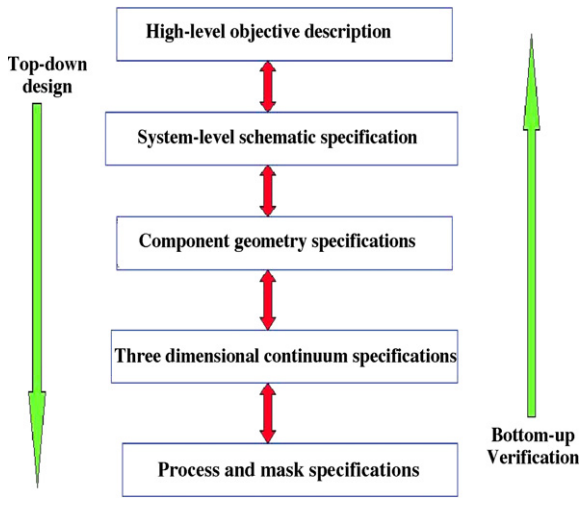

Fig. 1. Hierarchical design process of MEMS.

3.1. Bond graph

Bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems. The performance of a dynamic system that is composed of multi-domain elements is governed by energy conservation laws, which require that power-in equals power-out, also known as the power-balance equation. Power is the product of effort and flow variables. Bond graph models can describe the dynamic behavior of physical systems by the connection of idealized lumped elements based on the principle of conservation of power. These models provide very useful insights into the structures of dynamic systems [15,26,27,28]. Much recent research has explored bond graphs as tools for design [25,32,33,39]. The constitutive equations of the bond graph elements are readily introduced via examples from e.g. the electrical and mechanical domains. The nature of the constitutive equations imposes demands on the causality of the connected bonds. Bond graph elements are drawn as letter combinations (mnemonic codes) indicating the type of element. The bond graph elements are the following [2]:

1. *C*, storage element for a q-type variable, e.g. capacitor (stores charge), spring (stores displacement).
2. *I*, storage element for a p-type variable, e.g. inductor (stores flux linkage), mass (stores momentum).
3. *R*, resistor dissipating free energy, e.g. electric resistor, mechanical friction.
4. S_e and S_f, sources, e.g. battery (voltage source), gravity (force source), pump (flow source).
5. TF, transformer, e.g. an electric transformer, toothed wheels, lever.
6. GY, gyrator, e.g. electromotor, centrifugal pump.
7. 0, 1, 0- and 1-junctions, for ideal connection of two or more sub-models.

Because MEMS are intrinsically multi-domain systems, we need a uniform representation of MEMS so that designers can not only shift among different levels of design abstractions but also move around design partitions in different physical domains without difficulty. 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. It is the explicit representation of model topology that makes the bond graphs a good candidate for use in open-ended design search. It is natural to use bond graphs to represent a dynamic system, such as a mechatronic system, with cross-disciplinary physical domains and even controller subsystems (Fig. 2) [15]. For more notation details and methods of system analysis related to the bond graph representation, see Ref. [27]. Shah and co-workers [34] identifies the importance of bond graphs for unifying multi-level design of multi-domain systems. Tay et al. [33] use bond graphs and GA to generate and analyze dynamic system designs automatically. This approach adopts a variational design method, which means they make a complete bond graph model first, and then change the bond graph topologically using a GA, yielding new design alternatives. However, the efficiency of this approach is hampered by the weak ability of GA to search in both topology and parameter spaces simultaneously. Goodman and co-workers explored the combination of bond graphs and evolutionary computation [6,7]. Terpenny and Wang conducted closely related research [36,35]. Campbell et al. [3] also used the ideas of both bond graphs and genetic algorithms in his A-Design framework. In this research, an approach combining genetic programming and bond graphs is used to automate the process of MEMS design.

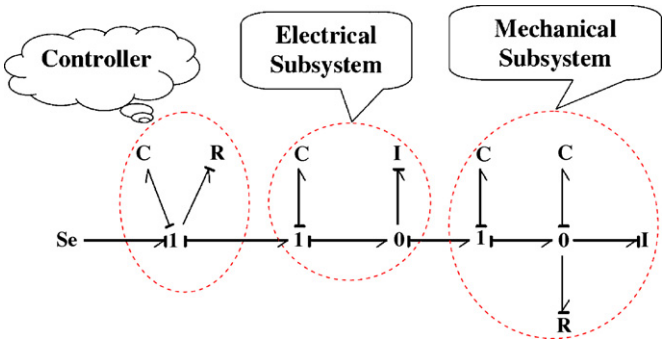

Fig. 2. Bond graph representing a mechatronic system with mixed energy domains and a controller subsystem.

3.2. Combining bond graphs and genetic programming

3.2.1. Genetic programming

Genetic programming is a special type of evolutionary computation, typically involving a graph-type (or other variable-length) representation. The most common form of genetic programming is due to Koza et al. [18–20], which uses trees to represent the entities to be evolved. Genetic programming can be used to "grow" trees that specify increasingly complex bond graph models. If the scope and analysis efficiency of bond graph model can be integrated with the strong search capability of genetic programming when utilized to its full potential, it can result in an extremely capable automated synthesis procedure. Defining a good function set is one of the most significant steps in using genetic programming. It may affect both the search efficiency and validity of evolved results and is closely related to the selection of building blocks for the system being designed. By executing the genotype – a genetic programming tree composed of functions in the function set as nodes of the tree – an arbitrary representative topology, or phenotype, can be generated in a developmental manner. This form of genetic programming, which begins with an embryo at the root of the tree and adds operations to the tree to direct the development of candidate designs from that embryo, is called developmental genetic programming. In this research, we have an additional dimension of flexibility in generating phenotypes, because bond graphs are used as modeling representations for multi-domain systems, serving as an intermediate representation between the mapping of genotype and phenotype, and can be interpreted as systems in different physical domains, chosen as appropriate to given

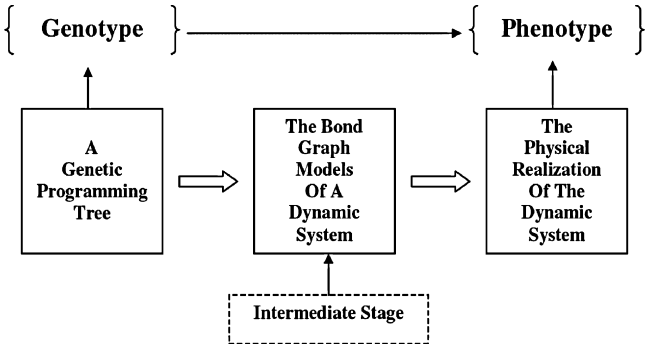

Fig. 3. Genotype–phenotype mapping.

circumstances. Fig. 3 illustrates the role of bond graphs in the mappings from genotypes to phenotypes [6]. And Fig. 4 illustrates a genetic programming tree. Fig. 5 gives a particular example in the domain of electrical circuits [6].

3.3. Filter topology

Automated synthesis of an RF MEM device, a micro-mechanical bandpass filter [37], is used as an example to illustrate structured synthesis procedure using evolutionary approaches. Through the investigation of two popular topologies used in surface micromachining of micro-mechanical filters, we found that they are topologically composed of a series of concatenated resonator units (RUs) and bridging units (BUs) or RUs and coupling units (CUs). Fig. 6 illustrates the layout of one widely accepted filter topology [37], where its corresponding bond graph representation is also shown.

3.4. Function set

In this research, a GP function set is presented and listed in Table 1. Examples of operators, namely insert_CU and insert_RU, are illustrated in Figs. 7 and 8. Fig. 7 explains how the insert_CU function works. A coupling unit (CU) is a subsystem that is composed of a capacitor attached to a 0-junction in the center and two bonds connecting 1-junctions at the left and right ends. After execution of the insert_CU function, an additional modifiable site (2) appears at the rightmost newly created bond. As illustrated in Fig. 8, a resonator unit (RU), composed of one I, R, and C component all attached to a 1-junction, is inserted in an original bond with a modifiable site through the insert_RU function. After the insert_RU function is executed, a new RU is created and one additional modifiable site, namely bond (3), appears in the resulting phenotype bond graph, along with the original modifiable site bond (1). The newly added 1-junction also has an additional modifiable site (2). As components C, I, and R all have parameters to be evolved, the insert_RU function has three corresponding numeric sites (4–6), for numerical evolution of parameters.

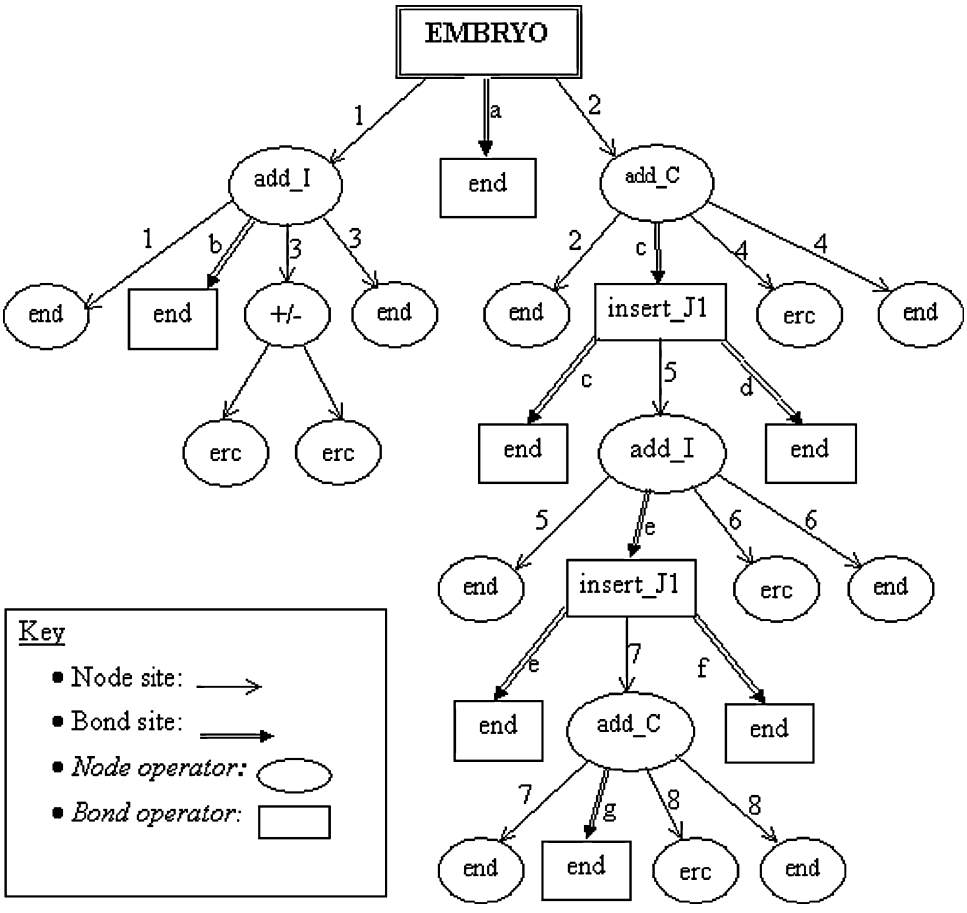

Fig. 4. Genetic programming tree.

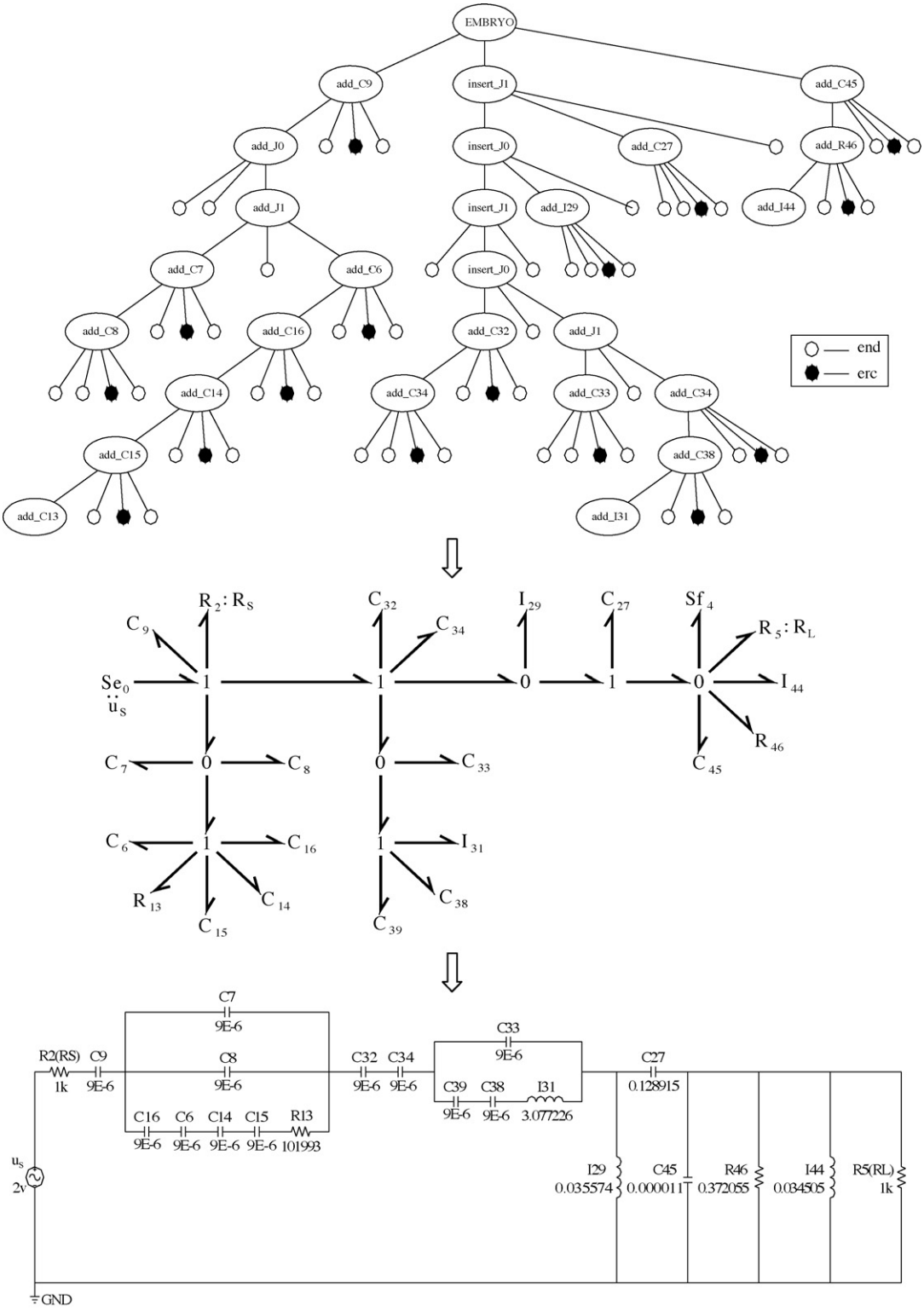

Fig. 5. Example of genotype–phenotype mapping in the electrical circuit domain.

3.5. Design embryo

All individual genetic programming trees create bond graphs from an embryo. To search for a new design using the GPBG design tool, an embryo model, represented by bond graph, is required. The embryo model is the fixed part of the system and the starting point for GP to generate candidates of system designs by adding new components in a developmental manner. 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 bond graph shown in Fig. 9 as an embryo.

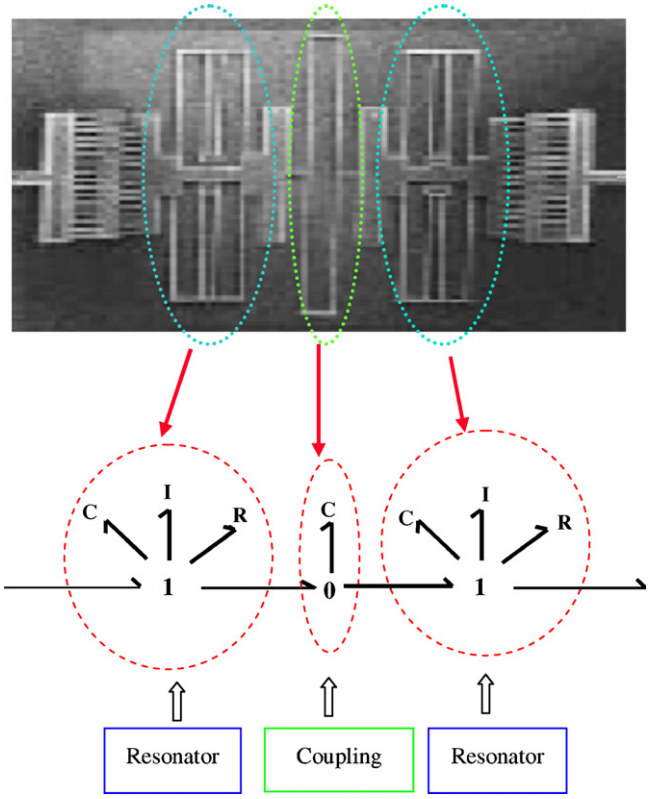

Fig. 6. MEM filter topology.

3.6. Fitness function

Within the frequency range of interest, $f_{range} = [f_{min}, f_{max}]$, sample 100 points logarithmically spaced. In this paper the $f_{range} = [0.1,\ 100\ K]$ Hz. Compare the magnitudes of the frequency response at target magnitudes, which are 1.0 within the band pass frequency range of [316, 1000] Hz, and 0.0 otherwise, between 0.1 and 100 KHz.

3.7. Experimental setup

Three major code modules were developed in this work. The algorithm kernel of HFC-GP was a strongly typed version [22] of an open software package developed in our research group, lilgp. Parameters for lilgp are shown in Table 2.

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 [38] was used to verify the dynamic characteristics of the evolved design.

Table 1
Operators in modular function set

Modular function set	
insert_RU	Insert a resonator unit
insert_CU	Insert a coupling unit
insert_BU	Insert a bridging unit
add_RU	Add a resonator unit
insert_J01	Insert a 0–1 junction
insert_CIR	Insert a special CIR
insert_CR	Insert a special CR
add_J	Add a junction compound

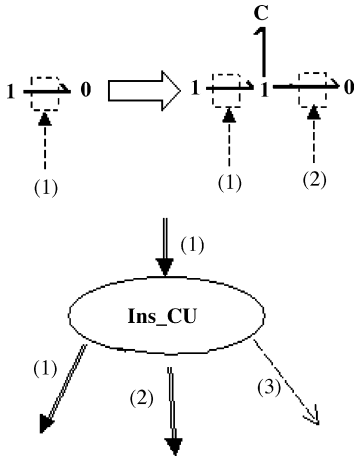

Fig. 7. Operator to insert bridging unit.

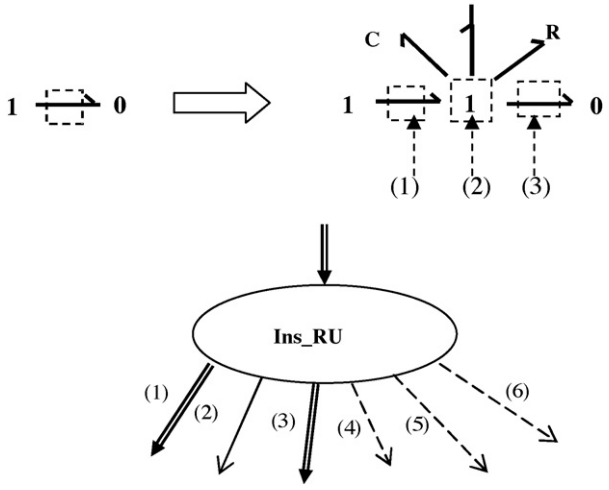

Fig. 8. Operator to insert resonator unit.

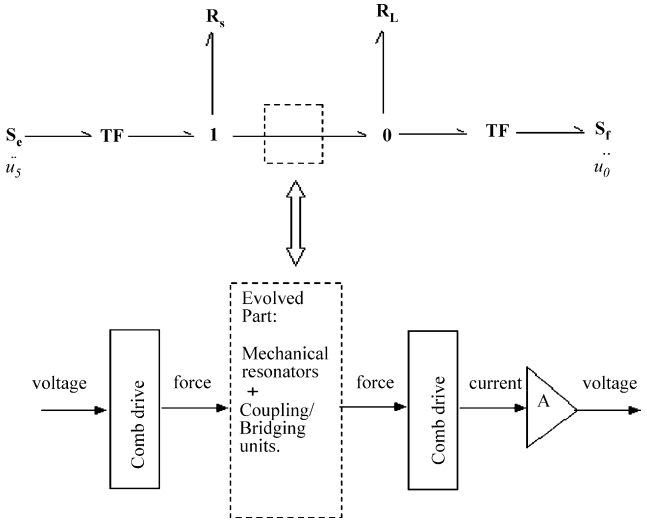

Fig. 9. Design embryo of a MEM filter.

Table 2
Evolution parameters

Population size	500 in each of 13 subpopulations
Initial population	half_and_half
Initial depth	4–6
Maximum depth	50
Maximum nodes	5000
Selection	Tournament (size = 7)
Crossover	0.9
Mutation	0.3

3.8. Experimental results

Experimental results show the strong topological search capability of genetic programming and feasibility of our GPBG approach for finding realizable system-level designs for micromechanical filters [8].

In Figs. 10 and 11, K is the number of resonator units appearing in the best design of the generation on the horizontal axis. As fitness improves, the number of resonator units (K) grows; this is because a higher-order system with more resonator units has the potential of better system performance than its low-order counterpart. The plots of corresponding system frequency responses at generations 27, 52,117 and 183, where a fitness leap happens, are shown in Fig. 10. The fitness improvement curve is illustrated in Fig. 11.

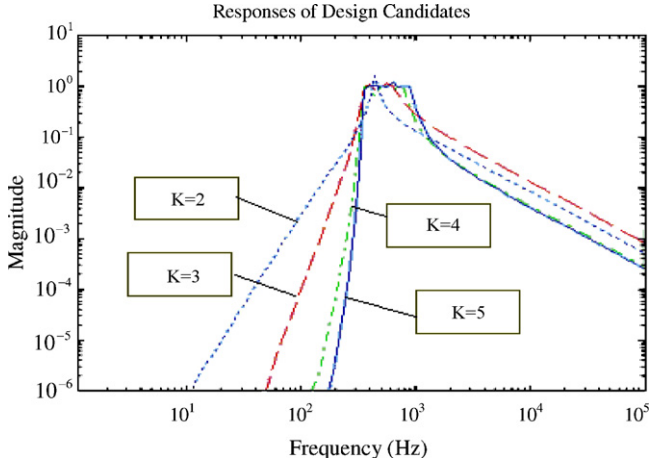

Fig. 10. Frequency responses of a sampling of design candidates, which evolved topologies with larger numbers, K, of resonators as the evolution progressed. All results are from one genetic programming run of the GPBG approach.

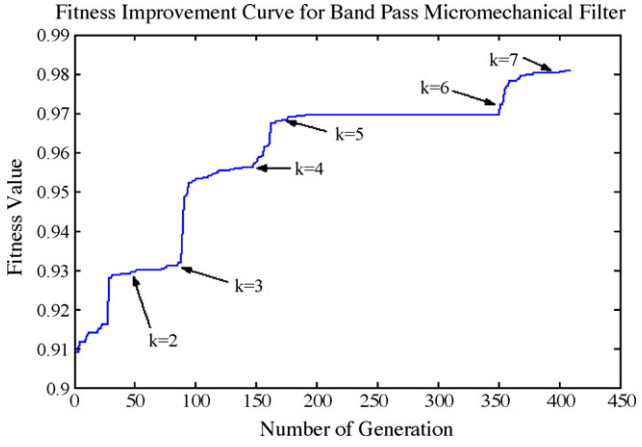

Fig. 11. Fitness improvement curve.

A layout of a design candidate with four resonators and three coupling units and its bond graph representation is shown in Fig. 12. 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.

Using the GPBG approach, it is also possible to explore novel topologies for MEM filter design. In this case, we may choose not to use a strictly realizable function set. Instead, a semi-realizable function set may be used to relax the topological constraints, with the purpose of finding new topologies not realized before but still realizable after careful design. Fig. 13 gives an example of a novel topology for a MEM filter design. An attempt to fabricate this kind of topology is being carried out in a university research setting.

4. Second level physical layout synthesis

Layout synthesis automatically generates valid or optimized geometric sizing parameters for cell components, which in most cases are chosen from micromechanical devices with fixed topologies, according to engineering design objectives. In this research, the cell component is a resonator device in the MEMS domain. The design objectives come from either high-level specifications, such as behavioral model parameters that must be satisfied, or from layout-level objectives such as minimizing the area occupied. Our approach is to model the design problem as a formal constrained optimization problem, and then solve it with powerful optimization techniques, resulting in a tool that automates the layout synthesis of MEMS structures. Two categories of optimization techniques are used: one category includes stochastic algorithms such as genetic algorithms, and the other category includes deterministic algorithms such as nonlinear programming. For both categories, the process of solving the optimization problem involves determining the design variables, the design constraints, and the design objectives.

There are 14 design variables for the cell component example in this research, i.e., a folded-flexure comb-driven microresonator [9] fabricated in a polysilicon surface microstructural process (Fig. 14). Design variables and their constraints are listed as follows (Fig. 15):

$$2 \le L_b \le 400, \quad 2 \le w_b \le 20, \quad 2 \le L_t \le 400, \quad 2 \le w_t \le 20, \quad 2 \le L_{sy} \le 400, \quad 10 \le w_{sy} \le 400, \quad 10 \le w_{sa} \le 400, \quad 10 \le w_{cy} \le 400, \quad 2 \le L_{cy} \le 700, \quad 8 \le L_c \le 400, \quad 2 \le w_c \le 20, \quad 2 \le L_{sa} \le 400, \quad 4 \le x_0 \le 400, \quad 0 \le V \le 100.$$

It is noted that the first 13 design variables have units of μm. The fourteenth design variable (V) has units of volts.

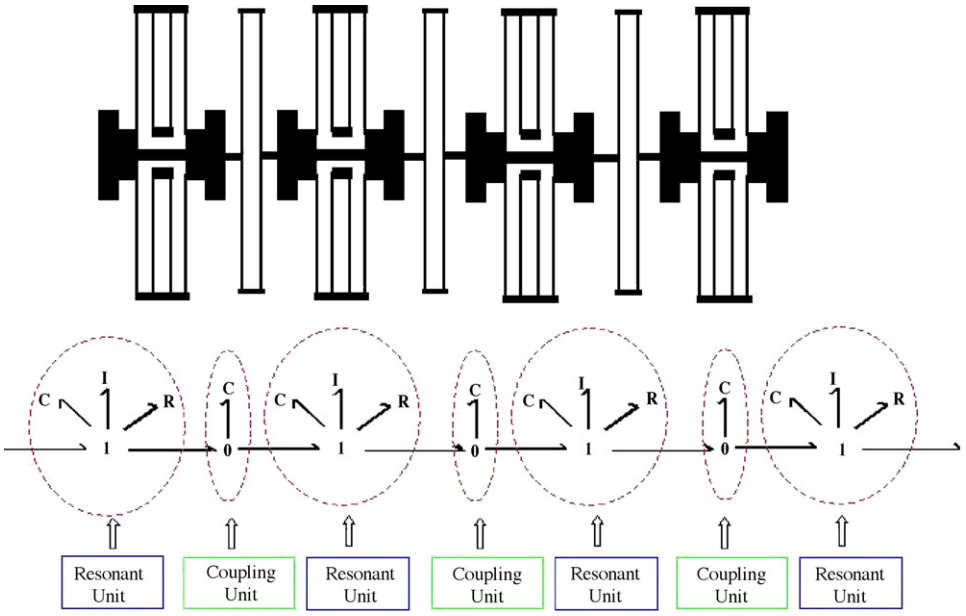

Fig. 12. Layout and bond graph representation of a design candidate from the experiment, with four resonator units coupled with three coupling units.

In addition, it is assumed that: $t = w_c = g = d$ in our design for simplicity. Some design variables are predefined as follows: $w_{ba} = 11$, $w_{ca} = 14$, $\delta = 4$ and $N = 10$.

There are also a number of design constraints for the microresonator cell component, including both geometric constraints and functional constraints. In this paper, without loss of generality, we consider the following constraints:

$$0 \le L_{cy} + 2g + 2w_c \le 700$$
$$0 \le L_{sy} + 2L_b + 2w_t \le 700$$
$$0 \le 3L_t + w_{sy} + 4L_c - 2x_0 + 2w_{cy} + 2w_{ca} \le 700$$
$$(2N+1)W_c + 2Ng \le L_{cy}$$
$$4 \le L_c - (x_0 + x_{\text{disp}}) \le 200$$
$$4 \le x_0 - x_{\text{disp}} \le 200$$

Among them, the first three are linear constraints, and the fourth is a nonlinear constraint because the term x_{disp} is highly nonlinear.

$$x_{\text{disp}} = \frac{QF_{e,x}}{K_x} \tag{1}$$

where:

$$F_{e,x} = 1.12\varepsilon_0 N \frac{t}{g} V^2 \tag{2}$$

Suppose that in the system-level synthesis, we get a set of behavioral parameters for the cell component of a microreso-

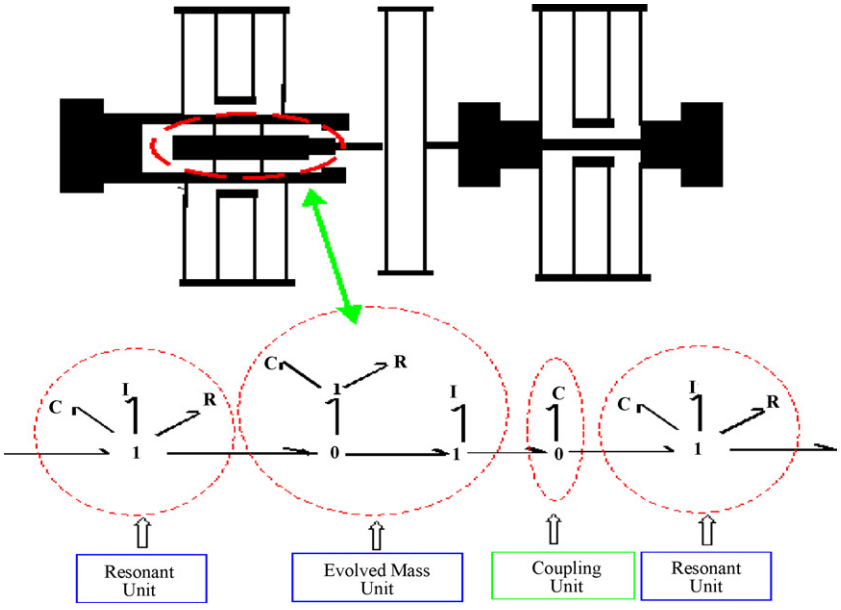

Fig. 13. A novel topology of MEM filter and its bond graph representation.

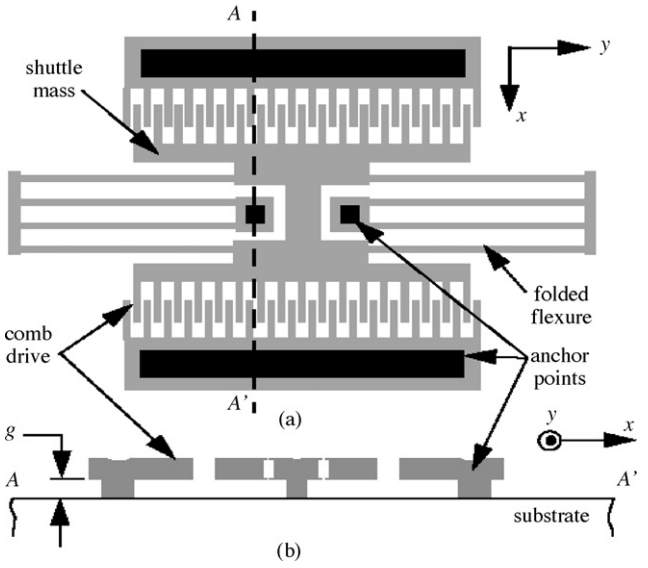

Fig. 14. A folded-flexure comb-drive microresonator fabricated in a polysilicon surface microstructural process (a) layout (b) cross-section A–A'.

nator as

$$\begin{cases} K_x = 1.45\,\mathrm{N/m} \\ B_x = 4.62\mathrm{e}-6\,\mathrm{kg\,m^2} \\ m_x = 4.10\mathrm{e}-11\,\mathrm{kg} \end{cases}$$

Then we have three additional equation constraints. Equations to relate the design variables and the three behavioral model parameters are as follows:

$$K_x = \frac{2EtW_b^3}{L_b^3}\,\frac{L_t^2 + 14\alpha L_t L_b + 36\alpha^2 L_b^2}{4L_t^2 + 41\alpha L_t L_b + 36\alpha^2 L_b^2},$$

$$\text{where } \alpha = \left(\frac{W_t}{W_b}\right)^3 \tag{3}$$

$$B_x = \mu\left[(A_s + 0.5A_t + 0.5A_b)\left(\frac{1}{d} + \frac{1}{\delta}\right) + \frac{A_c}{g}\right] \tag{4}$$

$$m_x = m_s + \frac{1}{4}m_t + \frac{12}{35}m_b \tag{5}$$

$$\text{where} \quad m_s = \rho A_s, \quad m_t = \rho A_t, \quad m_b = \rho A_b \tag{6}$$

$$A_s = w_{sa}L_{sa} + 2w_{sy}L_{sy} \tag{7}$$

$$A_t = 2w_{ca}L_{cy} \tag{8}$$

$$A_b = 8L_b w_b + 2w_t(2L_t + w_a + 2w_b) \tag{9}$$

As an alternative, we can also put reformulations of these three constraint equations into the design objectives, expressing them as differences to be minimized. In that case, we actually deal with a multi-objective constrained optimization problem. We have formulated the objective function in the following format:

$$f(\vec{x}) = \frac{1}{1.45}|K_x - 1.45| + \frac{1}{4.62\mathrm{e}^{-6}}|B_x - 4.62\mathrm{e}^{-6}| + \frac{1}{4.10\mathrm{e}^{-11}}|M_x - 4.10\mathrm{e}^{-11}| \tag{10}$$

The objective function is composed of three terms. Each term corresponds to the normalized deviation of a solution parameter, namely B_x, M_x or K_x, respectively, from its targeted value. As for $\vec{x}$, it represents the vector of all the design variables. We seek solutions such that each of the three terms is less than 1% of the target value and the sum of the terms is less than 2%, in order to satisfy the design constraints in this research.

Finally, it is important to note the role of feature size in VLSI and MEMS design. Feature size, which is often represented as λ, means the minimum size a particular design can achieve, based on specific fabrication procedures. In addition, the actual sizes of geometric shapes should be integer multiples of the

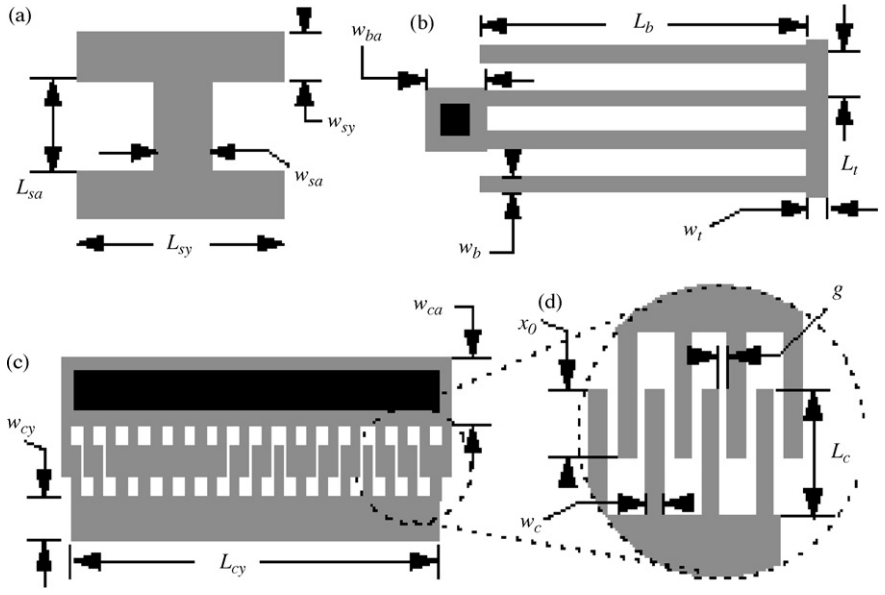

Fig. 15. Major design variables for microresonators.

feature size λ, such as λ, 2λ, 5λ, 10λ, . . ., etc. In this research, we set $\lambda = 0.09$ μm. While it is very difficult for many numerical optimization approaches (for example, gradient-based approaches) to include considerations of feature size constraints [9], it is quite convenient for genetic algorithms to do so. We need to modify the objective function only slightly, mapping real values of design variables to integer multiples of the feature size λ before using them in formulations of constraints and objectives. No modifications to the genetic algorithm are needed. The genetic algorithm was selected as the optimization tool for layout optimization also because it can deal with fixed variables (with both discrete variables and continuous variables, e.g. input voltage in this case study) with no difficulty.

4.1. Solving the optimization problem using HEEDS

This search was implemented using a genetic algorithm module within the commercial software code Hierarchical Evolutionary Engineering Design System (HEEDS) (©Red Cedar Technology) [13]. The choice of this software was motivated by several of its advantages regarding optimization and solutions search. Within HEEDS, a hybrid and adaptive search strategy is employed as the default search method. During a single search, multiple search methods are used simultaneously (as opposed to sequentially). This approach takes advantage of the best attributes of each method, while overcoming the known weaknesses associated with each method by reducing their participation in the search if/when they are found to be ineffective. A combination of global and local search methods is used, with the number of different methods used at any time ranging between two and ten. Each method contains tuning parameters that are modified automatically during the search according to knowledge gained about the nature of the design space. Furthermore, HEEDS makes all the decisions about which method to use, hence the designer does not have to worry about this dimension of the problem.

HEEDS minimizes the function presented in Eq. (10), which also represents the fitness function of the constrained GA.

The parameters for setting the constrained GA search are presented in Table 3.

In five runs of the genetic algorithm using different random seeds, we obtained the sizing parameters and values of the objective function (to be minimized) listed in Table 4. It can be seen that during the five GA runs using different seeds, the GA performs very steadily. Almost all runs achieved the performance objectives, i.e., the sum of normalized deviation of targeted behavioral parameters is less than 2%, and all of the three normalized deviations of targeted behavioral parameters were less than 1%. It also appears that there are many alternative and rather different ways in which parameters can be set and still produce behavior very close to that desired.

Table 3
Evolution parameters for the constrained GA

Total no of generations: 550	Population size: 250
Crossover type: one point	Crossover probability: 0.50
Mutation type: multi-field	Mutation probability (real): 0.20
Selection type: tournament	Selection parameter: 2

Table 4
Layout parameters obtained in five GA runs, with different random seeds

Run no.	1	2	3	4	5
L_b (μm)	167.85	186.39	163.80	196.38	193.95
w_b (μm)	2.07	2.52	2.07	2.43	2.34
L_t (μm)	14.94	32.85	13.68	11.07	48.15
w_t (μm)	4.59	2.97	8.37	2.25	3.69
L_{sy} (μm)	254.79	155.70	164.25	133.02	187.56
w_{sy} (μm)	38.61	34.11	32.76	97.38	70.83
w_{sa} (μm)	94.86	45.81	98.10	16.29	65.34
w_{cy} (μm)	166.86	209.34	263.07	81.90	170.28
L_{cy} (μm)	382.23	452.25	448.29	315.09	428.49
L_c (μm)	166.68	163.44	169.47	103.23	148.77
w_c (μm)	2.43	2.00	2.25	2.52	2.70
L_{sa} (μm)	67.41	216.63	129.60	210.33	16.92
x_0 (μm)	88.56	59.58	34.47	11.97	29.25
V (V)	4.05	22.60	23.50	9.00	4.80
Deviation of B_x (%)	0.024	0.067	0.23	0.006	0.27
Deviation of M_x (%)	0.00	0.00	0.00	0.00	0.00
Deviation of K_x (%)	0.025	0.14	0.12	0.138	0.02
Objective value (%)	0.05	0.21	0.35	0.14	0.29

5. Conclusion

This paper has suggested a design methodology for automatically synthesizing hierarchical designs for MEMS. While there has been much research using evolutionary computation techniques to synthesize MEMS [23,40], this is the first work reported that attempts to automates the hierarchical MEMS synthesis process in an integrated framework. Our first step is to synthesize system-level behavioral models using a combination of genetic programming and bond graphs. Then in the second step, we use a constrained genetic algorithm to automatically optimize the geometric sizing parameters for the cell components. An example of MEM filter design with coupling of multiple microresonators is used to illustrate the approach. Extension of this work can lead to a composable design and synthesis environment for micromechatronic systems [24]. In addition, target cascading in optimal system design needs to be investigated in depth to propagate the desirable top-level design specifications to appropriate specifications for the various subsystems and components in a consistent and efficient manner [16,17]. More work is underway to improve the efficiency of genetic programming to explore topologically open-ended design spaces, and the robustness of the constrained genetic algorithm to solve real-world constrained optimization problems.

Acknowledgements

The authors gratefully acknowledge the support of the National Science Foundation (NSF) through grant DMI 0084934. The authors are grateful to Ranny Sidhu and Johanna Burgueno at Red Cedar Technology, Inc., for their help in

setting up and using the HEEDS software. Financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) to enable Sofiane Achiche to work at Technical University of Denmark under Post-Doctoral Grant BP-328508-2006 is also gratefully acknowledged.

References

[1] E.K. Antonsson, Microsystem design synthesis, in: E.K. Antonsson, J. Cagan (Eds.), Formal Engineering Design Synthesis, Cambridge University Press, 2001, pp. 126–169.
[2] F. Broenink, Introduction to Physical Systems Modeling with Bond Graphs, SiE Whitebook on Simulation Methodologies (1999) http://www.rt.el.utwente.nl/bnk/papers/BondGraphsV2.pdf.
[3] M. Campbell, J. Cagan, K. Kotovsky, Agent-based synthesis of electromechanical design configurations, J. Mech. Des. 122 (2000) 61–69.
[4] S. Cho, K. Lee, T. Kim, Development of a geometry-based process planning for surface micromachining, International Journal of Production Research 40 (2002) 1275–1293.
[5] D. Eby, R. Averill, B. Gelfand, W. Punch, O. Matthews, E. Goodman, An injection island GA for flywheel design optimization, in: 5th European Congress on Intelligent Techniques and Soft Computing, EUFIT'97, vol. 1, 1997, 687–691.
[6] Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, B. Zhang, Bond Graph Representation and GP for automated analog filter design, GECCO-2001 Late-Breaking Papers, 2001, pp. 81–86.
[7] Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, A novel evolutionary engineering design approach for mixed-domain systems, Engineering Optimization 36 (2004) 127–147.
[8] Z. Fan, K. Seo, R. Rosenberg, J. Hu, E. Goodman, System-level synthesis of mems via genetic programming and bond graphs, in: Proceedings of GECCO'03, Lecture Notes in Computer Science, 2003, pp. 2058–2071.
[9] G. Fedder, T. Mukherjee, Physical design for surface-micromachined MEMS, in: Proceedings of the Fifth ACM/SIGDA Physical Design Workshop, 1996, pp. 53–60.
[10] G. Fedder, Q. Jing, A hierarchical circuit-level design methodology for microelectromechanical systems, IEEE Transactions on Circuits and Systems II (TCAS) 46 (1999) 1309–1315.
[11] J.S. Gero, Computers and creative design, in: M. Tan, R. The (Eds.), The Global Design Studio, National University of Singapore, 1996, pp. 11–19.
[12] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[13] HEEDS (Hierarchical Evolutionary Engineering Design System), Getting Started Manual, Red Cedar Technology, East Lansing, MI, 48823, USA, 2002, http://www.redcedartech.com.
[14] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[15] D.C. Karnopp, D.L. Margolis, R.C. Rosenberg, System Dynamics A Unified Approach, third ed., John Wiley & Sons, 2000.
[16] H.M. Kim, N.F. Michelena, P.Y. Papalambros, T. Jiang, Target cascading in optimal system design, in: Proceedings of the ASME Design Automation Conference, 2000, DAC-14265.
[17] H.M. Kim, Target cascading in optimal system design, Ph.D. Dissertation, Department of Mechanical Engineering, University of Michigan, 2001.
[18] J.R. Koza, F.H. Bennet, D. Andre, M.A. Keane, Genetic Programming III Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers, 1999.
[19] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[20] J.R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, 1994.
[21] H. Li, E.K. Antonsson, Evolutionary Techniques in MEMS Synthesis. In ASME International Mechanical Engineering Congress and Exposition (Anaheim, CA, Nov. 1998), ASME.
[22] S. Luke, Strongly Typed, Multithreaded C Genetic Programming Kernel, 1997, http://www.cs.umd.edu/users/-seanl/gp/patched-gp/.
[23] L. Ma, E.K. Antonsson, Automated mask-layout and process synthesis for MEMS, in: Technical Proceedings of the International Conference on Modeling and Simulation of Microsystems, 2000, pp. 20–23.
[24] C.J.J. Paredis, A. Diaz-Calderon, R. Sinha, P.K. Khosla, Composable models for simulation-based design, Engineering with Computers 17 (2001) 112–128.
[25] R.C. Redfield, Bond graphs in dynamic systems designs: concepts for a continuously variable transmission, in: International Conference on Bond Graph Modeling and Simulation, 1999, 225–230.
[26] R.C. Rosenberg, Y.-Y. Wang, Dynamic multiport subsystems, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 1, 1993, pp. 26–29.
[27] R.C. Rosenberg, Reflections on engineering systems and bond graphs, Journal of Dynamic Systems, Measurements and Control 115 (1993) 242–251.
[28] R.C. Rosenberg, M.K. Hales, M. Minor, Engineering icons for multidisciplinary systems, in: Proceedings ASMEIMECE, vol. 58, 1996, pp. 665–672.
[29] R.L. Schiek, R.C. Schmidt, Automated surface micro-machining mask creation from a 3D model, Microsystem Technologies 12 (2006) 204–207.
[30] R.L. Schiek, R.C. Schmidt, Automated and integrated mask generation from a CAD constructed 3D model, in: Nanotechnology Conference and Trade Show, NSTI, Nanotech Technical Proceedings, 2005, pp. 443–446.
[31] S.D. Senturia, Microsystem Design, Kluwer Academic Publishers, 2001.
[32] J.E.E. Sharpe, R.H. Bracewell, The use of bond graph reasoning for the design of interdisciplinary schemes, in: International Conference on Bond Graph Modeling and Simulation, 1995, 116–121.
[33] E. Tay, W. Flowers, J. Barrus, Automated generation and analysis of dynamic system designs, Research in Engineering Design 10 (1998) 15–29.
[34] N. Vargas-Hernandez, J. Shah, Z. Lacroix, Development of a computer-aided conceptual design tool for complex electromechanical systems, Computational Synthesis: From Basic Building Blocks to High Level Functionality, Papers from the 2003 AAAI Symposium Technical Report SS-03-02, 2003, pp. 255–261.
[35] J. Wang, Z. Fan, J.P. Terpenny, E.D. Goodman, Knowledge interaction with genetic programming in mechatronic systems design using bond graphs, IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews 35 (2005) 172–182.
[36] J. Wang, J.P. Terpenny, Integrated active and passive mechatronic system design using bond graphs and genetic programming, GECCO'03 Late Breaking Paper, Chicago, Illinois, July 12–16, 2003, pp. 322–329.
[37] K. Wang, C.T.C. Nguyen, High-order medium frequency micromechanical electronic filters, Journal of Microelectromechanical Systems 8 (1999) 534–556.
[38] http://www.20sim.com.
[39] K. Youcef-Toumi, Y. Ye, A. Glaviano, P. Andrson, Automated zero dynamics derivation from bond graph models, in: International Conference on Bond Graph Modeling and Simulation, 1999, 39–44.
[40] N. Zhou, B. Zhu, A.M. Agogino, K.S.J. Pister, Evolutionary synthesis of MEMS design, in: Proceedings of ANNIE 2001, Intelligent Engineering Systems through Artificial Neural Networks, vol. 11, ASME Press, 2001, pp. 197–202.
[41] V. Oduguwa, R. Roy, D. Farrugia, Development of a soft computing-based framework for engineering design optimisation with quantitative and qualitative search spaces, Applied Soft Computing 7 (2007) 166–188.
[42] H. Liu, M. Tang, Evolutionary design in a multi-agent design environment, Applied Soft Computing 6 (2006) 207–220.