This is to certify that the
dissertation entitled

DESIGN AUTOMATION OF MECHATRONIC SYSTEMS USING EVOLUTIONARY COMPUTATION AND BOND GRAPH

presented by

ZHUN FAN

has been accepted towards fulfillment
of the requirements for the

PhD	degree in	Department of Electrical and Computer Engineering

Erik D Goodman
Major Professor's Signature

May 12, 2004
Date

MSU is an Affirmative Action/Equal Opportunity Institution

DESIGN AUTOMATION OF MECHATRONIC SYSTEMS USING EVOLUTIONARY COMPUTATION AND BOND GRAPH

By

ZHUN FAN

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

2004

ABSTRACT

DESIGN AUTOMATION OF MECHATRONIC SYSTEMS USING EVOLUTIONARY COMPUTATION AND BOND GRAPH

By

ZHUN FAN

The research of this dissertation is of significance because it is one of the first endeavors to address the challenging issue of design automation of mechatronic systems, at a time when mechatronics is emerging as an integrated and independent discipline of the 21st century. Just as Electronic Design Automation (EDA) has changed the face of design of electronic systems, Mechatronic Design Automation (MDA) is gaining more and more importance in addressing the ever-growing, competing challenges of the current market. In fact, design automation and optimization have become mainstream disciplines in the area of engineering design.

The motivation of this research is two-fold. First, we want to find a way to generate a population of topologically open-ended design alternatives and provide for the designer, in an automated manner, a variety of satisfactory design candidates to choose among and trade off. Second, we want our method to be applicable not only in one physical domain, but in multiple domains or a mixture of them, as is required for design of mechatronic systems. To meet these ends, the capability of genetic programming, a special type of evolutionary computation techniques, to search automatically in an open-ended search

space and the strong capability of bond graphs to represent and model mixed-domain systems are studied and ways to blend their merits in one unified approach are investigated. In this research, the BG/GP method, combining bond graphs and genetic programming, has been developed to automate the conceptual design process for general multidisciplinary mechatronic systems.

Several design problems, in macro- and micro-domains, and in different physical domains, have been used as design examples to test the feasibility of the BG/GP approach. The analog electronic filter design problem shows the efficiency and effectiveness of the proposed approach. A vibration absorber design for a mechanical printer demonstrates that the approach can also be used for redesign and is very effective in exploring in an open-ended topology space and capable of providing designers with a variety of good design candidates for further analysis and tradeoff. Finally, a Micro-Electro-Mechanical (MEM) filter design problem shows that the BG/GP approach can be applied in a very general class of conceptual design problems with severe topology and/or parameter constraints. The results show that the BG/GP method is a powerful synergistic approach for automated, mixed-domain, and topologically open-ended conceptual design of mechatronic systems.

A structured and hierarchical design methodology for Micro-Electro-Mechanical-Systems (MEMS) is also studied. MEMS are actually micro-mechatronic systems. The research of hierarchical evolutionary synthesis of MEMS in this thesis includes the system-level behavioral synthesis and second-level layout synthesis of MEMS. Preliminary results show that automated synthesis of MEMS is a very promising research area.

For my mother, father, and sister

Their love and supports make my dream come true

ACKNOWLEDGEMENTS

First I would like to thank Professor Erik Goodman, who gave me guidance through out my research. Without his knowledge, patience and support this proposal would have not been possible. He is not only my academic advisor, but also a mentor and good friend. The knowledge and friendship I gained from him will definitely influence the rest of my life. I would also thank Professor Ronald Rosenberg, his keen insights and valuable discussions often gave me stimulating ideas in research. Though kept busy by his duty and work, he is always willing to share his time and knowledge with us. I also owe many thanks to Dr. Kisung Seo, whose diligence and strictness in work gave me very deep impression. His great organization of a group research makes our research a steady progress. I have a lot to learn from him, both as a teacher and as a friend.

I am grateful to Professor Aslam and Professor MacCluer for taking time to serve on the guidance committee and overseeing this work. Their insightful comments and suggestions have enhanced the technical soundness of this proposal. I am also grateful to all my friends and colleagues. The discussions with them substantially contributed to my work and broadened my knowledge.

Last but not the least, many thanks go to my family: my Mom and Dad, and my sister. Without their encouragement and continuous support, I would not be who I am today.

This research is supported by National Science Foundation.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION 1
1.1 Automated Synthesis 2
1.2 Representation of Multidisciplinary Mechatronic Systems 5
1.3 Related Work 7
1.3.1 Bond Graphs 7
1.3.2 GA/GP 8
1.3.3 Automated Design Theory and Practice 9
1.3.4 The BG/GP Approach 12
1.4 Contributions of the Dissertation 16
1.5 Organization 17

CHAPTER II

BOND GRAPHS 18
2.1 Causality of Bond Graphs 20
2.2 Bond Graph Evaluation 21
2.2.1 Causality Analysis 22
2.2.2 Model insight via causal analysis 23
2.2.3 State Equation Formulation 24
2.3 Simplification of Bond Graphs 25
2.4 Strengths of Bond Graphs 30

CHAPTER III

EVOLUTIONARY DESIGN 31
3.1 Evolutionary Design with Bond Graphs 32
3.1.1 Generation of Design Candidates 32
3.1.2 Bond Graph Construction 32
3.1.3 Reconfiguration of Designs 36
3.1.4 Fitness Evaluation 40
3.1.5 Selection 40
3.1.6 Premature Convergence 41
3.2 Overall Design Procedure 42

CHAPTER IV

CASE STUDIES OF BG/GP APPROACH 45
4.1 Analog Filter Design Problem 45
4.1.1 Bond Graphs Representation of Circuits 46
4.1.2 Problem Definition 48
4.1.3 Results 49

4.2 Design of Vibration Absorber for Mechanical Printer 56
4.2.1 Problem Formulation 56
4.2.2 Embryo of Design 58
4.2.3 Results 61
4.3 Discussions 66

CHAPTER V

EVOLUTIONARY SYNTHESIS OF MEMS 68
5.1 Introduction to MEMS Design and Synthesis 69
5.2 Promises and Challenges of MEMS Design and Synthesis 70
5.3 Hierarchical MEMS Design Methodology 73
5.4 System-Level Synthesis of MEMS 74
5.4.1 Bond Graphs 76
5.4.2 Combining Bond Graphs and Genetic Programming 78
5.4.3 Filter Topology 81
5.4.4 Realizable Function Set 84
5.4.5 Design Embryo 87
5.4.6 Adaptive Fitness Function 88
5.4.7 Experimental Setup 89
5.4.8 Experimental Results 90
5.5 Second-Level Physical Layout Synthesis 93
5.5.1 Formulation of Layout Synthesis as an Optimization Problem 94
5.5.2 Solving the Optimization Problem Using GA 97
5.6 Conclusions 99

CHAPTER VI

CONCLUSIONS 102
6.1 Contributions 102
6.2 Future Work 103

APPENDIX A 106
APPENDIX B 109
BIBLIOGRAPHY 113

LIST OF TABLES

Table 1.1 Comparisons between GP and 'classical' GA 4

Table 1.2 Comparisons of various design approaches 15

Table 2.1 Flow and effort variables for each domain 19

Table 3.1 Definition of function set 33

Table 4.1 Summary results (errors, fitnesses) for filter designs 55

Table 4.2 Summary results of fitness for printer 65

Table 5.1 Operators in basic function set 85

Table 5.2 Operators in modular function set 85

Table 5.3 Layout parameters obtained in ten GA runs (different random seeds) 98

LIST OF FIGURES

Figure 1.1 Requirements for Automated Design of Mixed-Domain Systems 1

Figure 1.2 Bond Graphs Representation of Mixed-Domain Systems 6

Figure 1.3 The combinatorial nature of bond graphs generation 7

Figure 1.4 General flow chart of the BG/GP design 14

Figure 2.1 Example of causality assignment 20

Figure 2.2 Evaluation flow of bond graphs models 21

Figure 2.3 Example of causality assignment 23

Figure 2.4 Elimination of redundant junctions in bond graphs 25

Figure 2.5 Merging of junctions in bond graphs 26

Figure 2.6 Merging of elements in bond graphs 27

Figure 2.7 An example of bond graphs simplification 29

Figure 3.1 Illustration of add_R operator 35

Figure 3.2 Illustration of insert_J0 operator 35

Figure 3.3 An example of a GP tree 37

Figure 3.4 The bond graphs model generated by the GP tree of Figure 3.3 38

Figure 3.5 Illustration of crossover operator and mutation operator 38

Figure 3.6 The extensible search capability of GP for an unbounded design space 39

Figure 3.7 The overall design procedure of BG/GP approach 44

Figure 4.1 Bond graph representation of an electrical circuits 47

Figure 4.2 Embryo of electrical circuit and its bond graphs model 49

Figure 4.3 Frequency response of a high-pass filter design with fitness value of 0.917 .. 51

Figure 4.4 Frequency response of a high-pass filter design with fitness value of 0.992.. 51

Figure 4.5 Frequency response of a low-pass filter design with fitness value of 0.980... 52

Figure 4.6 Frequency response of a band-pass filter design with fitness value of 0.884 . 52

Figure 4.7 Evolved bond graphs model for high-pass filter 53

Figure 4.8 Evolved electrical circuit for high-pass filter design..... 54

Figure 4.9 Fitness history for a typical high-pass filter run..... 55

Figure 4.10 The schematic of the original printer system 56

Figure 4.11 Bond graphs model for the original printer system 57

Figure 4.12 Simulation result of the original printer drive subsystem. 57

Figure 4.13 The critical printer drive subsystem 58

Figure 4.14 The design embryo of printer subsystem..... 59

Figure 4.15 Fitness history for a typical printer drive redesign run..... 62

Figure 4.16 The evolved bond graph model I..... 62

Figure 4.17 The physical realization of evolved bond graph model I 63

Figure 4.18 Simulation result of evolved bond graph model I 63

Figure 4.19 The evolved bond graph model II 64

Figure 4.20 The physical realization of evolved bond graph model II..... 64

Figure 4.21 Simulation result of evolved bond graph model II..... 65

Figure 5.1 Examples of MEMS 70

Figure 5.2 Hierarchical Design of MEMS..... 73

Figure 5.3 Structured MEMS Design Flow 75

Figure 5.4 A Single Bond Graph Represents a Resonator Unit in Three Domains.....77

Figure 5.5 Bond graph representing a mechatronic system with mixed energy domains and a controller subsystem.....77

Figure 5.6 Genotype-Phenotype mapping....78

Figure 5.7 Example of Genotype-Phenotype Mapping in the Electrical Circuit Domain....79

Figure 5.8 Resonator Unit and its Representations as both Bond Graph and Equivalent Circuit....81

Figure 5.9 MEMS Filter Topology I....82

Figure 5.10 MEMS Filter Topology II....83

Figure 5.11 Operator to Insert Bridging Unit....86

Figure 5.12 Operator to Insert Resonator Unit....86

Figure 5.13 Design Embryo of the MEM Filter....88

Figure 5.14 Fitness Improvement Curve....90

Figure 5.15 Frequency responses of a sampling of design candidates, which evolved topologies (and associated parameter sets) with larger numbers, K, of resonators as the evolution progressed. All results are from one genetic programming run of the BG/GP approach....91

Figure 5.16 Layout and bond graph representation of a design candidate from the experiment, with four resonator units coupled by three coupling units....92

Figure 5.17 A novel topology of MEM filter and its bond graph representation as evolved by the BG/GP approach using a semi-realizable function set....93

Figure 5.18 A folded-flexure comb-drive microresonator fabricated in a polysilicon surface microstructural process a) Layout; b) Cross-section A-A' (Fedder G. and Mukherjee T. [1996])....94

Figure 5.19 Major design variables for microresonators....95

Figure 5.20 Curve of Normalized SSE vs. Generation....99

CHAPTER I

INTRODUCTION

Several issues in design currently demand significant attention, including multi- and mixed-energy-domain systems, automated synthesis, and topologically open-ended design (Fig. 1.1).

First, there is a great demand for improved capabilities to design high-performance, multi-domain, dynamic systems, particularly in the area of mechatronics. The inclusion of components from multiple energy domains (such as electrical, mechanical, hydraulic, thermal and/or magnetic) and demands for rigorous performance and consideration of cost constraints make design of these systems very challenging.

Second, the need for automated synthesis is growing ever stronger. Design of such complex systems is typically an iterative process in a very large solution space, with multiple objectives. Traditional CAD design processes are tedious, inefficient and quite time-consuming.

Third, compared to parametric design, topological design is more challenging because it has a much larger and less well-defined search space. In order to achieve the desired performance of complex mechatronic systems, open-ended topological search is required to incorporate enough topological variations.

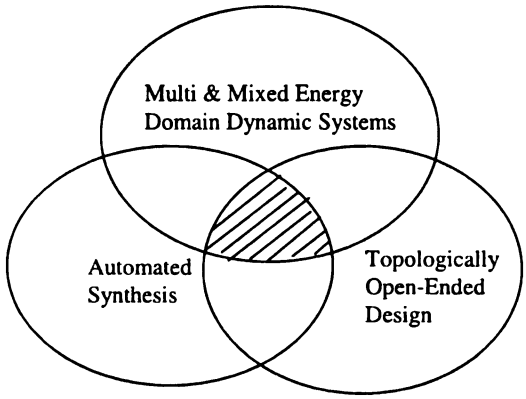


Figure 1.1 Requirements for Automated Design of Mixed-Domain Systems

1.1 Automated Synthesis

Computer-aided design (CAD) and computer-aided engineering (CAE) have been powerful tools that have revolutionized engineering practice and education since the advent of high-performance computers. The biggest influence of CAD and CAE is to give engineers the ability to design and test products on a testbed based on computational simulation before fabricating them. This ability has profound implications, especially because fabricating a product or system is time-consuming and costly. With the capability of numerical simulation in computers, engineers can compare more design concepts and prototypes, make judgments and tradeoffs, and be much more sure that the final product will satisfy the design specifications before he or she starts to fabricate it in the physical domain.

The computer tools we discussed above, including analysis tools that can simulate and measure the performance of designs, are passive design tools. Using such tools, the designer is at the center of the design scheme, controlling all aspects of the design process. The design tools just serve to provide information that the designers want or need, as feedback about performance of designs presented to them. Their roles are passive relative to the designer's, in the sense that they only "answer" or provide feedback when the designer "asks" a question and presents a design.

We describe another type of computer tool as active, rather than passive, in that it not only "answers" when the designer "asks", but also "thinks" when the designer is "thinking." In other words, such tools not only perform analysis, but suggest designs, with guidance from the designer only at a more abstract level. As a result, they not only gather and evaluate, but also to analyze and process information, make decisions, foster design insights and guide the design process.

While computers are definitely faster and more accurate in calculation than human beings, it is generally believed that they lack the cognitive capability humans use to make

creative designs and true inventions. This is not challenged in this work. It is also argued that in order to automate any phase of the design process, one must first understand the cognitive theory of how humans design; were this true, active computer tools could hardly be successful, because establishment of such a cognitive theory of human design is still an extremely distant goal. This argument sounds reasonable to many, but has one assumption that people should attend to carefully – that is, it is assumed in this argument that the human designer offers the only example of a successful design system. However, other successful design systems do exist. Nature is one of them. Even before the history of human beings, nature invented many wonderful designs of species that far exceed any human designs in terms of complexity, without any intervention of humans. Although nature spends a prohibitively long period of time (for a human designer) to "evolve" its designs, the ever-increasing speed and capacity of current computer technology provides a possible answer to shorten the time consumption to an acceptable range, for a design system that draws on principles of design from nature.

Over the past two decades, computational algorithms based on the principles of evolution first formulated by Charles Darwin have developed from academic curiosities into practical and effective tools for scientists and engineers. Evolutionary computation refers to a class of general-purpose search algorithms based on (admittedly very incomplete) abstraction of principles of biological evolution and natural selection. These algorithms implement biologically inspired computations that manipulate a population of candidate solutions (the "parents") to generate new variations (the "offspring"). At each step (or "generation") in the computation, some of the less promising candidates in the population are discarded and replaced by new candidates ("survival of the fittest"). The process continues until a satisfactory solution to the problem has been found. In this research, genetic programming (GP), a special form of evolutionary computation, is taken as the essential mechanism for design automation. While basing a system on evolutionary principles is certainly no guarantee that it can create new and innovative

designs, neither can one reject out-of-hand the possibility that such a system could do so without duplicating, or even emulating, the process performed by humans.

Genetic programming is an extension of the genetic algorithm, and it uses 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. Differences between GP and GA are summarized in Table 1.1. The most common form of genetic programming is due to John Koza [1992, 1994, 1999a], and uses trees to represent the entities to be evolved. Because GP (genetic programming) can manipulate variable-sized strings, it is especially useful for representing developmental processes. Most design methods require a preliminary design, which is a solution with enough components and a valid configuration, even if it is not a complete solution, in order to define the desired properties of a good solution. A developmental design process does not require a preliminary design, but only a design embryo, which need not contain all of the necessary components, or the necessary number of components, or a valid configuration, but only enough information to allow specifying the behaviors desired of the system (defining objectives and variables constrained, for example).

Table 1.1 Comparisons between GP and 'classical' GA

Properties	GA	GP
genome representation:	String	Tree
genome size:	Fixed length	Variable length
operators:	Representation-blind	Representation-specific

It is important to point out that when using passive design tools, designers' decision-making is biased by both the capabilities of simulation tools and the designer's experience and intuition. It is hard for the designer to make an "imaginative jump or creative leap" from one design candidate to another. But active design tools can free designers from this kind of "design fixation" and the limitations of conventional wisdom,

allowing them to explore a huge number of possible candidates for a design problem, and increasingly, the probability to discover novel designs uncharted before by human exploration.

1.2 Representation of Multidisciplinary Mechatronic Systems

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 the language of the bond graph, 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.

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis for physically-based dynamic systems. Bond graph models can describe the dynamic behavior of physical systems by the connection of idealized lumped-parameter elements based on the principle of conservation of power. Bond graphs consist of elements and bonds. There are several types of elements, each of which performs analogous roles across energy domains. The first type -- C, I, and R elements -- are , in their simplest forms, passive one-port elements that contain no sources of power, and represent capacitors, inductors, and resistors (in the electrical domain). A second type, Se and Sf, are active one-port elements that are sources of power and/or boundary conditions, and that represent effort sources and flow sources, respectively (for example, sources of specified voltage or current, respectively, in the electrical domain). A third type, TF and GY, are two-port elements in their simplest forms, and represent transformers and gyrators, respectively. Power is conserved in these elements. A fourth type, denoted as 0 and 1 on bond graphs, represents junctions, which are three-port (or more) power conserving elements. They serve to interconnect other elements into

subsystems or system models. Other types of multiport elements may be defined, but will not be used here.

Some example bond graph models are shown below. Figure 4 consists of a mechanical system at the left, an electrical system at the right, and a bond graph representation at the center. The bond graph representation includes a Se , 1-junction, C, I, and R elements, and that same bond graph represents either a mechanical mass, spring and damper system, or an RLC electric circuit. S_e corresponds with force in the mechanical system and voltage in electrical system. The 1-junction implies a common velocity for 1) the end of the spring, 2) the end of the damper, and 3) the mass in the mechanical system, and implies that the current in the RLC loop is common in the electrical system. The R, I, and C represent the damper, inertia (of mass), and spring in the mechanical system, or the resistor, inductor, and capacitor in the electrical circuit.

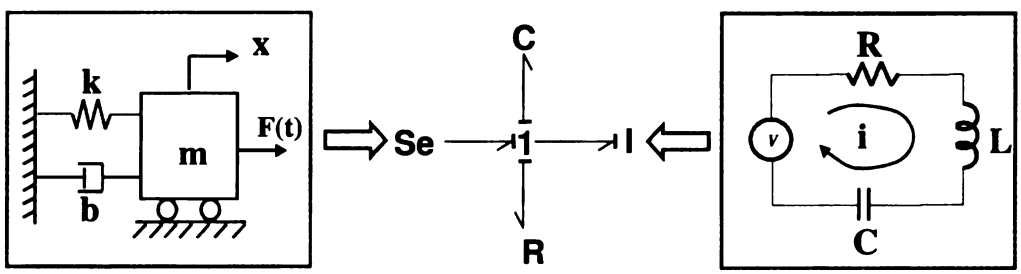


Figure 1.2 Bond Graphs Representation of Mixed-Domain Systems

Bond graphs have two major advantages for design application – their efficiency for evaluation of design alternatives and the natural combinatorial features of bond and node components for generating design alternatives.

The analysis efficiency of the bond graph model results because the causal relationships and power flow between elements and subsystems reveal certain system properties and inherent characteristics very efficiently. A set of state variables is easily determined and the state equations can be generated systematically. Particular

efficiencies are possible in the classification of models as to whether or not they merit dynamic simulation.

The other characteristic of bond graphs as shown in Figure 1.3 is their graphical (topological) structure, which allows structural manipulation separate from the equations. This means that any system model can be generated by a combination of bond and node components, because of their free composition and unbounded growth capabilities. Therefore it is possible to span a large search space, refining simple designs discovered initially, by adding size and complexity as needed to meet complex requirements.

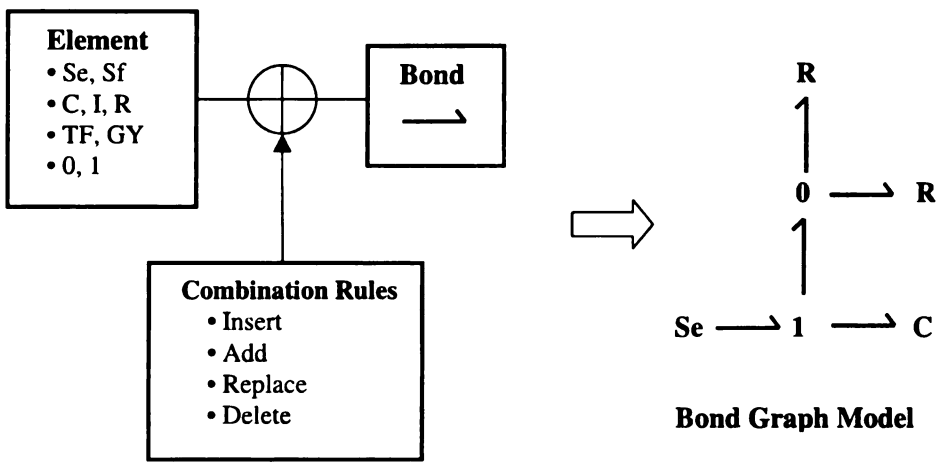


Figure 1.3 The combinatorial nature of bond graphs generation

1.3 Related Work

1.3.1 Bond Graphs

Rosenberg and many others have described bond graph methods in detail in the literature (see, for example, Karnopp, Margolis and Rosenberg [1999], Rosenberg [1992, 1993a, 1993b, 1996]). Prabhu [1989] presents a set of basic theorems for using a variant of bond

graphs in design. They exploit the graph nature of bond graphs for design. A set of graph-rewriting rules to generate bond graph models that represent feasible physical systems is presented in Hoover and Rinderle [1989]. An important feature of this work is the exploration of all the behaviors a component might have. Stein and Louca [1995] develop a two-level-based Component Modeling Procedure to exploit the power of several existing model order deduction algorithms. This procedure is implemented in a computer program, CAMBAS. CAMBAS uses expandable bond graph models and automatically builds global bond graphs of systems according to the design engineer's selection of templates. Sharpe and Bracewell [1995] present the use of bond graph reasoning for the design of interdisciplinary schemes. They describe how conceptual scheme synthesis may be assisted and structured by the use of functions-mean trees developed by the application of bond-graph-inspired rules. Coelingh et al. [1998] present a computer-based design tool for conceptual design of mechatronic motion systems. Youcef-Toumi [1999] introduces an algorithm which identifies automatically the physical components and/or subsystems that are responsible for zero dynamics. Redfield [1999] demonstrates the value of using bond graphs as a conceptual or configurational design tool for dynamic systems, using as an example a continuously variable transmission.

1.3.2 GA/GP

Numerous design-generating tools using GA and GP by members of the Genetic Algorithms Research and Applications Group ("GARAGe") are presented by Goodman and his co-authors (Raymer et al. [1996], Goodman [1996], Goodman et al. [1997a], Wang et al. [1997b], and Eby et al. [1998]). (One of the most powerful and widely used GP systems, Lil-gp, was developed in the GARAGe.) Carlson-Skalak et al. [1998] have developed a catalog design method using an evolutionary algorithm, applied to a manufacturing floor piping network. This approach allows for simultaneous alterations of configurations and components. Koza et al. [1997a, 1997b] present a single uniform

approach using genetic programming for the automatic synthesis of both the topology and sizing of a suite of various prototypical analog circuits, including low-pass filters and operational amplifiers. Koza et al. [1999b] present a general automated method for synthesizing the design of both the topology and parameter values for controllers. This method automatically makes decisions concerning the total number of processing blocks to be employed in the controller, the type of each block, the topological interconnections between the blocks, the values of all parameters for the blocks, and the existence, if any, of internal feedback between the blocks of the overall controller. It has already shown itself to be extremely promising, having produced a number of patentable designs for useful artifacts, and is the most closely related approach to that proposed here; however, it works in a single energy domain. Danielson, Foster and Frincke[1998] use both bond graphs and a genetic algorithm to design a 2-stroke combustion engine. They start from a preliminary design, find near-optimal values for 15 physical parameters for a combustion engine, but without allowing topological variation. Tay, Flowers and Barrus [1998] use a genetic algorithm to vary bond graph models. This approach adopts a variational design method, which means they make a complete bond graph model first, then change the bond graph topologically using a GA, yielding new design alternatives. Their goal is to provide a wider range of possible designs, and is closely related to that presented here, but within a topologically more limited search space.

1.3.3 Automated Design Theory and Practice

Reich [1995] presents a critical review of General Design Theory (GDT), a mathematical framework for design. He reviews the assumptions (axioms) and predictions (theorems) of GDT with respect to design and illustrates them with simple examples. Gero [1995]. investigates evolutionary systems as computational models of creative design and studies the relationships among genetic engineering, style emergence, and complex evolution. Kota and Lee [1993] present a configuration design technique employing a functional

reasoning approach. As in traditional catalog design, a configuration is formed based on functions, and then components are selected. Chakrabarti and Bligh [1994, 1996a, 1996b] describe one approach to synthesis of solutions to a class of mechanical design problems; these involve transmission and transformation of mechanical forces and motion, and can be described by a set of inputs and outputs. The approach involves (1) identifying a set of primary functional elements and rules of combining them, and (2) developing appropriate representations and reasoning procedures for synthesizing solution concepts using these elements and their combination rules. Schmidt and Cagan [1996] have used a grammar-based system for design in which the grammar's vocabulary represents functions or subfunctions. Rosen and Peters [1996] seek to demonstrate the diversity of applications of topology within engineering design. A complementary goal is to introduce the engineering design community to topology as a rich, formal, well-established mathematical discipline that may be of value for wider study. Whitney [1996] describes fundamental reasons, based on natural phenomena, that keep mechanical design from approaching the ideal of contemporary VLSI design methods. Campbell et al. [1999] provide an introduction to a new design methodology known as A-Design, which combines aspects of multi-objective optimization, multi-agent systems, and automated design synthesis.

Design automation is undoubtedly a very difficult task. However, we have seen some very successful applications in specific areas. For example, analog/mixed-signal design is one of the most dynamic and vital research areas in both academy and industry. In industry, two leading companies in the area, ADA in Canada and Neolinear in the US, have done much breakthrough research and successfully applied their research results in the Electronic Design Automation (EDA) industry. Both companies, I believe, have a focused application of computational intelligence techniques in their products. Take the instance of ADA: the company has gathered many famous researchers specializing in

computational intelligence as well as analog CAD. Madan M. Gupta, an IEEE fellow and pioneer in fuzzy and neural systems, is a member of the advisory board. Trent McConaghy, the Chief Scientist of ADA, is also a renowned specialist on artificial neural networks, fuzzy logic, evolutionary algorithms, pattern recognition, and classification. Neolinear, on the other hand, has Rob Rutenbar in its research advisory board. As the Director of the Center for Electronic Design Automation (CEDA) at CMU, Rutenbar is leading one of the most influential groups in analog/mixed-signal CAD. In one of his publications, he explicitly states that he uses Parallel Recombinative Simulated Annealing (PRSA), an idea originated from Goldberg's combining of a genetic algorithm and simulated annealing optimization. Though striking and quite successful in their first attempts, the biggest limitation of these industry-oriented approaches is that they only accept fixed topologies. In academic circles, much research has been done on design automation of single-domain systems capable of topological exploration using an evolutionary computation approach. 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 1995). Some GA approaches evolve both topology and component parameters; however, they typically allow only a limited amount of components to be evolved (Lohn 1999). Using netlists as the representation technique for the circuit, and genetic programming as the evolutionary tool, Koza has developed very successful approaches to deal with circuit synthesis problems, evolving topologies and parameters simultaneously (Koza, 1999). Although their work basically achieves good results in analog circuit design, it is not easily extendable to interdisciplinary systems like mechatronic systems.

Mechatronic system design differs from conventional design of electronic circuits, mechanical systems, and fluid power systems in part because of the need to integrate several types of energy behavior as part of the basic design (Coelingh [1998]). Multi-

domain design is difficult because such systems tend to be complex and most current simulation tools operate over only a single domain. In order to automate design of multi-domain systems, such as mechatronic systems, a new approach is required. The essential goal of the work reported in this dissertation is to develop an automated procedure capable of designing mechatronic systems to meet given performance specifications, subject to various constraints. The most difficult aspect of the research is to develop a method that can explore the design space in a topologically open-ended manner, yet find appropriate configurations efficiently enough to be useful.

1.3.4 The BG/GP Approach

The goal of this thesis is to develop an integrated design tool for the purpose of automatic, topologically open-ended synthesis of multi-energy-domain systems. In order to achieve this goal, a novel approach is needed, to satisfy the three principal requirements – multi-energy-domain design, automated synthesis, and topologically open-ended design. To date, most design approaches have lacked at least one of these characteristics: domain independence, efficient analysis, or broad search. Some do strong search but weak analysis, while others do good analysis but weak search. Bond graphs are domain independent and efficient for classification and analysis of models, allowing rapid determination of various types of acceptability or feasibility of candidate designs, thereby sharply reducing the time needed for analysis of designs which are infeasible or otherwise unattractive. Genetic programming is well recognized as a powerful tool for open-ended search. The combination of these two powerful methods, called the BG/GP approach, is therefore an appropriate target for a better system for synthesis of complex mechatronic systems. Figure 1.4 shows a general flow chart of the BG/GP design process. Design specifications, including problem descriptions, design objectives, design constraints, etc., are first defined. After that, bond graphs are used to model and represent dynamic systems to be designed. In the BG/GP approach, bond

graph representations for dynamic systems are used for each design candidate of the design population of each generation in a genetic programming run. The genetic programming technique is the combinatorial basis of the BG/GP approach to realize design automation. It is genetic programming that possesses the mechanisms to generate a preliminary population of design candidates, to present each design individual for evaluation according to a specified fitness function, to reconfigure the topologies and/or parameters of design candidates (represented by bond graphs) in the population, and to guide the design process to the next generation by producing a new population of design candidates, typically with better average performance.

This loop of design generation, evaluation, reconfiguration and guidance is typically iterated until at some generation, all design specifications are met by one design candidate or a group of design candidates. If so, the design process can be ended and design candidate/candidates satisfying design specifications can be saved for further analysis and post-processing.

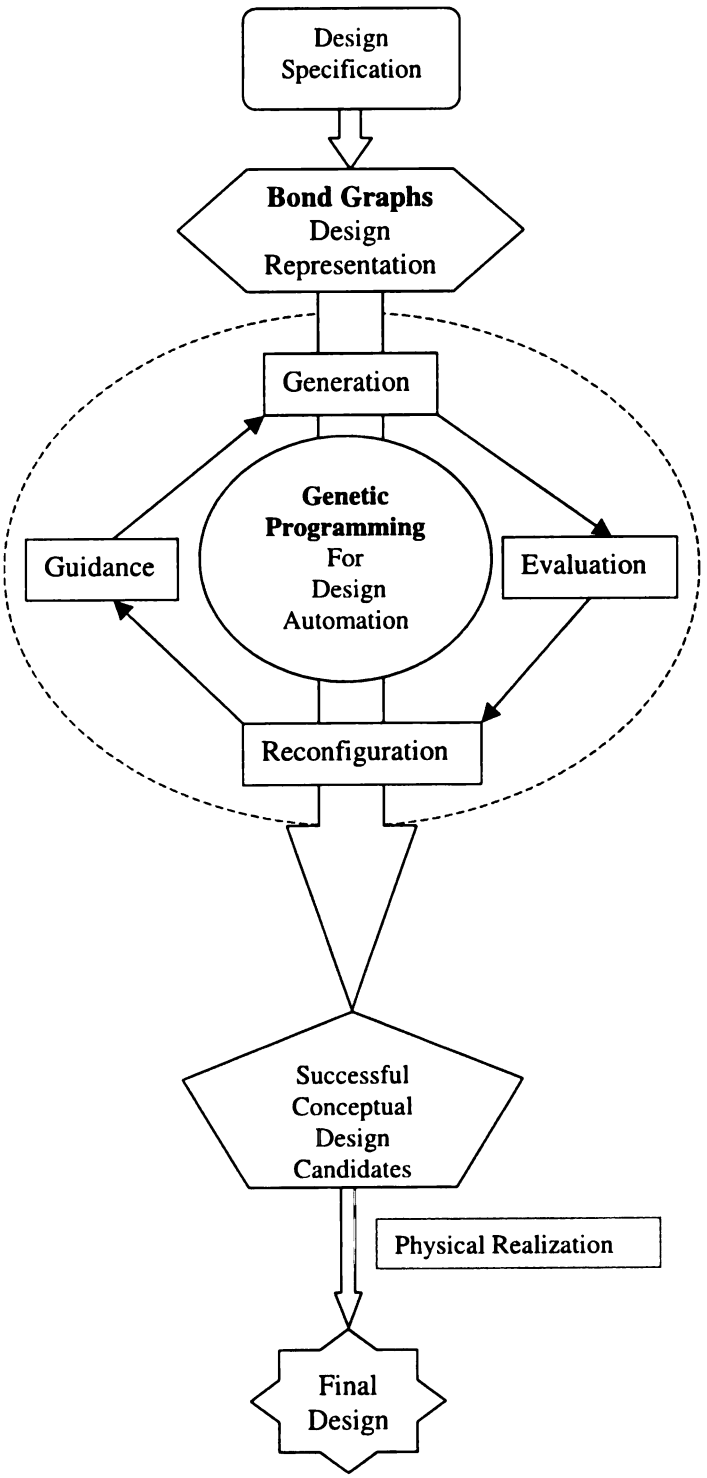


Figure 1.4 General flow chart of the BG/GP design

Table 1.2 summarizes the similarities and differences between the proposed BG/GP approach and several others. In this table, parametric variation means variation of parameters within a fixed configuration. Limited topological variation means the configuration can be changed, but only within limited bounds. Open-ended topological variation means the configuration can be changed not only topologically, but also by increasing or decreasing the number of components and altering their interconnections, without fixed bounds.

Table 1.2 Comparisons of various design approaches

Properties	Design with Bond Graphs	Design with GA	Design with GP	Design with Bond Graphs & GA	BG/GP approach
Multi-domain	X			X	X
Open-ended Topological variation			X		X
Developmental Process			X		X
Automated synthesis		X	X	X	X
Design Optimization		X	X	X	X
Efficient evaluation	X			X	X

Automatic synthesis means that the iterative analysis and design search process can be performed without a designer's intervention. Developmental process means that the designer need only set the embryo design initially (thereby defining the measurable quantities specifying the problem to be solved), and it evolves, generating a complete design solution. Efficient evaluation means that infeasible designs can be rapidly detected without the need for full simulation of design performance.

1.4 Contributions of the Dissertation

With mechatronics emerging as an independent and integrated discipline of the 21st century, this dissertation is of significance because it is one of the first endeavors to address the challenging issue of design automation of mechatronic systems. The main contributions of the dissertation are:

- A general framework, the BG/GP approach, for automated conceptual design of mechatronic systems, is described. The approach combines search capability of genetic programming to explore open-ended design space automatically and bond graphs to unify representation of mixed-domain systems across different physical domains in mechatronic systems.
- The BG/GP approach has been verified in the electrical domain in an electrical analog filter design problem.
- The BG/GP approach has been verified in the mechanical domain in a mechanical printer redesign problem.
- Instructive comparisons between Mechatronic Design Automation (MDA) and Electronic Design Automation (EDA) have been made and the promise of MDA has been suggested.
- A framework of hierarchical evolutionary synthesis of MEMS has been recommended and further research directions have been indicated.
- System level behavioral synthesis of MEMS has been studied and implemented using extended BG/GP approach.
- Second level layout synthesis of MEMS has been studied and implemented using a constrained genetic algorithm.

1.5 Organization

A novel BG/GP design approach is presented in this dissertation. The early chapters introduce the background and explain the fundamental elements of the theory and the later chapters test the theory in various facets and discuss insights gained through experiments.

Chapter 2 discusses advantages of bond graphs as a tool for design representation; some implementation issues in this research are also addressed. Chapter 3 introduces fundamentals of genetic programming and explains its functionality in design generation, evaluation, reconfiguration and guidance. The preparatory steps needed to apply this technique in the BG/GP approach are also discussed. Chapter 4 includes case studies of three real-world engineering design problems. Through experiments of an electrical analog filter design, and a vibration absorber design for a mechanical printer system design, various facets of using the BG/GP approach to facilitate and automate the design process for mixed-domain dynamic systems are studied and several insights regarding design are gained in the process. While these design cases are basically in the macro-world, in Chapter 5, we extend the BG/GP approach to a micro-scale domain and discuss the research of hierarchical evolutionary synthesis of MEMS. First, we stratify the design process of MEMS into several levels. At the system level, after integrating severe topological constraints imposed by the specific application, we show that the BG/GP approach can be used to automate system-level design or conceptual design of a general class of dynamic systems, exemplified by a MEM filter design problem. At the second level, we synthesis the layout of the cell component exemplified by a resonator unit using constrained genetic algorithm. Some further research directions are also indicated. Finally, Chapter 6 provides conclusions and suggestions for further research.

CHAPTER II
BOND GRAPHS

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems. 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 (Karnopp, Margolis and Rosenberg [2000], Rosenberg [1992, 1993a, 1993b, 1996]). Much recent research has explored bond graphs as tools for design (Sharpe and Bracewell [1995], Tay, et al. [1998], Youcef-Toumi [1999], Redfield [1999]).

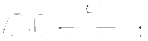
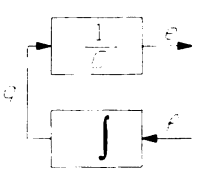
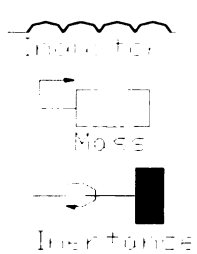
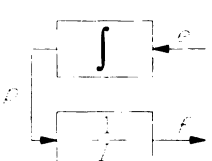
The constitutive equations of the bond graph elements are readily introduced via examples from the electrical and mechanical domains. The nature of the constitutive equations imposes demands on the causality of the connected bonds. Bond graph elements are drawn as letter combinations (mnemonic codes) indicating the type of element. The bond graph elements are the following (Broenink [1999]):

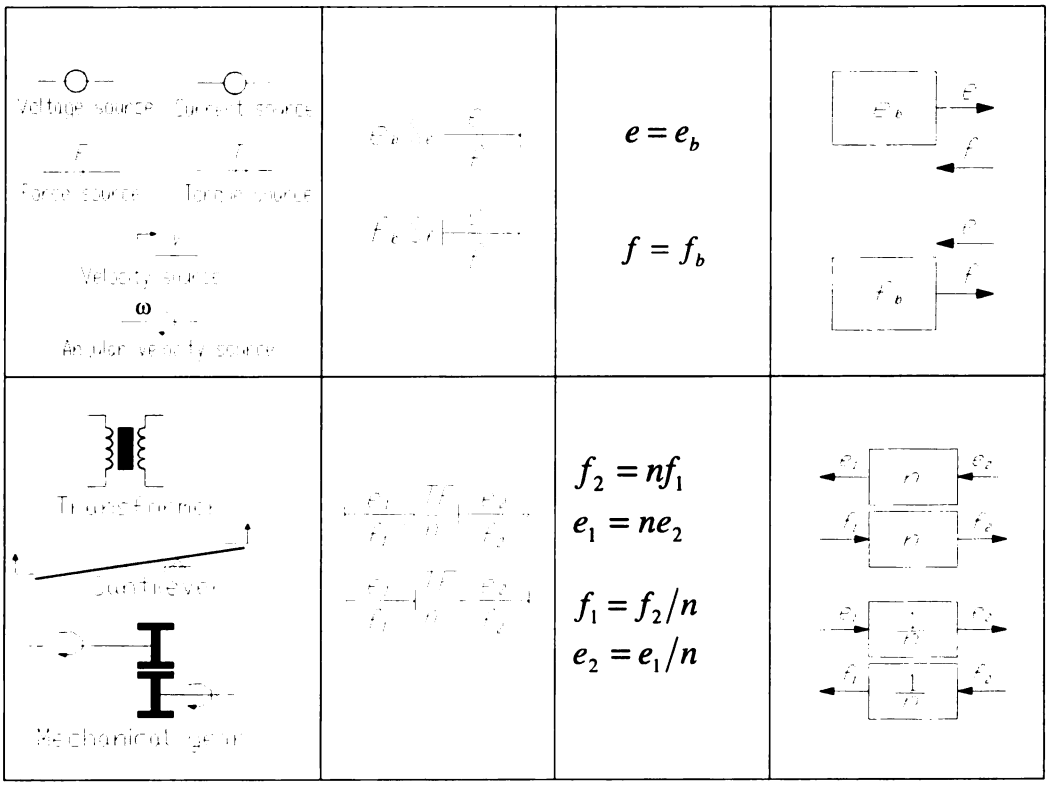
- C, storage element for a q-type variable, e.g. capacitor (stores charge), spring (stores displacement).
- I, storage element for a p-type variable, e.g. inductor (stores flux linkage), mass (stores momentum).
- R, resistor dissipating free energy, e.g. electric resistor, mechanical friction.
- Se, S_f, sources, e.g. battery (voltage source), gravity (force source), pump (flow source).
- TF, transformer, e.g. an electric transformer, toothed wheels, lever.

- GY gyrator, e.g. electromotor, centrifugal pump.
- 0, 1, 0– and 1–junctions, for ideal connection of two or more sub-models.

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. Table 2.1 summarizes effort and flow variables in translational, rotational, electrical and hydraulic domains, respectively, with their corresponding bond graph representations.

Table 2.1 Flow and effort variables for each domain

Domain-specific symbols	Bond graph element	Equations	Block diagram expansion
		$e = \frac{1}{C} q$ $q = \int e dt + q(0)$	
		$f = \frac{1}{I} p$ $p = \int e dt + p(0)$	



2.1 Causality of Bond Graphs

One of the important concepts in bond graph theory is causality. If two components are bonded together in a bond graph, we can think of one effort as causing one component to respond with a flow while the flow causes the first component to respond with an effort. Thus the cause-effect relations for efforts and flows are represented in opposite directions. A single mark on a bond, which is called the causal stroke, indicates how effort and flow simultaneously are determined causally on a bond.

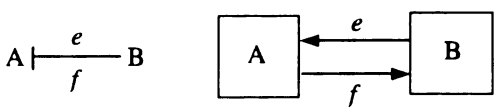


Figure 2.1 The meaning of causal stroke

Causal analysis can give insight into the correctness and competency of the model. This concept plays a great role in determining the feasibility of a design very simply at an early stage.

Dependent on the kind of equations of the elements, the element ports can impose constraints on the connected bonds. There are four different constraints, which should be treated before a systematic procedure for causal analysis of bond graphs is discussed (the reader unfamiliar with these constraints is directed to Appendix A for that treatment) (Broenink [1999]).

2.2 Bond Graph Evaluation

To take advantage of the causal analysis that is possible for bond graphs, a two-stage evaluation procedure is executed to evaluate bond graph models. The first, causal analysis (Karnopp et al. [2000]), allows rapid determination of feasibility of candidate designs, thereby sharply reducing the time needed for analysis of designs that are infeasible. Then, for those designs "passing" the causal analysis, the state model is automatically formulated. The process is illustrated in Figure 2.2.

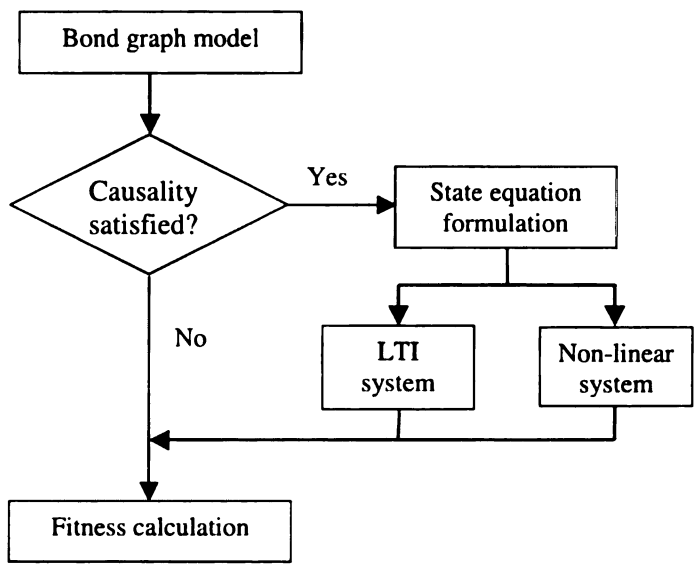


Figure 2.2 Evaluation flow of bond graphs models

2.2.1 Causality Analysis

The causality assignment procedure is described as follows (quoting from Broenink [1999]) (refer to Figure 2.3):

> "1a. Choose a fixed causality of a source element, assign its causality, and propagate this assignment through the graph using the causal constraints. Go on until all sources have their causalities assigned.
>
> 1b. Choose a not-yet-causally-assigned port with fixed causality (non-invertable equations), assign its causality, and propagate this assignment through the graph using the causal constraints. Go on until all ports with fixed causality have their causalities assigned.
>
> 2. Choose a not-yet-causally-assigned port with preferred causality (storage elements), assign its causality, and propagate this assignment through the graph using the causal constraints. Go on until all ports with preferred causality have their causalities assigned.
>
> 3. Choose a not-yet-causally-assigned port with indifferent causality, assign its causality, and propagate this assignment through the graph using the causal constraints. Go on until all ports with indifferent causality have their causalities assigned.
>
> Often, the bond graph is completely causally determined after step 2, without any causal conflict (all causal conditions are satisfied). If this is not the case, then the moment in the procedure where a conflict occurs or where the graph becomes completely causally determined, can give insight into the correctness and instantiability of the model."

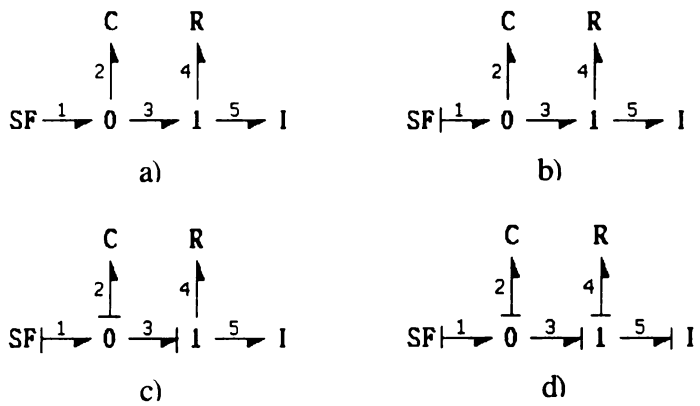


Figure 2.3 Example of causality assignment

2.2.2 Model insight via causal analysis

As Broenink [1999] continues:

> "When the bond graph is completely causally determined after step 2, without any causal conflict, each storage element represents a state variable, and the set of equations is an explicit set of ordinary differential equations (not necessarily linear or time invariant).
>
> When the bond graph is completely causally determined after step 1a, the model does not have any dynamics. The behavior of all variables now is determined by the fixed causalities of the sources. If a causal conflict arises at step 1a or at step 1b, then the problem is ill posed. The model must be changed, by adding some elements. An example of a causal conflict at step 1a is two effort sources connected to one 0-junction. Both sources 'want' to determine the one effort variable.
>
> In case of a conflict at step 1b, a possible adjustment is changing the model of some fixed–causality element such that its describing equations

become invertible, and thus the fixedness of the constraint disappears. When a conflict arises at step 2, a storage element receives a non—preferred causality. This means that this storage element does not represent a state variable. The initial value of this storage element cannot be chosen freely. Such a storage element often is called a dependent storage element. This indicates that a storage element was not taken into account during modeling, but it should be there from physical systems viewpoint. It can be deliberately omitted, or it might have been neglected in the modeling. In a hoisting device example, the load of the hoist (I-element) is such a dependent storage element. Elasticity in the cable was not modeled. If it had been modeled, a C-storage element connected to a 0-junction between the cable drum and load would have appeared.

When step 3 of the causality algorithm is necessary, a so-called algebraic loop is present in the graph. This loop causes the resulting set differential equations to be implicit. Often this is an indication that a storage element that should be there from a physical systems viewpoint was not modeled."

2.2.3 State Equation Formulation

For those designs "passing" the causal analysis, the state model may be automatically formulated. However, as bond graphs are pictorial descriptions of dynamic systems, to obtain the numerical performance of the dynamic systems, it is necessary to derive a mathematical model from the pictorial description. There is a systematic procedure to transform a bond graph representation of a dynamic system to a state equation (Rosenberg, [1971]) or transfer function. In our research, we focus on the problem of

state equation formulation. The details of this formulation procedure are provided in Appendix B.

2.3 Simplification of Bond Graphs

Bond graph models can be simplified in some cases. This fact is important in our research because some seemingly different topologies of bond graph models are actually the same after simplification. As comparison of topologies of designs for dynamic systems (represented by bond graphs) is useful in many applications, it is desirable to develop an algorithm to automatically simplify bond graph topology, rather than to do it manually. Currently we have implemented three simplification rules as follows:

1). Rule 1, elimination of redundant junctions. Junctions can be removed from a graph if the energy flow is not branched at the junction, nor a signal bond connected to the junction. Please refer to Figure 2.4

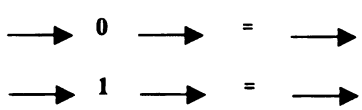


Figure 2.4 Elimination of redundant junctions in bond graphs

2). Rule 2, merging of junctions. Two junctions of the same type can be joined if there is exactly one power bond between the junctions. The simplification is carried out by removing the bond between the junctions and transplanting all connections of one junction to the other junction. The first junction can then be removed. Please refer to Figure 2.5

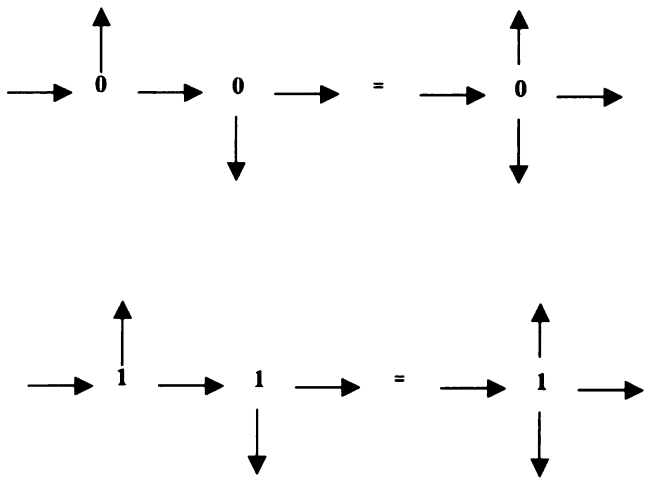
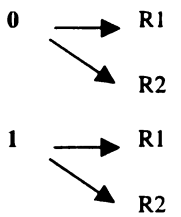


Figure 2.5 Merging of junctions in bond graphs

3). Rule 3, merging of elements. Elements of the same type connected to the same junction can be joined. The simplification is carried out by calculating the expression for the new parameter value of the element, replacing one of the parameters by the new expression and removing the other element and its power bond. Please see details in Figure 2.6.

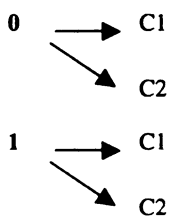
1) R



$1/Req = 1/R1 + 1/R2$

$Req = R1 + R2$

2) C



$Ceq = C1 + C2$

$1/Ceq = 1/C1 + 1/C2$

3) I

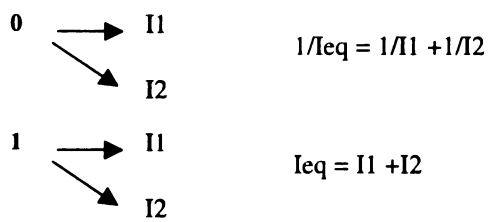


Figure 2.6 Merging of elements in bond graphs

We implemented this algorithm in the Simplification( ) member function of the BondGraph class in our code. Applying simplification for a bond graph is very direct, as shown in the simple illustrative example:

BondGraph A;
A.Simplification( );

The pseudo code for the simplification algorithm for bond graphs is listedbelow:

```
Input : Bond graph output generated by GPBG
Output : Simplified bond graph model
Procedure
  begin
   i = 0
   for all junction(i)
    apply Rule 1
    i++
   i = 0
   for all junction(i)
    apply Rule 2
    i++
   j = 0
   for all element(j)
    apply Rule 3
    j++
  end
```

An example showing a bond graph model before and after simplification is shown in Figure 2.7. This is a result taken from a BG/GP run for the filter design problem. The top figure is the bond graph model is taken from generation 96 of a typical BG/GP run for the filter design problem. It is not simplified at the moment, with several elements that can be merged highlighted by dashed circles. After the simplification algorithm, the resulting simplified bond graph model is shown in the bottom figure. The two bond graphs models have identical dynamic behaviors, but the simplified one has fewer elements and can be physically realized with fewer physical components. Another purpose of using simplification methodology is that when comparing two structures, these two seemingly different topologies are actually the same in terms of dynamic behavior. Thus we can more easily draw conclusions about the differences between two bond graphs if we compare simplified structures.

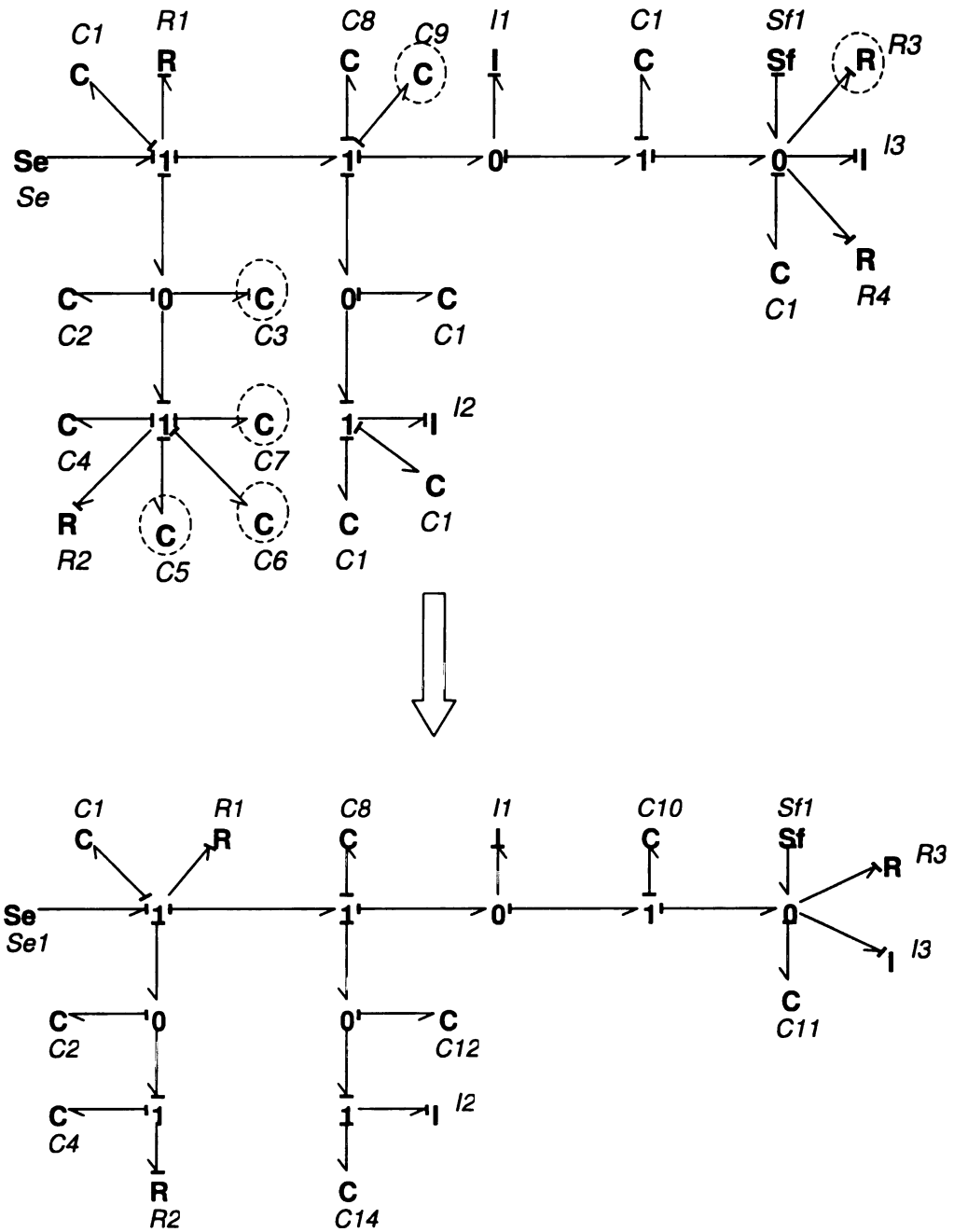


Figure 2.7 An example of bond graph simplification

2.4 Strengths of Bond Graphs

In summary, bond graphs have three embedded strengths for design applications. First, multi-domain systems (electrical, mechanical, hydraulic, pneumatic, thermal) can be modeled using a common notation, which is especially important for design of mechatronic systems. Second, the graphical (topological) structure characteristic of bond graphs allows their generation by combination of bond and node components, rather than by specification of equations. This means that any system model can be generated by a combination of bond and node components, because of their free composition and unbounded growth capabilities. Therefore it is possible to span a large search space, refining simple designs discovered initially, by adding size and complexity as needed to meet complex requirements. Third, in causality analysis, the causal relationships and power flow among elements and subsystems can reveal various system properties and inherent characteristics that can make the model unacceptable, and therefore make dynamic simulation unnecessary. While the strong typing used in the GP system will not allow the GP system to formulate "ill-formed" bond graphs, even "well-formed" bond graphs can have causal properties that make it undesirable or unnecessary to derive their state models or to simulate the dynamics of the systems they represent. Causality analysis is fast, and can rapidly eliminate further cost for many models that are generated by the genetic programming system, by performing assignment of effort and flow variables and making checks for violations of the appropriate constraints. This simple filtering cuts the evaluation workload dramatically. For systems passing causal analysis, state equations are easily and systematically derived from bond graph models. Then various analyses or simulation based on the state model allow computation of the desired performance measures.

CHAPTER III
EVOLUTIONARY DESIGN

As its name implies, evolutionary design uses concepts borrowed from Darwin's concept of evolution to 'breed' good solutions to design problems. The potential success of this idea is based on the observation that nature is a great non-human designer -- without any intervention by humans, nature has created many varied species that far exceed any man-made designs in terms of complexity, during the last billion years. However, in design of man-made artifacts, the engineer cannot afford to wait for the millions of years that the evolution of organizations in nature has taken. The much-simplified computational model used in evolutionary design and the ever-increasing speed and capacity of current computer technology can help to shorten the time consumption for design of engineered artifacts to an acceptable range.

In this research, we focus on a special type of evolutionary computation technique, namely 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 is due to John Koza [1992,1994,1999], and uses trees to represent the entities to be evolved. Genetic programming can manipulate variable-size strings and can be used to "grow" trees that specify increasingly complex bond graph models, as described below. If the scope and analysis efficiency of the bond graph model can be successfully integrated with the impressive search capability of genetic programming when utilized to its full potential, an extremely capable automated synthesis procedure, without need for user intervention, should result.

3.1 Evolutionary Design with Bond Graphs

3.1.1 Generation of Design Candidates

Unlike most other approaches, genetic programming will generate a population of design candidates at one time, rather than just one single design. If we look at designing as a search process for optimized designs, genetic programming, as a design automation and optimization approach, starts the search not at one single point, but from a 'population' of points scattered in the search space. Genetic programming takes advantage of the collective information acquired from the whole population of design candidates, feeds it back to influence the collective behaviors of the population through fitness evaluation of each individual, and guides them to search for better positions/points by imposing a search pressure. In the process, each individual may reconfigure itself through crossover and mutation operations. This is an important feature to have the ability to explore a topologically open-ended search space. In the next section, we will first discuss how to generate an individual design for a dynamic system represented as a bond graph.

3.1.2 Bond Graph Construction

A typical GP system evolves GP trees, rather than more general graphs. However, bond graphs can contain loops, so we do not represent the bond graphs directly as our GP "chromosomes." Instead, a GP tree specifies a construction procedure for a bond graph. Bond graphs are "grown" by executing the sequence of GP functions specified by the tree, using the bond graph embryo as the starting point.

Defining a proper function set to generate candidates 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 designed system. We define the GP functions and

terminals for bond graph construction in Table 3.1. There are four types of functions: first, *add* functions that can be applied only to a junction and which add a C, I, or R element;

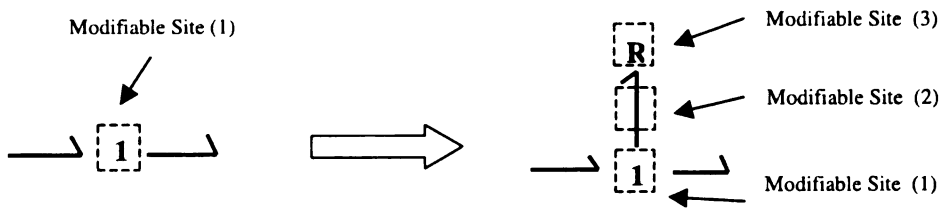
Table 3.1 Definition of function set

Name	#Args	Description
add_C	4	Add a C element to a junction
add_I	4	Add an I element to a junction
add_R	4	Add an R element to a junction
insert_J0	3	Insert a 0-junction in a bond
insert_J1	3	Insert a 1-junction in a bond
replace_C	2	Replace the current element with a C element
replace_ I	2	Replace the current element with an I element
replace_ R	2	Replace the current element with an R element
+	2	Add two ERCs
-	2	Subtract two ERCs
enda	0	End terminal for add element
endi	0	End terminal for insert junction
endr	0	End terminal for replace element
erc	0	Ephemeral random constant (ERC)

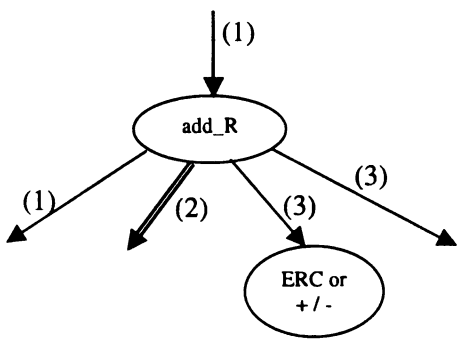
second, *insert* functions that can be applied only to a bond and which insert a 0-junction or 1-junction into the bond; third, *replace* functions that can be applied only to a node and which change the type of element and corresponding parameter values for C, I, or R elements; and fourth, arithmetic functions that specify arithmetic operations and are used to determine the numerical values associated with components.

Some typical operations -- add_R (a 1-port resistor) and insert_J0 (a 0-junction) -- are explained in detail as follows. In Figure 3.1, the R element is added to an existing junction by the add_R function. This function adds a node with a connecting bond. An R element also requires an additional parameter value (ERC -- ephemeral random constant). Please note that in the GP tree fragment, a single line is used to denote a node site, which is either a component or a junction in the bond graph fragment, while a double line is used to denote a bond site. The insert_J0 function can be applied only at a bond, and performs insertion of a 0-junction at the given modifiable site (refer to Figure 3.2).

Inserting a 0-junction between node R and a 1-junction yields a new bond graph (the right side of Figure 3.2). As a result, three new modifiable sites are created in the new bond graph. At each modifiable site, various bond growth functions can be applied, in accordance with its type. In GP terminology, this is a strongly typed GP.

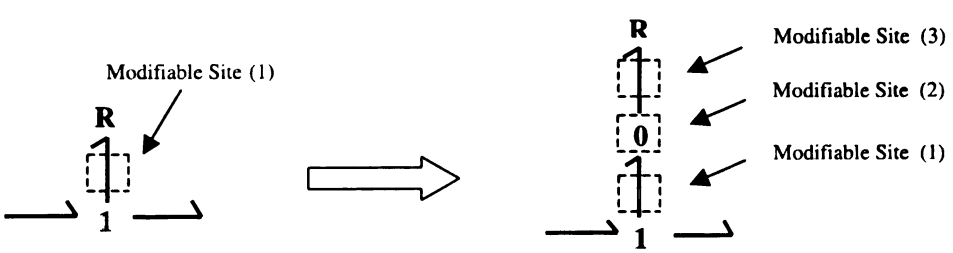


(a) bond graph fragment

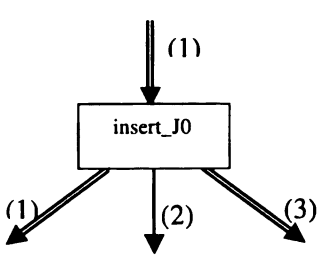


(b) GP tree fragment

Figure 3.1 Illustration of add_R operator



(a) bond graph fragment



(b) GP tree fragment

Figure 3.2 Illustration of insert_J0 operator

Figure 3.3 shows an example of a GP tree, generated at random from the embryo root node. There are three modifiable embryo sites, denoted "1" (bond graph node), "a" (bond), and "2" (bond graph node). Each is denoted by an edge of the GP tree. Following edge 1 first, shows that an I element (I3 in Figure 3.4) is added by the add_I to the 1-junction (11) of the bond graph, together with the I element's parameter value and a new bond. The result is to preserve modifiable site "(1)" and to add modifiable sites "(b)" and "(3)". The next set of operations under add_I in the GP tree shows that all three sites happen to have been made unmodifiable in the example tree by appending of *end* functions.

Turning next to the edge labeled "a", we see that the first function applied to it is "end." That bond site is thereby made unmodifiable. On the other hand, site "(2)" is the locus of additional bond graph growth. A C element, C4 in Figure 3.4, is added by add_C to the 0-junction (02). In the next operation, insert_J1, a 1-junction (15) is inserted between the 0-junction (02) and C4. After the remaining operations, the bond graph of Figure 3.4 is generated from the GP tree of Figure 3.3.

3.1.3 Reconfiguration of Designs

Reconfiguration of design candidates is performed mainly through crossover and mutation operations embedded in the genetic programming technique (refer to Figure 3.5).

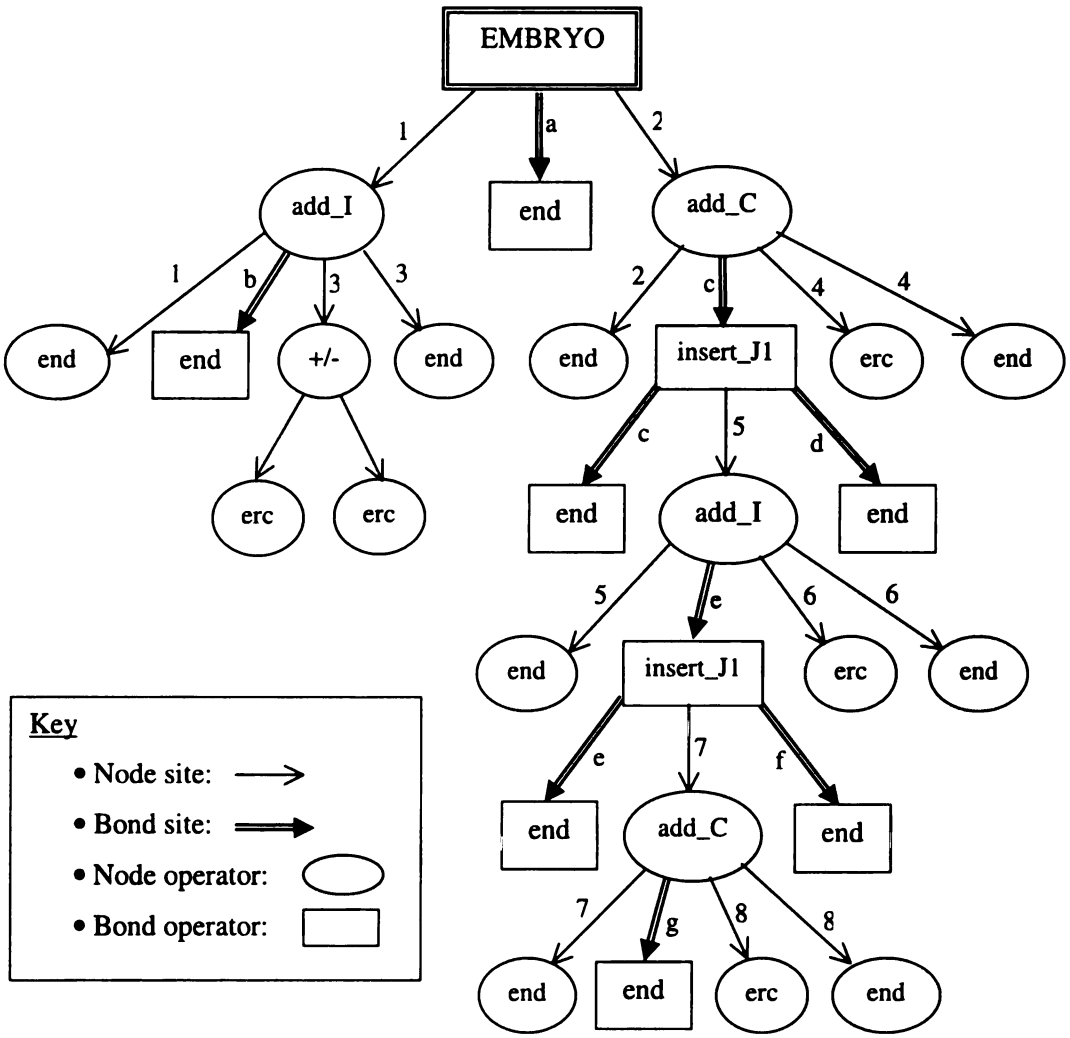


Figure 3. 3 An example of a GP tree

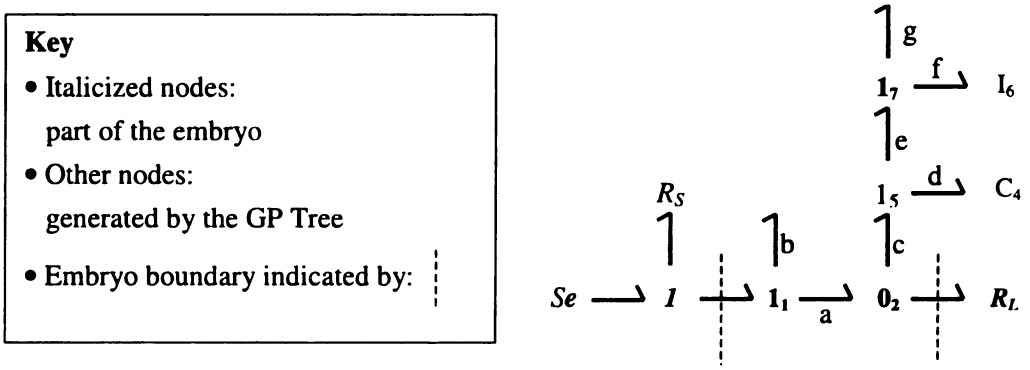


Figure 3.4 The bond graph model generated by the GP tree of Figure 3.3

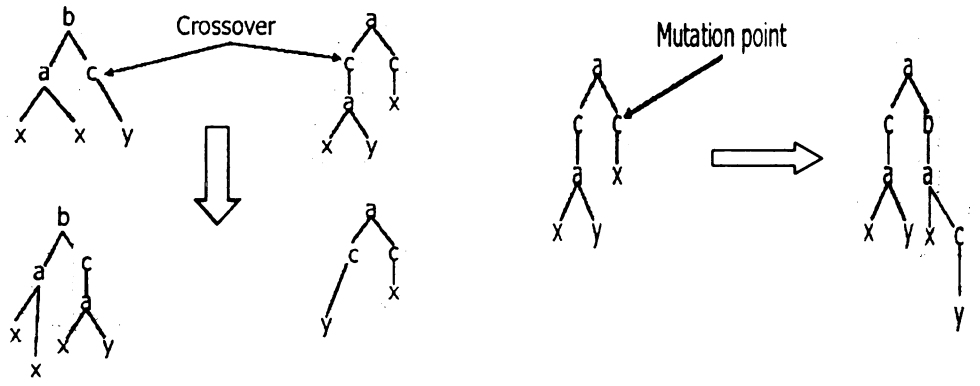


Figure 3.5 Illustration of crossover operator and mutation operator

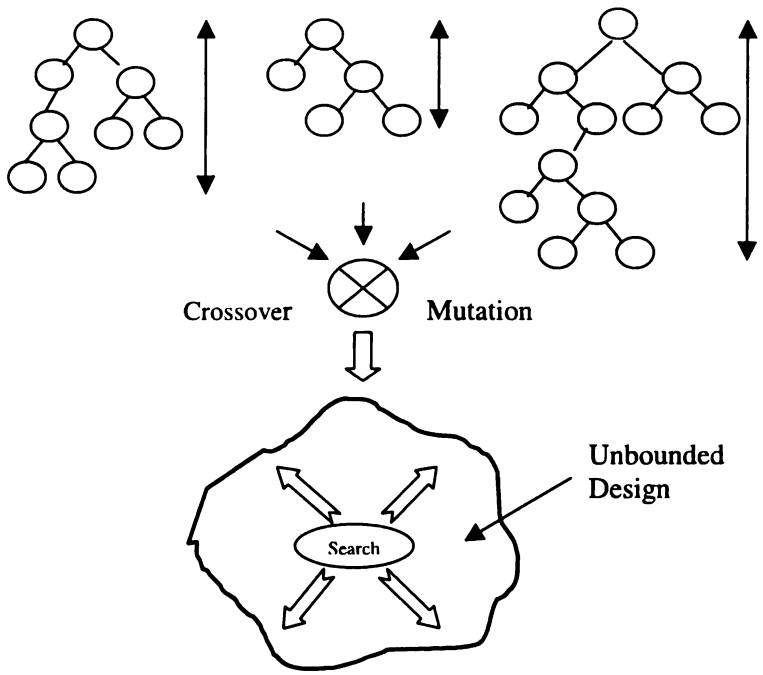


Figure 3.6 The extensible search capability of GP for an unbounded design space

Although crossover and mutation operators are both implemented in the genotype, namely the genetic programming tree, the result of executing the genotype generates the phenotype, a bond graph representation of a design. As the tree depths of genetic programming trees are not fixed and theoretically not limited, the possibilities of the shapes and parameters of resulting bond graph models (after the genotype-to-phenotype mapping) are actually unbounded. In this way, the combined capabilities of genetic programming to do efficient search in topologically unbounded space and bond graphs to model and represent mixed-domain dynamic systems lead to a powerful design synthesis approach for general open-ended multiport dynamic systems.

3.1.4 Fitness Evaluation

Fitness evaluation involves defining an objective or fitness function against which each individual is tested for suitability for matching the design specifications under various design constraints. As the algorithm proceeds, we would expect the individual fitness of the "best" individual, or design candidate in the particular case of our research, to increase, as should the total fitness of the population as a whole. An actual definition of fitness function is quite dependent on problem domain. Each application may have a different definition of the fitness function. More importantly, as design is the art of making products for a changing world, and the creation of new products is an ever-adapting and interactive process of integrating new information, new technologies and new biases from the marketplace, the fitness function may therefore be adaptive itself, enabling it to reflect changing design environments or preferences.

3.1.5 Selection

We need to select individuals from the current population for reproduction, or in other words, to create another population of design candidates in the next generation. By comparing the population of design alternatives, the best ones are selected to propagate to the next iteration while the remaining ones are discarded to make room for new solutions If we have a population of size 2N, the selection procedure picks out two parent individuals, based on their fitness values, which are then used by the crossover and mutation operators to produce two offspring for the new population. This selection/crossover/mutation cycle is generally repeated until the new population contains 2N individuals. A rule of thumb for selection is, the higher the fitness value, the higher the probability of that individual being selected for reproduction. This principle of selection pressure is called "survival of the fittest," which is the primary motivating factor for finding successful designs.

3.1.6 Premature Convergence

Premature convergence is often a tough problem to be addressed by practitioners of evolutionary computation. There is no guarantee that, for an arbitrary function to be optimized, approaches using finite populations and search times, based on evolutionary computation (EC), will always find a globally optimal solution. In fact, in practice, they often do not. Premature convergence is one underlying reason for this phenomenon. The EC-based approach may cease to search effectively for better solutions because all individuals in the population converge to one region of the search space – offspring tend to be only minor modifications of their parents. In the case of genetic programming, if the population is converged, simple tuning of parameters or adjusting of ad-hoc operators is not sufficient to make much difference, so few new individuals out of crossover and mutation operations will survive even if mutation rates are increased. As a result, the whole population tends to get stuck in one place and the evolutionary computations are not able to do further search in the search space. Many approaches have been proposed to combat the problem of premature convergence to sustain a continuing search pressure for better solutions.

A Hierarchical Fair Competition (HFC) model is developed and is the major topic of another dissertation in our group. It suggests a building block assembly line structure for continuing evolutionary machines. In this model, individuals are organized into different levels according to their fitnesses. Random individuals are continuously incorporated into the lowest fitness level, while new individuals at any level with fitnesses higher than others in that level progressively move out to higher levels. This kind of hierarchical organization of individuals allow new individuals with promising new building blocks to "grow up" gradually, without the severe competition from highly developed individuals.

The hierarchy of fitness serves as a repository for different levels of implicit building blocks. As this is the major part of another parallel research effort, it is not elaborated on further here, but is used throughout the experimental runs. Interested readers may refer to (Hu, et al. [2002]).

3.2 Overall Design Procedure

Now it is time to summarize our overall design procedure. As with any fairly general system for design automation, the user must, as part of the specification of the problem to be solved, indicate the target performance that is desired and how it is to be evaluated. That generally includes identifying some input variable(s) or driver(s) and some output(s) which are used to observe the behaviors. The desired behaviors must be specified. For a system to be represented as a bond graph, this amounts to specifying an "embryonic" physical model for the target system, which will remain invariant during the design process. That embryo should include any exogenous inputs, usually specified as sources of effort or flow (e.g., voltages, currents, forces, velocities, pressures, etc.). It must include any outputs required to evaluate fitness (for example, voltages across a given load resistance or flow rates through pipes). That these components should NOT be allowed to be changed/eliminated during design evolution is obvious – the problem is not defined without their presence. When the user has formulated the problem (i.e., the external boundaries of the physical model with its environment and the performance measures to be used), the user must specify it as an embryonic bond graph model and a "fitness" function. The user also specifies one or more "sites" in the embryo model where modifications/insertions are allowed. Then an initial population of GP trees is created at random, using that embryo as a common starting point. For each GP tree ("individual"), the bond graph is generated and analyzed. This analysis, including both causal analysis and (under certain conditions) state equation analysis, results in assignment of fitness to

the individual. Then genetic operations – selection, crossover and mutation – are performed on the evaluated population in the GP tree domain, generating new individuals (designs) to be evaluated. The loop, including bond graph analysis and GP operation, is iterated until the termination condition is satisfied. The result is one or more "best" bond graphs that satisfy predefined specifications and ready for physical realization. There is, of course, no basis for asserting the global optimality of any solution that arises – it is simply the best generated, and the procedure is considered successful if the quality of that design is adequate for the designer's purposes.

It is also important to point out that it is possible to get an idea of the design domain from "good" design candidates, not just "the best". For example, the designer may notice that a group of "good " design candidates share commonality of design topology and most component parameters. The only difference among those design candidates is the sizing for one particular component (for example, a C component). Then the designer can get a piece of heuristic knowledge that this C component may be very vital to the optimization of the design, and can focus on choosing a "best" parameter for this C component to further optimize the whole design.

The flow of the complete algorithm described above is shown in Figure 3.7. This loop of bond graph analysis and GP operation is iterated until a termination condition is satisfied. The final step in instantiating a physical design would be to realize the highest-fitness bond graph in physical components. We are going to illustrate this design procedure in several case studies in our research.

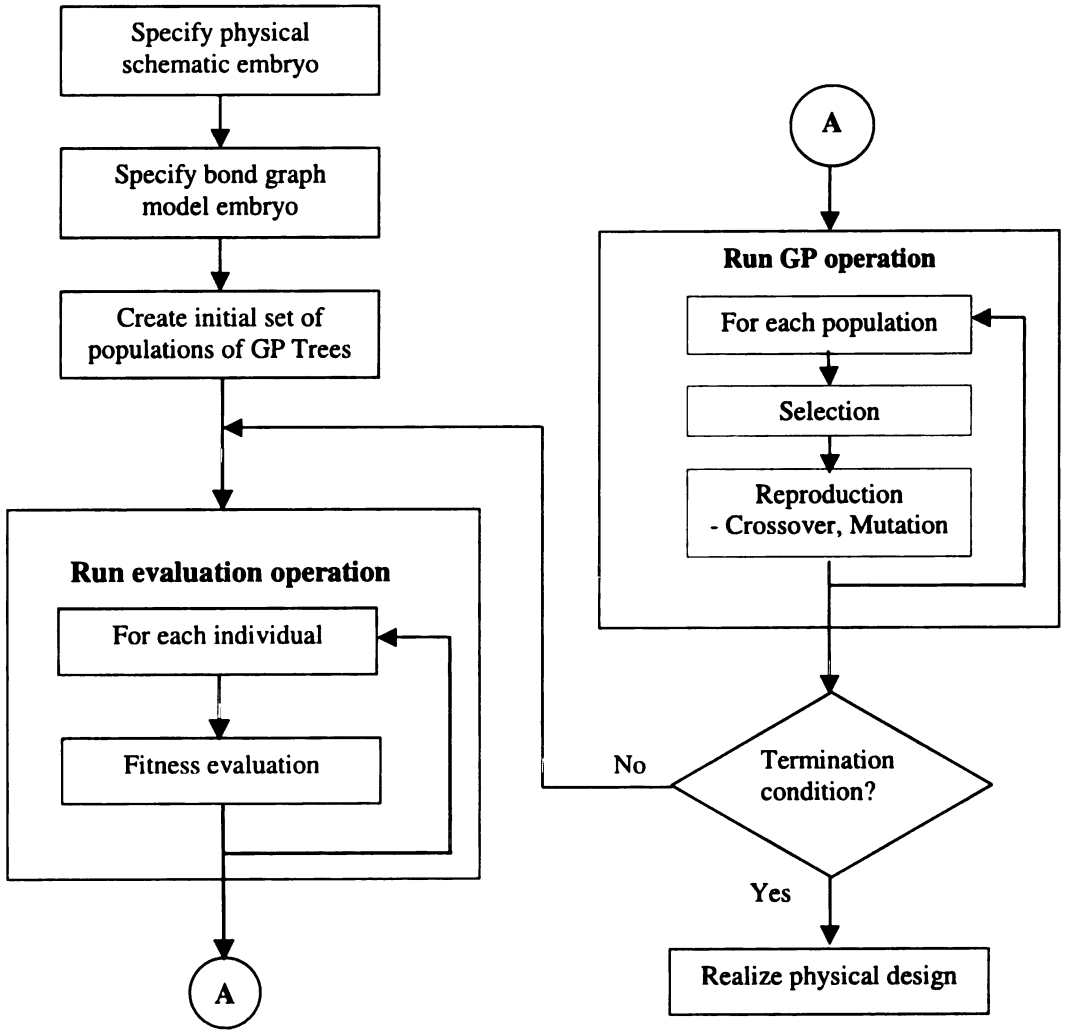


Figure 3.7 The overall design procedure of BG/GP approach

CHAPTER IV
CASE STUDIES OF BG/GP APPROACH

To test the ability of BG/GP approach for topologically open-ended design automation for mixed-domain dynamic systems, we choose two design problems mainly belonging to two different physical domains. They are 1). A passive analog filter design problem that belongs to the electrical domain, and 2) a printer design problem that mainly belongs to the mechanical domain.

4.1 Analog Filter Design Problem

Automatic synthesis of analog circuits is of great significance for electronic systems design, which involves the determination of the topology of circuits and sizing/parameterizing of their components. Many techniques have been used for such problems. Some methods incorporate heuristics; some predefine the topology, and then let the automated procedure optimize the parameters of the circuits. Some divide the design into two stages -- topology optimization via a GA and parameter optimization with numerical optimization methods (Grimbleby, [1995]). Some genetic algorithm approaches also evolve both topology and component parameters; however, they typically allow only a limited amount of components to be evolved (Lohn, [1999]). Using netlists as the representation technique for the circuit, and genetic programming as the evolutionary tool, Koza has developed very successful approaches to deal with circuit synthesis problems, evolving topologies and parameters simultaneously (Koza, [1999]). However, those applications are currently confined to the electrical domain, and exhibit very heavy demand for computing resources.

4.1.1 Bond Graph Representation of Circuits

In the context of circuit design, a bond graph consists of the following types of elements:

- C, I, and R elements, which are passive one-port elements that contain no sources of power, and represent capacitors, inductors, and resistors.

- Power source elements including S_e and S_f, which are active one-port elements representing sources of voltage or current, respectively. In addition, when the current of a current source is fixed as zero, it can serve as an ideal voltage gauge. Similarly, when the voltage of a voltage source is fixed as zero, it can serve as an ideal current gauge

- Transformer (TF) and gyrator (GY), which are two-port elements, and represent transformers and gyrators, respectively. Power is conserved in these elements.

- 0-junctions and 1-junctions, which are multi-port elements for representing series and parallel relationships among elements. They serve to interconnect elements into subsystem or system models

- Bonds, which are used to connect any two elements in the bond graph.

A unique characteristic of bond graphs is their use of 0- and 1-junctions to represent the series and parallel relationships among components in circuits. In fact, it is this concept that led to the foundation of the bond graph field (Paynter, [1991]). Junctions transform common circuits into a very clean structure with few loops, which can otherwise make circuits appear very complicated. Figure 4.1 shows the comparison of a circuit and a corresponding bond graph. The evaluation efficiency of the bond graph

model is further improved due to the fact that analysis of causal relationships and power flow between elements and subsystems can reveal certain system properties and inherent characteristics. This makes it possible for us to discard infeasible design candidates even before numerically evaluating them, thus reducing time of evaluation to a large degree. In addition, as virtually all of the circuit topologies created is valid, our system does not need to check validity conditions of individual circuits to avoid singular situations that could interrupt the running of a program evaluating them.

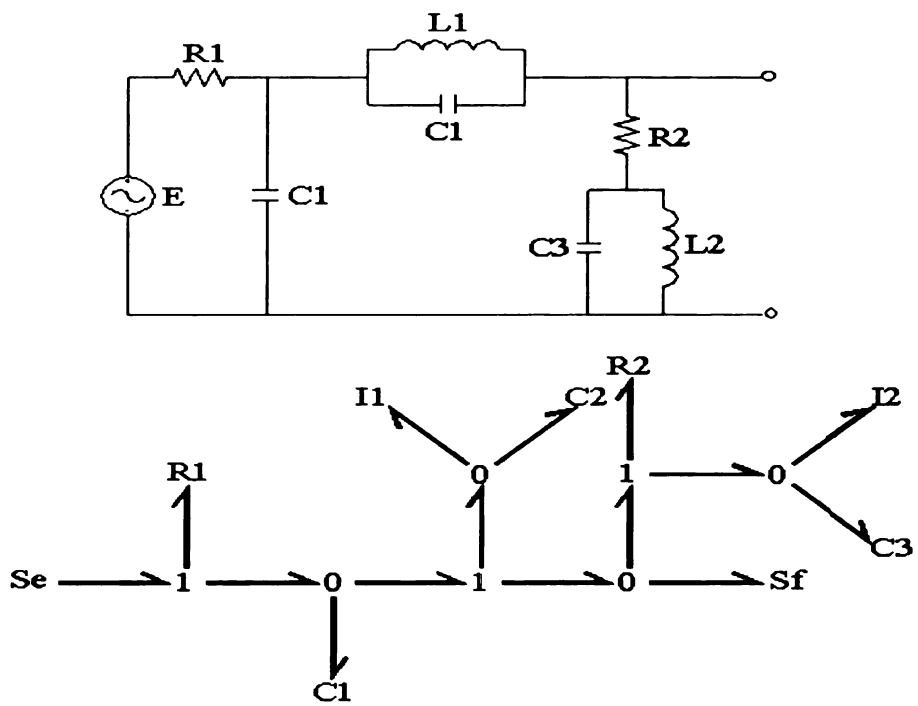


Figure 4.1 Bond graph representation of an electrical circuit

4.1.2 Problem Definition

Three kinds of filter designs were chosen to verify our approach - high-pass, low-pass, and band-pass filters. The embryo electric circuit and corresponding embryo bond graph model used in our filter design are shown in Figure 4.2. We used converted Matlab routines to evaluate frequency response of the filters created. As Matlab provides many powerful toolboxes for engineering computation and simulation, it facilitates development of source codes for our genetic programming evaluation dramatically. In addition, as all individual circuits passed to Matlab code for evaluation are causally valid, the occurrence of singularities is excluded, which enables the program to run continuously without interruption. 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; compute their differences and obtain the squared sum of differences as raw fitness. Then normalized fitness is calculated according to:

$$Fitness\ (\ Filter\ ) = 100 \Big/ (100\ + \sum Error\ )$$

The GP parameters used for eigenvalue design were as follows:

Number of generations: 100 Population size: 300 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

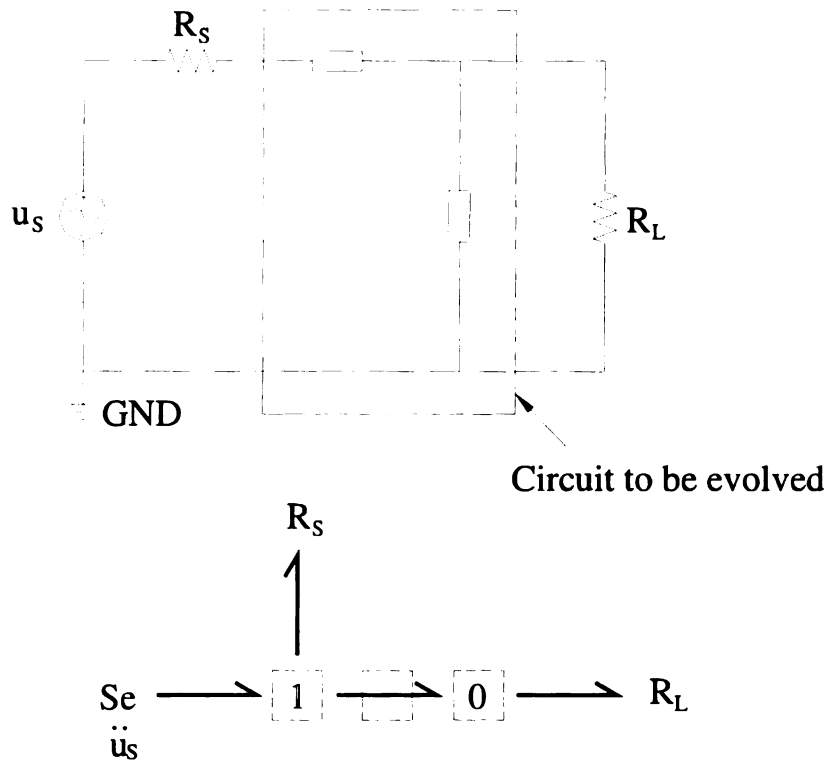


Figure 4.2 Embryo of electrical circuit and its bond graph model

4.1.3 Results

To illustrate an intermediate step in the evolution of a high-pass filter with a target cutoff frequency of 1000Hz, the performance of the best design evolved at generation 10 is shown in Figure 4.3. It is clear that this design is far inferior to that evolved by the end of the run (fewer than 100 generations), as shown in Figure 4.4. Figure 4.5 gives the frequency response of an evolved low-pass filter with the same cut-off frequency. It shows that this result is also quite satisfactory. Figure 4.6 gives the frequency response of an evolved band-pass filter with cutoff frequencies at 10Hz and 1000Hz. Obviously, it is the most difficult of the three filter design problems. The evolved high pass filter circuit and bond graph are shown in Figures 4.7 and 4.8.

The statistical results of 10 runs each for high-, low- and band-pass filters are shown in Table 4.1. The distance errors between ideal frequency output and the output obtained, together with fitness values, are summarized. With the exception of some of the band-pass results, most were quite acceptable. Figure 4.9 shows the fitness history of a typical high-pass filter run.

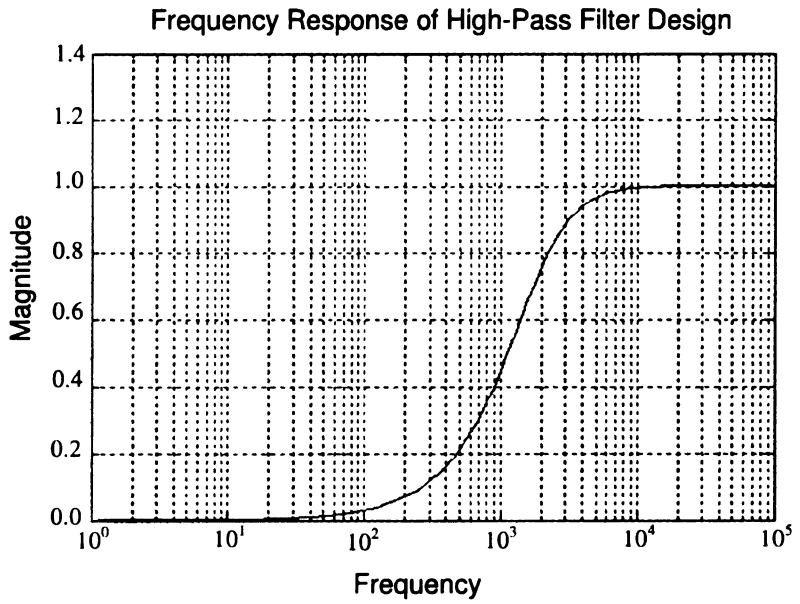


Figure 4.3 Frequency response of a high-pass filter design with fitness value of 0.917

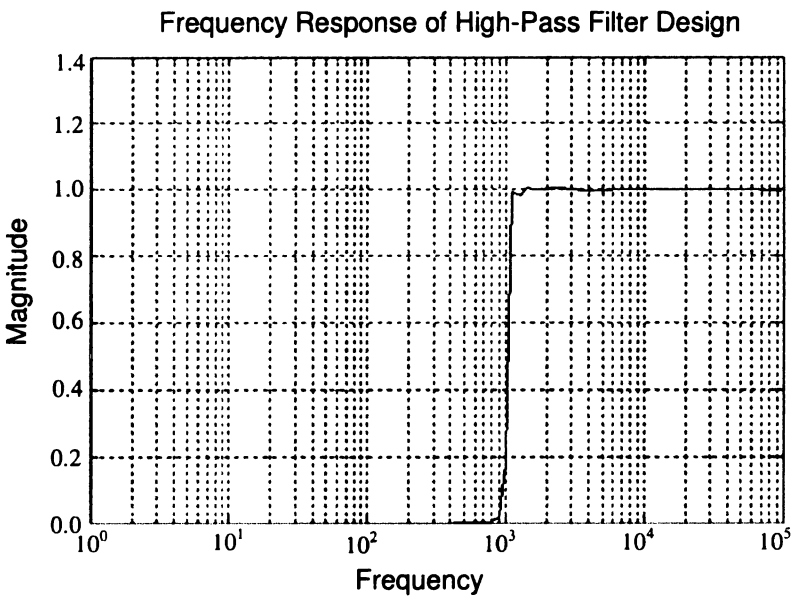


Figure 4.4 Frequency response of a high-pass filter design with fitness value of 0.992

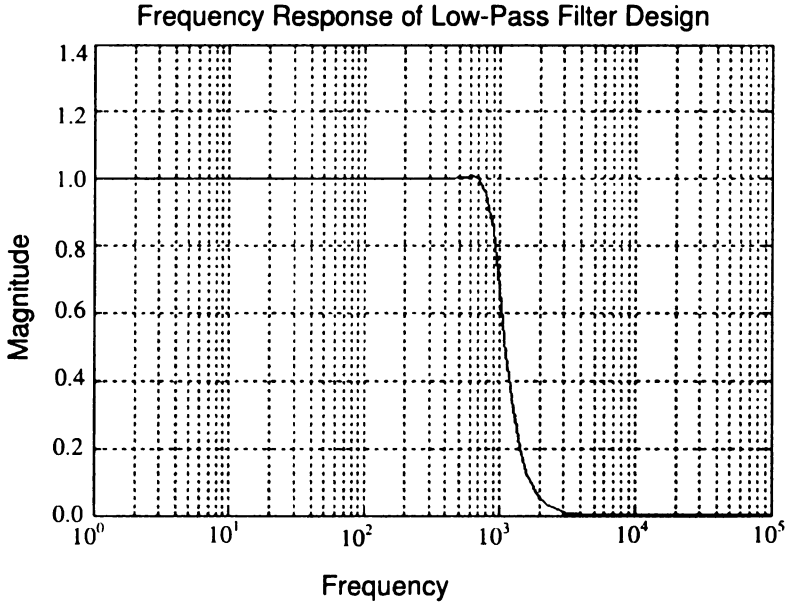


Figure 4.5 Frequency response of a low-pass filter design with fitness value of 0.980

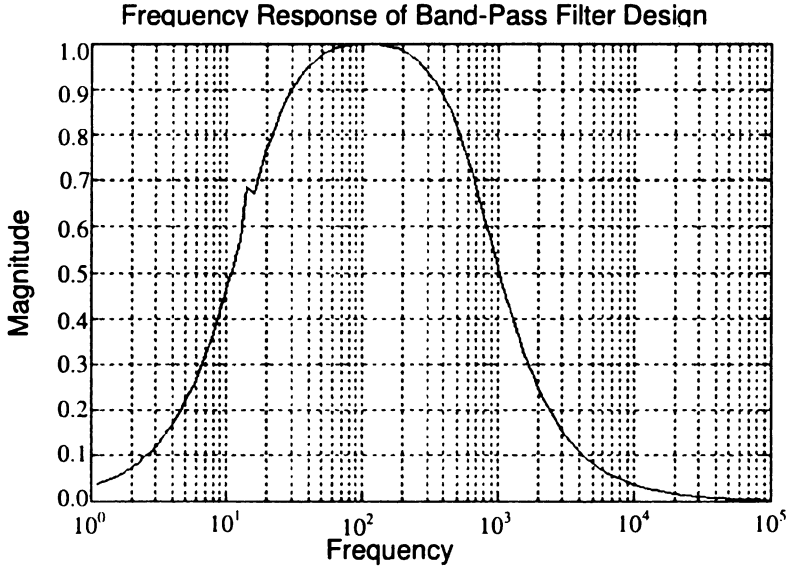


Figure 4.6 Frequency response of a band-pass filter design with fitness value of 0.884

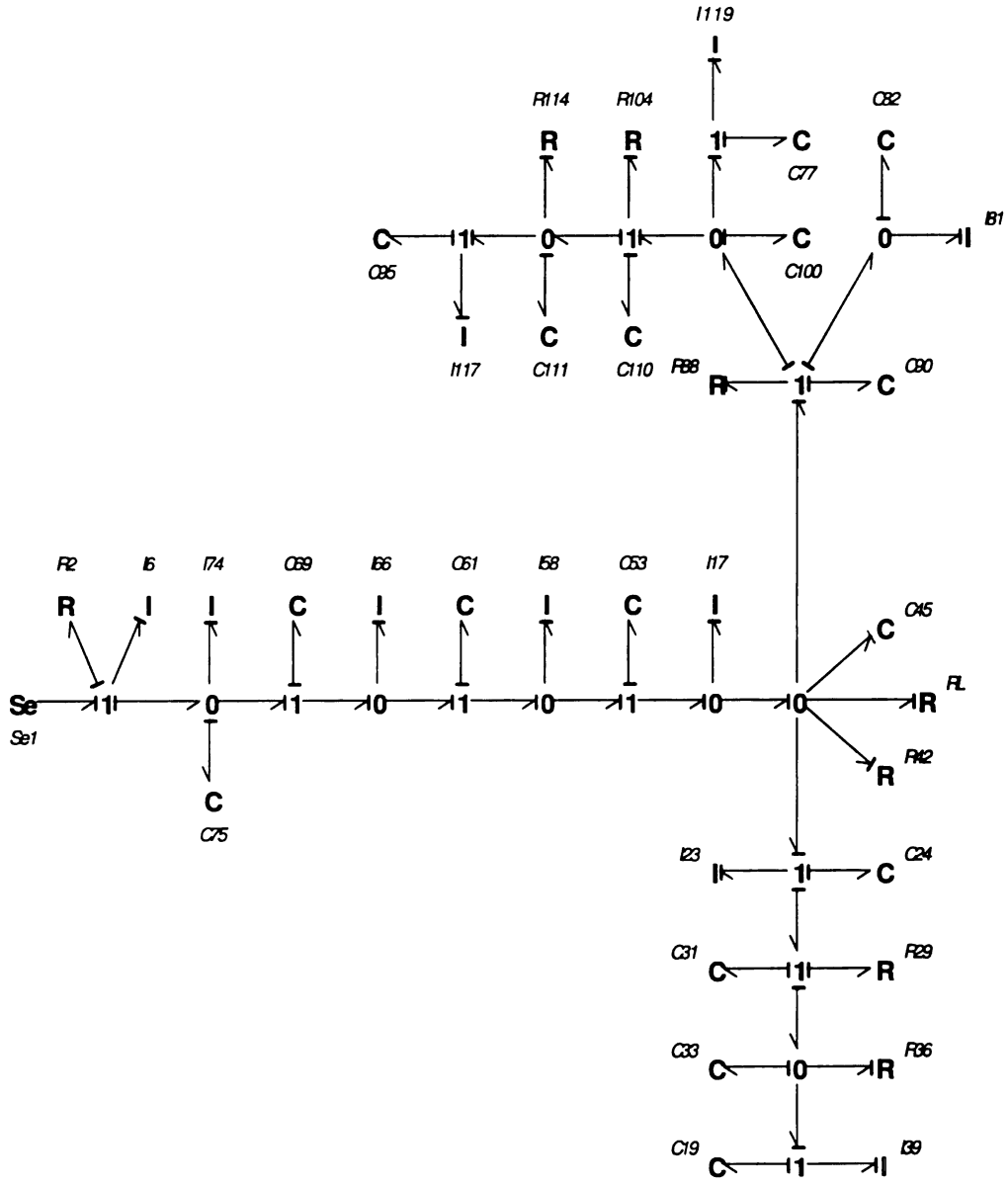


Figure 4.7 Evolved bond graphs model for high-pass filter

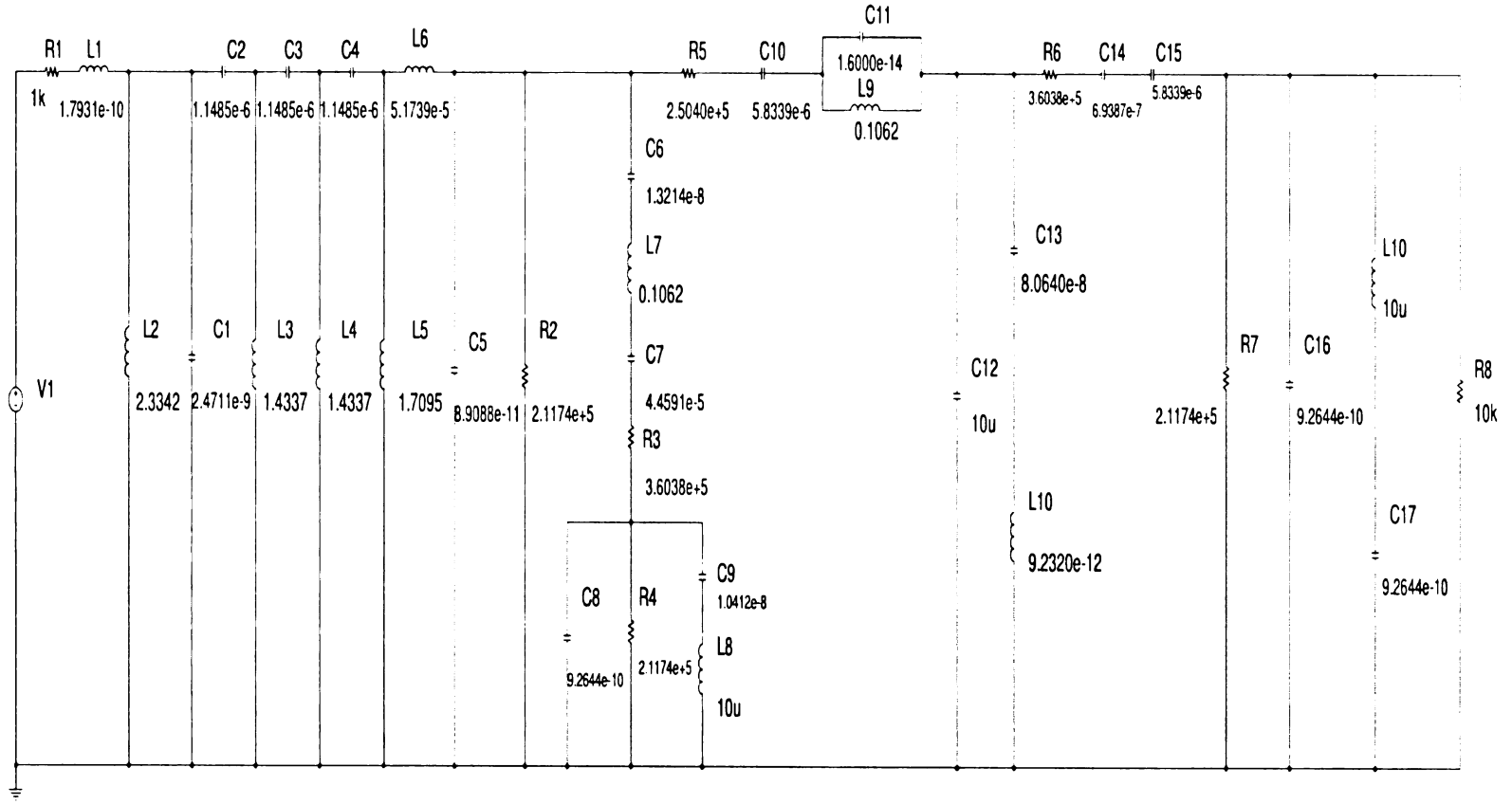


Figure 4.8 Evolved electrical circuit for high-pass filter design

Table 4.1 Summary results (errors, fitnesses) for filter designs

Run No.	Low-pass		High-pass		Band-pass	
	Error	Fitness	Error	Fitness	Error	Fitness
1	2.334	0.977	3.349	0.968	9.067	0.917
2	3.428	0.967	2.031	0.980	12.861	0.886
3	2.202	0.978	1.159	0.989	12.698	0.887
4	3.032	0.971	2.337	0.977	12.672	0.888
5	2.162	0.979	0.828	0.992	8.662	0.920
6	3.427	0.967	2.860	0.972	12.864	0.886
7	3.026	0.971	3.287	0.968	13.100	0.884
8	2.951	0.971	0.725	0.993	13.090	0.884
9	2.154	0.979	1.141	0.989	6.003	0.943
10	1.988	0.981	1.917	0.981	13.049	0.885
Best	1.988	0.981	0.725	0.993	6.003	0.943
Worst	3.427	0.967	3.349	0.968	13.100	0.884
Avg	2.670	0.974	1.963	0.981	11.407	0.898
S.D	0.530	0.005	0.936	0.009	2.541	0.021

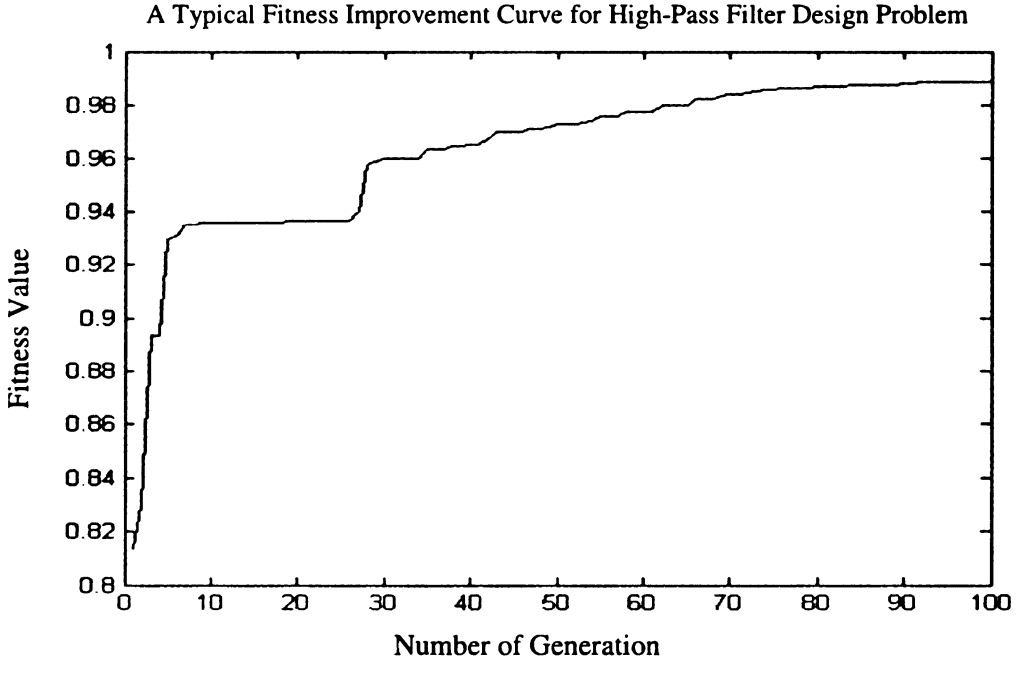


Figure 4.9 Fitness history for a typical high-pass filter run

4.2 Design of Vibration Absorber for Mechanical Printer

4.2.1 Problem Formulation

The original design problem was presented by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Figure 3 shows a closed-loop control system to position a rotational load (inertia) denoted as J_L. The system includes electric voltage source, motor and mechanical parts. Bond graphs are used for modeling the system (please refer to Figure 4.10 and Figure 4.11)

The problem with the design is the position output of the load J_L for a step input in voltage has intense vibrations (see figure 4.12). The design specification is to reduce the vibration of the load to an acceptable level, given certain command conditions for rotational position. We want the settling time to be less than 70ms when the input voltage is stepped from zero to one. Note that the settling time of the original system is about 2000ms. The time scale in Figure 4.12 is 4000 ms.

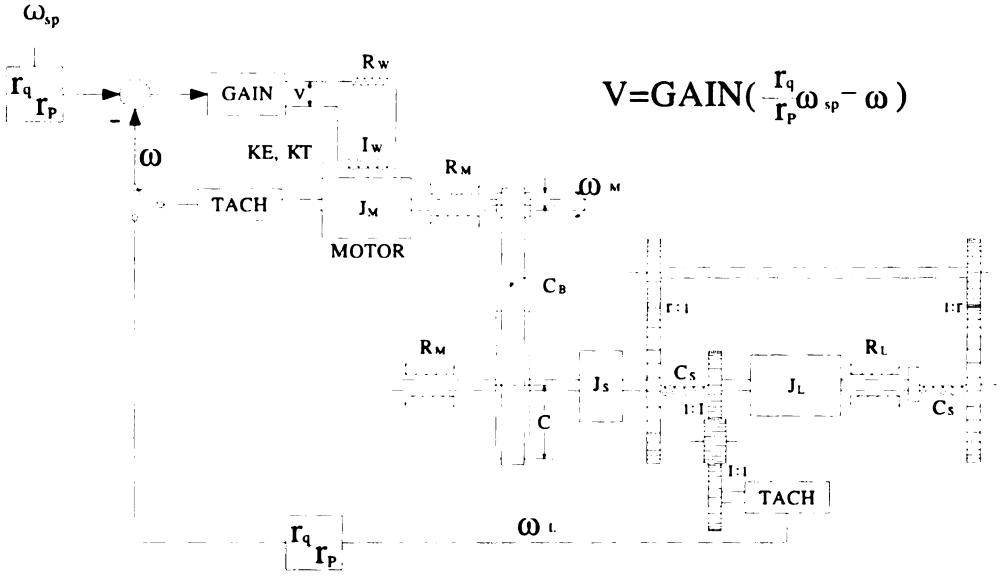


Figure 4.10 The schematic of the original printer system

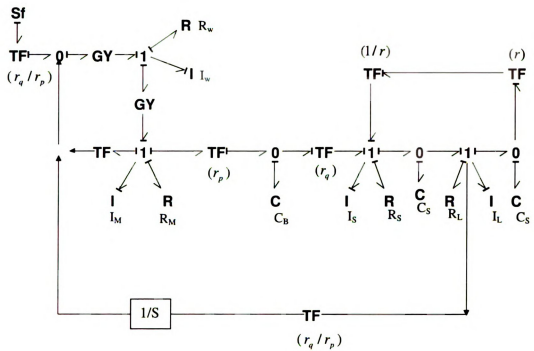


Figure 4.11 Bond graphs model for the original printer system

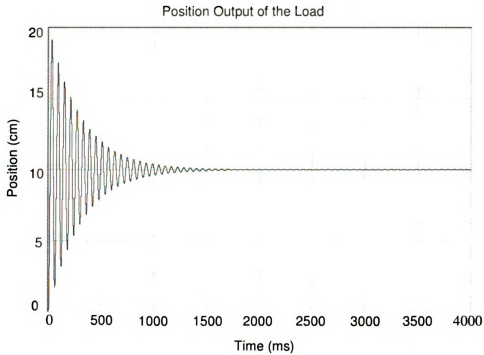


Figure 4.12 Simulation result of the original printer drive subsystem.

By analyzing the model, we conclude that the critical part for the design is a subsystem that involves the drive shaft and the load (figure 4.13). The input is the driving torque, T_d, generated through the belt coupling back to the motor.

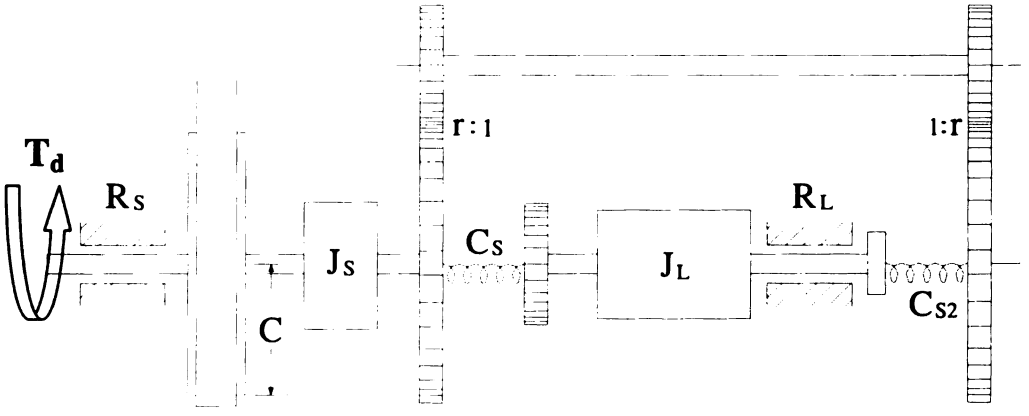


Figure 4.13 The critical printer drive subsystem

This subsystem was deemed a logical place to begin the design problem. The questions left to the designer now are: 1) at which exact spots of the subsystem new components should be inserted, 2) which types of components and how many of them should be inserted, in which manner, and 3) what should be the values of the parameters for the components to be added? The approach reported in this paper is able to answer these three questions in one stroke in an automated manner, once the embryo system has been defined.

4.2.2 Embryo of Design

To search for a new design using the BG/GP design tool, an embryo model 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. The embryo used for this example, expressed in bond graph language, is shown

in Figure 4.14, with the modifiable sites highlighted. The modifiable sites are places that new components can be added. The choice of modifiable sites is typically easy for the designer to decide. Note that modifiable sites are only possible spots for insertion of new components; the search may not use all of them. In this particular example, designers need have no idea whether assemblies of new components will be inserted at modifiable site (1), or at modifiable site (2), at site (3), or at any combinations of them. Instead, the algorithm will answer these questions in an automatic way, without intervention by the human designer.

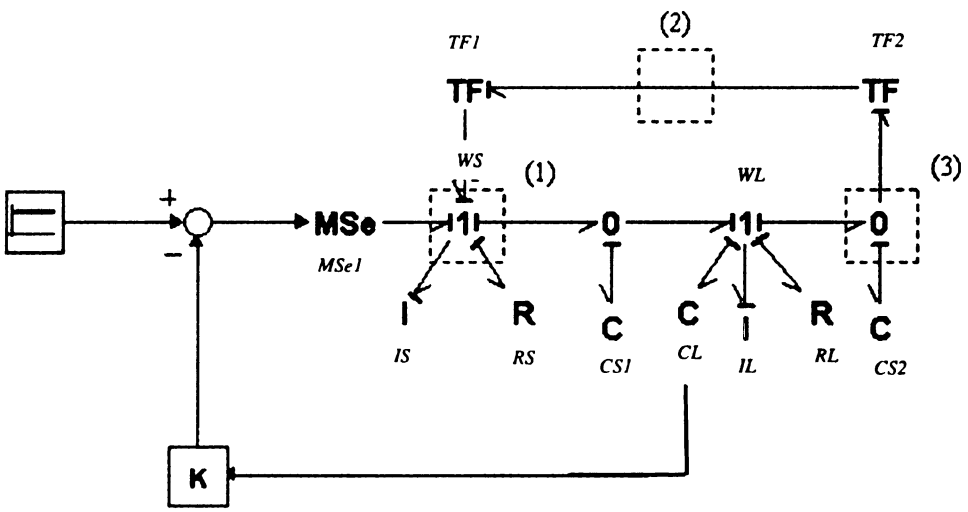


Figure 4.14 The design embryo of the printer subsystem

The parameters for the embryo model are:

I_s : $6.7\times10^{-6}\,kg\cdot m^2$

R_s : $0.013\times10^{-3}\,N\cdot m\cdot \sec / rad$

C_{s1}: $0.208\ \ N\cdot m\cdot / rad$

C_{s2}: $0.208\ \ N\cdot m\cdot / rad$

R_L : $0.58\times10^{-3}\,N\cdot m\cdot \sec / rad$

I_L : $84.3\times10^{-6}\,kg\cdot m^2$

C_L : $1.0\times10^{6}\,N\cdot m\cdot / rad$

$TF1$: 0.1, $TF2$: 10

For simplicity and without loss of generality, both K and MSe gain are set to be unit.

A notable difference exists between this design embryo and that of the filter design problem as discussed in the last session. While the embryo for the filter design was quite simple, the embryo for the printer redesign is much more complex. This is because in the printer redesign problem, most parts of the printer system are fixed. The designer only wants to insert or reconfigure components at a few positions in the original system, in an effort to form a mechanical vibration absorber subsystem. This difference of embryos manifests the major difference of solving design and redesign problem using BG/GP approach. In a design problem, the approach should generate and evolve a design from scratch, so the embryo is left to be simple and trivial. While for the redesign problem, the major part of the system is required to be intact. The modifiable part of the system, on the other hand, becomes relatively minor part of the whole system. As a result, in a redesign problem, we are more apt to see a nontrivial embryo for the design, which means we are going to spend more time in analyzing and defining a suitable embryo in a redesign problem before we start a genetic programming run.

The following cases were run on a single Pentium III 1GHz PC with 256MB RAM. The GP parameters were as shown below.

Number of generations: 100 Population sizes: 200 in each of 15 subpopulations Initial population: half_and_half Initial depth: 3-6 Max depth: 17 Selection: Tournament (size=7) Crossover: 0.9 Mutation: 0.1

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 5~15 minutes, which shows the efficiency of our approach in finding good design candidates

4.2.3 Results

Ten runs of this problem have been done and most of the runs produced very good solutions. The fitness history of a typical run is shown in Figure 4.15. Two competing design candidates with different topologies, as well as their performances, are provided in Figure 4.16 to Figure 4.21 (evolved components are circled). We can see from the output rotational position responses that they all satisfy the design specification of settling time less than 70ms. Note that the time scale of the plots is 100 ms.

One of the designs is shown in Figure 4.16. It is generated in only 20 generations with 200 designs in each of 15 subpopulations, and has a very simple structure. Three elements, one each of 0-junction, C, and R, are added to modifiable site 1 of the embryo model (Figure 4.16). The performance of this model is shown in Figure 4.18. The position response for step function input quickly converges in about 50msec, which was an acceptable timeframe. Physical realization of the bond graphs model is shown in Figure 4.17. A spring and a damper are added and coupled to the original printer subsystem as shown in Figure 4.13.

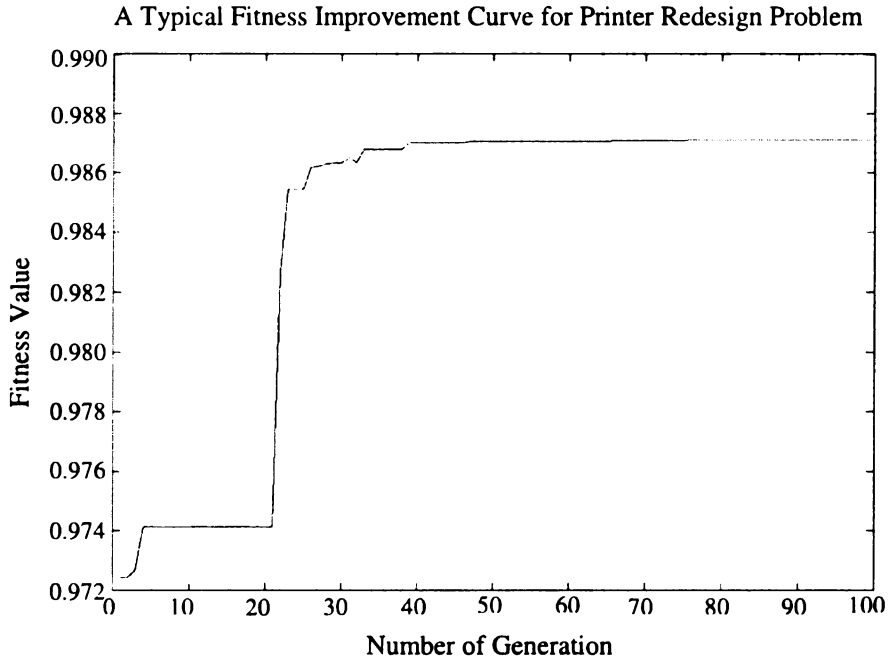


Figure 4.15 Fitness history for a typical printer drive redesign run

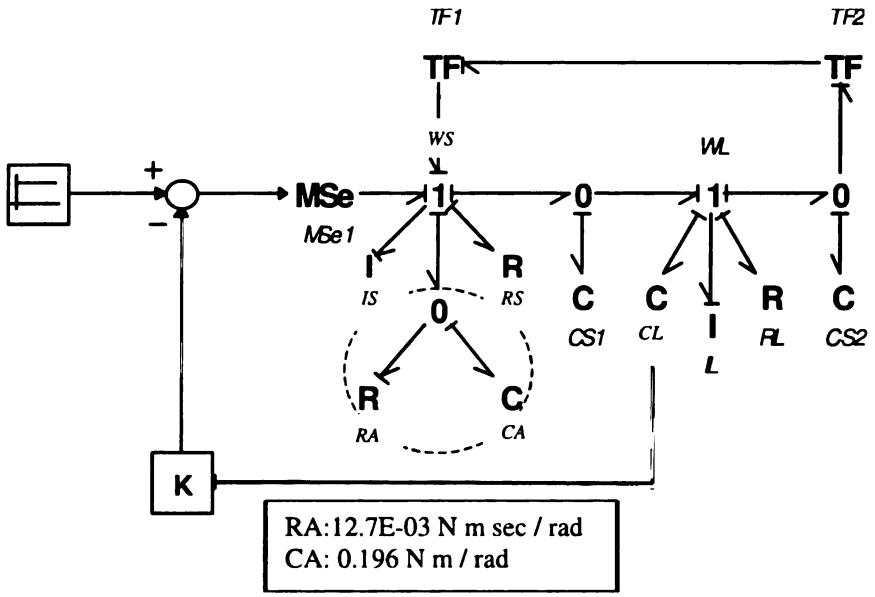


Figure 4.16 The evolved bond graph model I

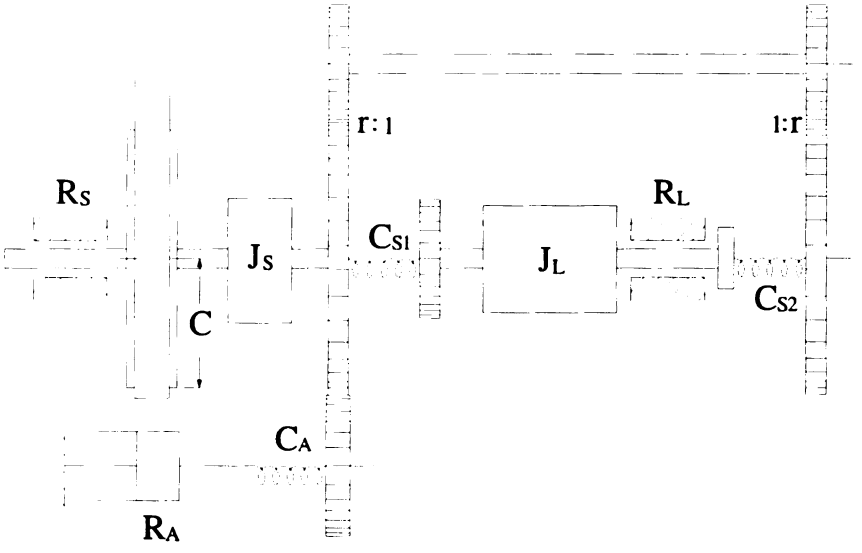


Figure 4.17 The physical realization of evolved bond graph model I

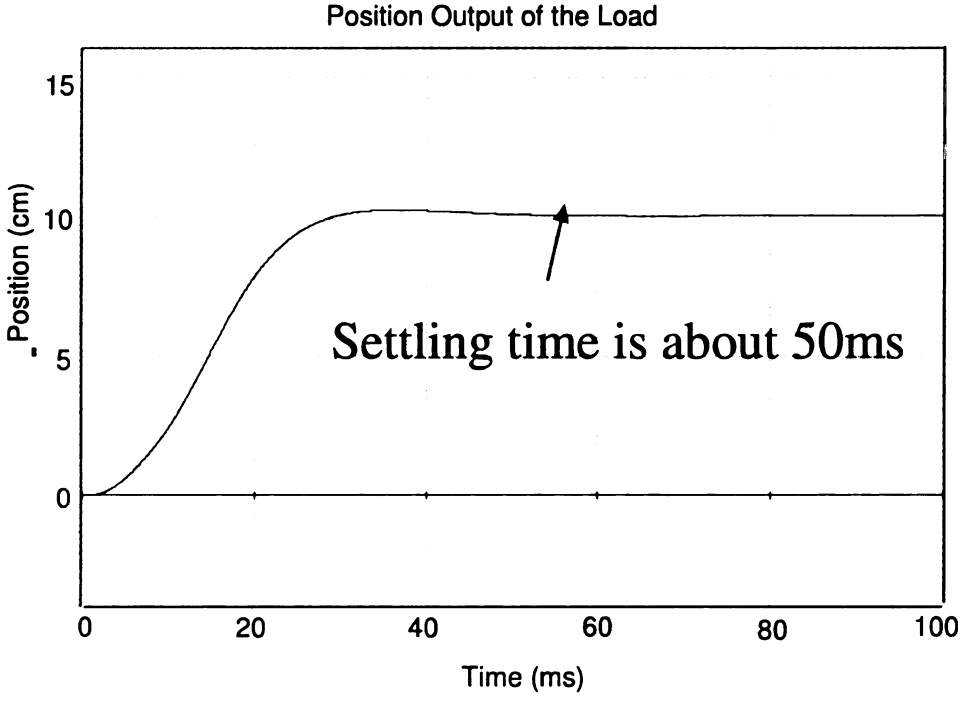


Figure 4.18 Simulation result of evolved bond graph model I

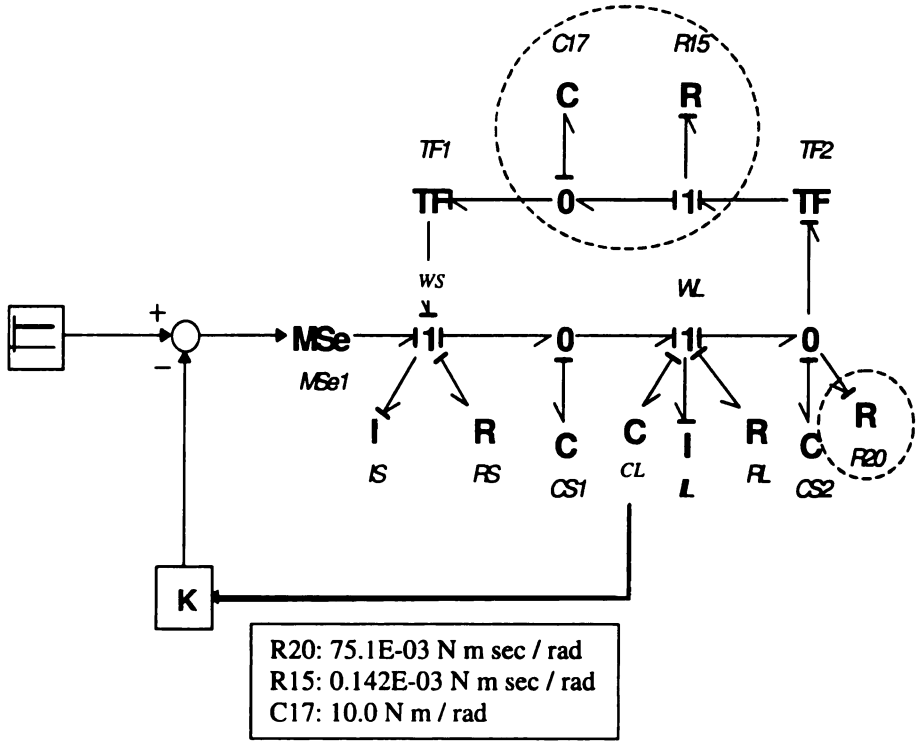


Figure 4.19 The evolved bond graph model II

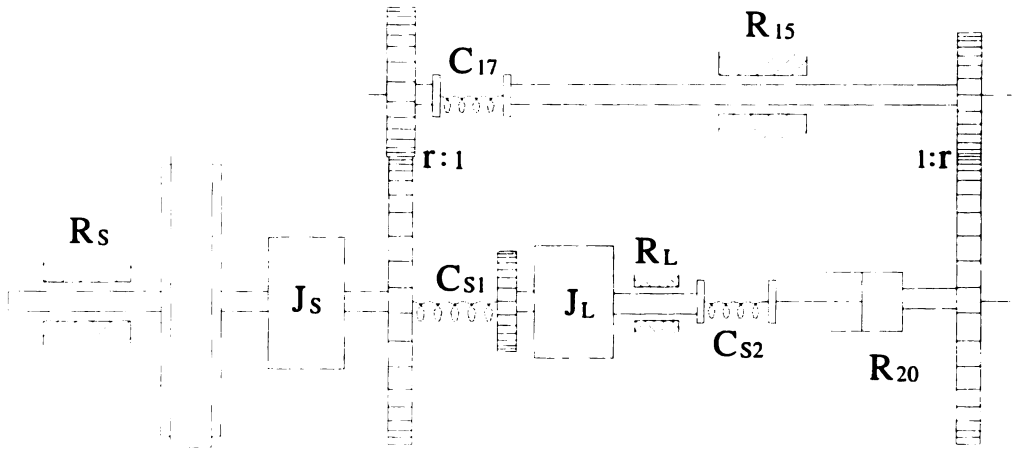


Figure 4.20 The physical realization of evolved bond graph model II

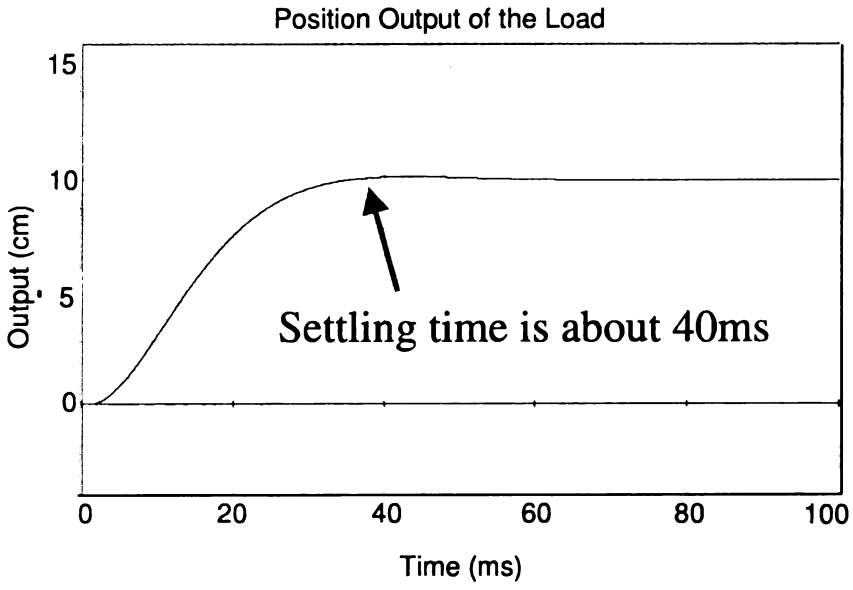


Figure 4.21 Simulation result of evolved bond graph model II

Table 4.2 Summary results of fitness for printer

Run No.	Fitness of Printer	
	Distance	Fitness
1	15.076	0.985
2	15.818	0.984
3	15.188	0.985
4	16.720	0.983
5	15.053	0.985
6	14.085	0.986
7	15.122	0.985
8	15.502	0.985
9	15.132	0.985
10	15.881	0.984
Best	14.085	0.986
Worst	16.720	0.984
Avg	15.358	0.985
S.D	0.6903	0.000669

Another design is shown in Figure 4.19. Four elements, 0-junction with C, 1-junction with R are added to modifiable site 2 and one R is added to modifiable site 3. The physical realization of the design is shown in Figure 4.20. Figure 4.21 displays the performance of this model.

Table 4.2 represents the statistical results of 10 runs for the printer drive.

4.3 Discussion

Two design examples show the feasibility of the proposed BG/GP approach in various aspects. First, the two design examples belong to different physical domains. Filter design problem is the design of an electrical system, while printer redesign problem is basically a design problem for a mechanical vibration absorber. This fact simply demonstrates the mixed-domain design capability of BG/GP approach. Second, the result of the passive high-pass analog filter design demonstrates both effectiveness and efficiency of our approach combining bond graphs and genetic programming. It shows that the approach is capable of evolving very satisfactory results in a moderate period of time on a single personal computer. To get the results shown in section 4.1, a typical program ran in a P-III 1GHz for 44.8 minutes. It took the genetic programming algorithm 100 generations to evolve it. This result is considered to be acquired in an efficient manner because for an evolutionary computation algorithm to evolve designs with similar complexity, it usually takes a much longer time and consumes much more computational resources, such as clusters of computers (Koza et al. [1997]). No one single factor stands out as the sole reason for this efficiency -- we believe several factors contribute. First, the bond graph representation of dynamic systems has strong topological expression

capability. Second, the genetic operators used promote efficient generation and reconfiguration of bond graph topologies. Third, causality analysis of the bond graph model before evaluating design candidates in detail helps to discard a large volume of improper designs without requiring full evaluations, thus reducing computation time and resources. In summary, the printer redesign problem demonstrates the strong topological exploration ability of BG/GP approach. In a very short period of time, BG/GP approach successfully identified a variety of design candidates satisfying design specifications for further analysis and tradeoff by design engineers.

CHAPTER V
EVOLUTIONARY SYNTHESIS OF MEMS

Even though the successful case studies discussed in the previous chapter show that the BG/GP approach can be a useful tool for dynamic systems design, one is still driven to ask, "Why is mechanical systems design not more like VLSI design?" As is well known, Electronic Design Automation (EDA) has achieved tremendous success in both industry and academia. However, similar success has not been achieved in design automation of mechanical systems. One fundamental reason for this is that mechanical systems lack highly modularized components that have clearly specified interfaces among each other, as VLSI components do. Fortunately, mechatronic systems, which are increasingly replacing conventional mechanical systems, can transfer energy and information flows among their components through electric wires, thus can be modularized far more than mechanical systems. This feature makes mechatronic systems generally more amenable to design automation approaches and it is expected that next generation mechatronic systems will become increasingly modularized. Accordingly, Mechatronic Design Automation (MDA), as an emerging research area, holds great promise. In particular, Micro-Electro-Mechanical-Systems (MEMS), actually micro-mechatronic systems, might be the first type of mechatronic systems to achieve success comparable to that already attained by EDA, due to its close affinity with VLSI. MEMS actually evolved from microelectronics and inherited many fabrication techniques of VLSI.

This chapter starts with an analysis of both the challenges and promises of MEMS design and synthesis. A structured design automation method is strongly recommended, by which the design process is deliberately divided into several levels. Each level has its own design focus and objectives, as well as its own design automation and optimization

approaches. After following a top-down design process, a bottom-up verification process is also carried out to verify that at each level the design specifications are exactly satisfied. The BG/GP approach discussed in the previous chapters is very suitable to be extended and applied to the first level, or system-level design for MEMS. The feasibility of the extended BG/GP approach is demonstrated through an example of MEMS design in a particular domain of RF MEM devices, namely, micromechanical bandpass filter design level. Then at the second level, the physical layout synthesis problem is formulated as a constrained optimization problem and treated with a special type of constrained genetic algorithm presented by Deb, [2000]. Finally, some implementation considerations to extend the approach across various design levels are also identified and discussed.

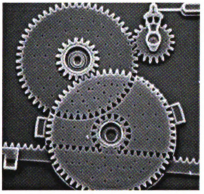
5.1 Introduction to MEMS Design and Synthesis

Simply put, MEMS are electromechanical systems built on a very tiny length scale. Figure 4.1 shows two typical MEMS. The left one shows a gear-mechanism with a length scale of millimeters, while the right one shows a combination of parallel comb-driven resonators with a length scale of micrometers.

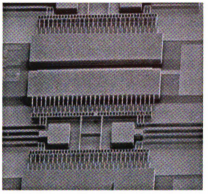
The comb driven resonators, which have a length scale of micrometers, can hardly be seen clearly by the naked eye. Design of systems on such a tiny scale is very difficult. The following is a paragraph quoted from Professor G. K. Fedder, a pioneer and specialist in MEMS design and synthesis.

"No rapid design process is available today for MEMS... this is very expensive... Full verification of designs requires months of effort, and design optimization is not realistic in all but the simplest of cases."

–G.K. Fedder et al., 1999



A geared mechanism of MEMS
Length scale is millimeters



Comb Driven Resonators
Length Scale is micrometers

Figure 5.1 Examples of MEMS

5.2 Promises and Challenges of MEMS Design and Synthesis

Some people may be surprised that MEMS design and synthesis is so difficult. Their argument is that MEMS evolved from microelectronics, so should have similar design tools available. A strong relationship between Very Large Scale Integrated circuits (VLSI) and MEMS does exist. Actually, MEMS has borrowed or inherited the fabrication process of VLSI. As is known, VLSI has such successful and highly structured "toolkits" for design automation that the whole new industry of Electronic Design Automation (EDA) has been created based on them. It seems that a similar design automation approach for MEMS should be very promising.

However, one major difference between VLSI and MEMS makes design of MEMS much more difficult. MEMS are intrinsically a hybrid system with both electrical parts and mechanical parts, while VLSI is basically a single-energy-domain system comprised of only electronic or electrical components. The mechanical subsystems of MEMS give rise to many difficulties and design problems. For example, the interface between an

electrical subsystem and a mechanical subsystem is still not well studied and definitely needs further investigation, because a large portion of design and fabrication problems arise in the interface zone where signal and energy transitions across physical domains occur very frequently. Another example of a difficulty is that the mechanical subsystem often includes moving parts, like vibrating beams or shifting combs. These moving parts are usually more fragile than fixed parts under external pressure loads or environmental changes (e.g. temperature changes). Design of these moving parts requires considerations not required of electronic parts, and is more complicated.

Due to the complexity and intricacy involved in MEMS design, designing MEMS still remains an art in most applications, requiring a large investment of human resources, time and money. Much of the investment is consumed in the iterative trial-and-error design process. As a result, we have only seen a handful of successful commercial MEMS products – those that the market has demanded in large quantities, including automotive accelerometers and gyroscopes, pressure sensors, ink-jet print heads and a few others. Prevalence of design and fabrication of MEMS application-specific integrated circuits (ASICs) analogous to electronic ASICs is still not seen.

Despite the numerous difficulties presented in automated synthesis of macro-mechanical systems, MEMS holds the promise of being amenable to structured automated design due to its similarities with VLSI, provided that the synthesis is carried out in a properly constrained design domain. However, it turns out that translating the key insights of the successful silicon evolution into MEMS technologies is a much more challenging task than most people had 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) abstracting design hierarchies to stratify and conquer design complexity, thus making the design more amenable to an automated process,
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, and
4) combining MEMS component layout extraction and lumped-parameter bond graph (or other multi-domain) simulation and design synthesis to provide MEMS designers with VLSI-like environments enabling faster design cycles and improved design productivity.

This chapter seeks to partially address the above challenges, especially the first two. The proposed hierarchical and evolutionary design framework for MEMS aims to eliminate tedious and repetitive design tasks, facilitate hierarchical problem decomposition, and combine the power of multiple evolutionary computation algorithms working simultaneously to identify better product designs and process solutions. In particular, we divide design representations of MEMS design 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 satisfy more detailed design constraints

5.3 Hierarchical MEMS Design Methodology

In MEMS, there are a number of levels of designs that need to be synthesized (Fedder and Jing [1999]). 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

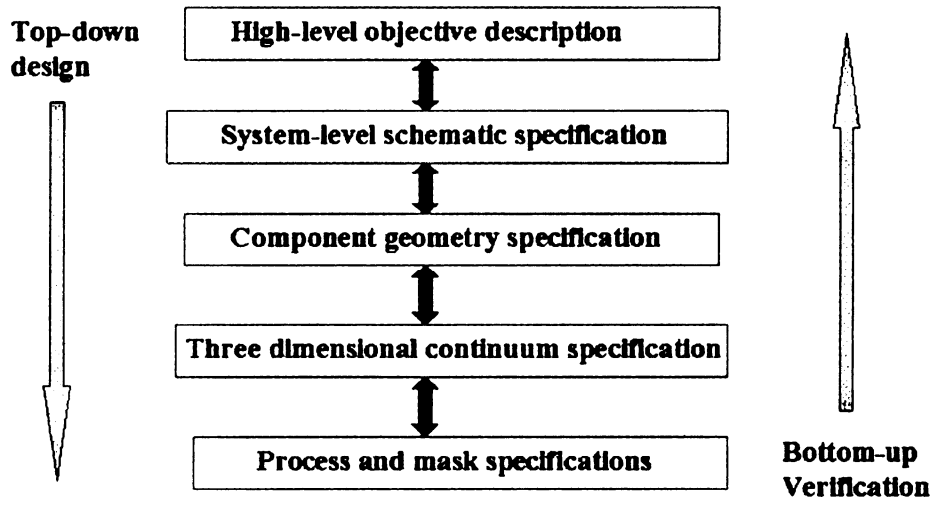


Figure 5.2 Hierarchical Design of MEMS

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 at this level will end one cycle of design. More generally, modeling and analysis of a 3-D solid model for MEMS is necessary. 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 fabrication procedures. Automated mask layout and process synthesis tools would be very helpful to relieve designers from considering the fabrication details and allow them to focus on the functional design of the device and system (Ma and Antonsson [2000]). After a "top-down" design path, a

"bottom-up" verification process usually follows to guarantee that at each design level the design specifications are met exactly as defined (Fig. 5.2). 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.

5.4 System-Level Synthesis of MEMS

For system-level design, hand calculation is still the most popular method in current design practice. This is largely for the following reasons: 1) The MEMS systems we are considering, or designing, are relatively simple in dynamic behavior -- especially the mechanical parts -- largely due to limitations in fabrication capability. 2) There is no powerful and widely accepted synthesis approach to automated design of multi-domain systems. In addition, most MEMS system-level design is accomplished by modeling entire microelectromechanical systems as single behavioral entities having no lower hierarchical level in design. If there is any change in geometric parameters or topology, a whole new model must be created, and this substantially lengthens design cycles.

The BG/GP 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 (Fan et al. [2001][2002]). At the first or higher level of system synthesis of MEMS, the BG/GP approach can help to obtain a high-level description of a system that assembles the system from a library of existing components in an automated manner to meet a predefined design specification. Then at 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 rather 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 as well as different results. Figure 5.3 shows a typical structured MEMS synthesis procedure; the BG/GP approach aims to solve the problem of system-level synthesis in an automated manner at the first level.

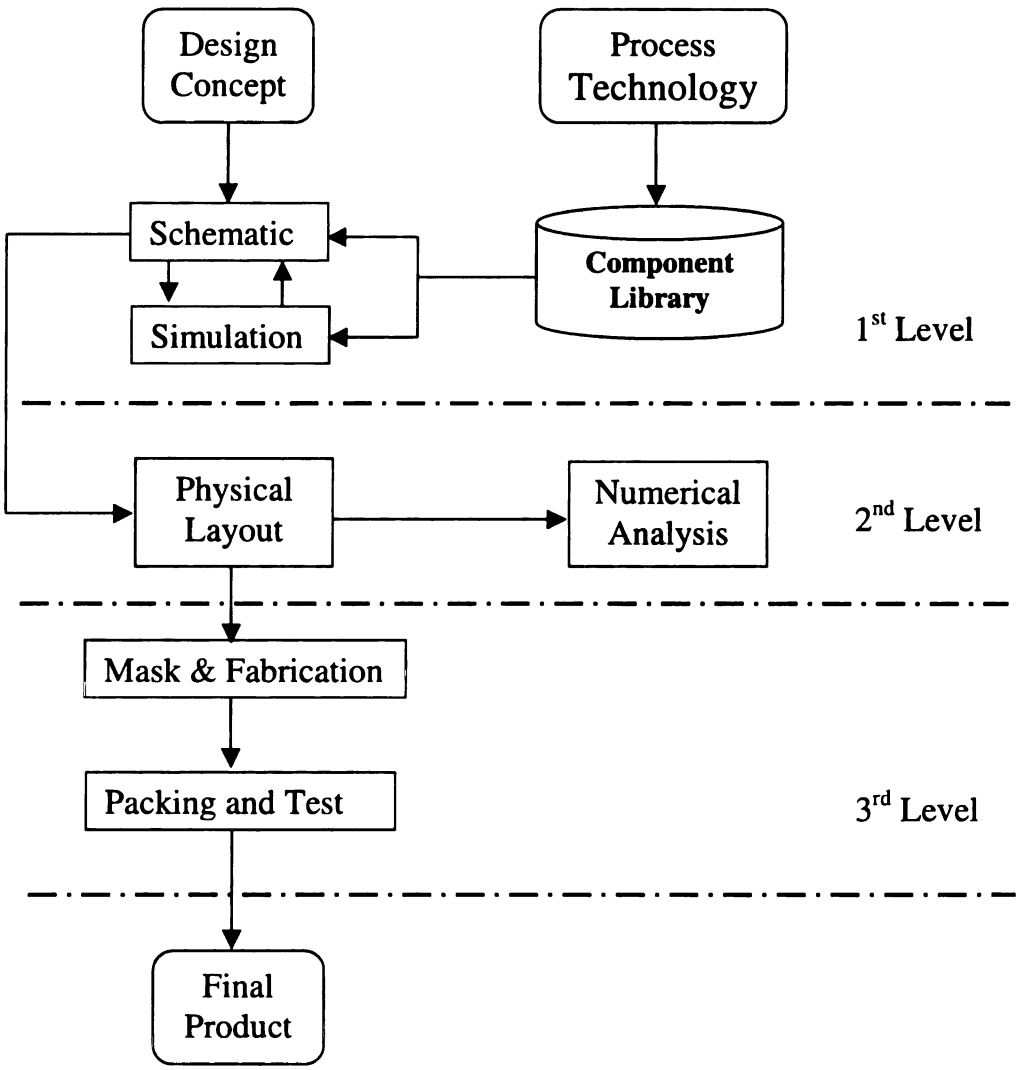


Figure 5.3 Structured MEMS Design Flow

However, in trying to establish an automated synthesis approach for MEMS, we should take cautious steps. Due to the limitations of fabrication technology, there are many constraints in design of MEMS. Unlike 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.

5.4.1 Bond Graphs

The reason we used bond graphs in research on MEMS synthesis is because MEMS are intrinsically multi-domain systems, unlike electronic systems. We need a uniform representation of MEMS so that designers can not only shift among different hierarchies of design abstractions but also can 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 graph a good candidate for use in open-ended design search. Figure 5.4 shows an example of a single bond graph model that represents a resonator unit in any of three different application domains. It is also very 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. 5.5).

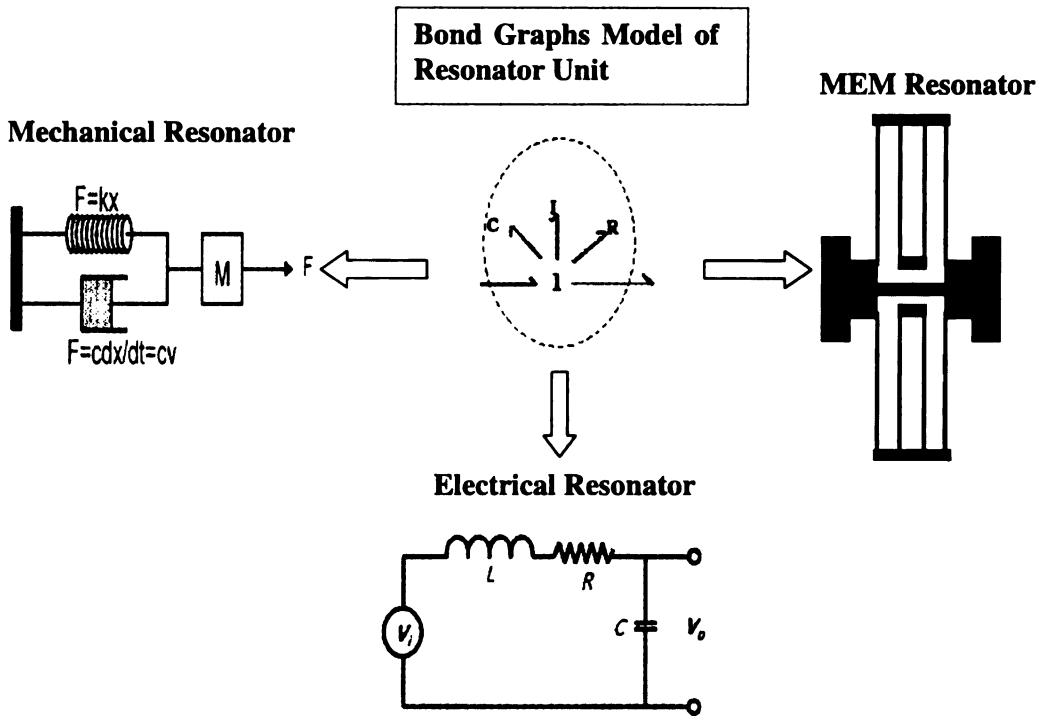


Figure 5.4 One bond graph represents resonators in different application domains

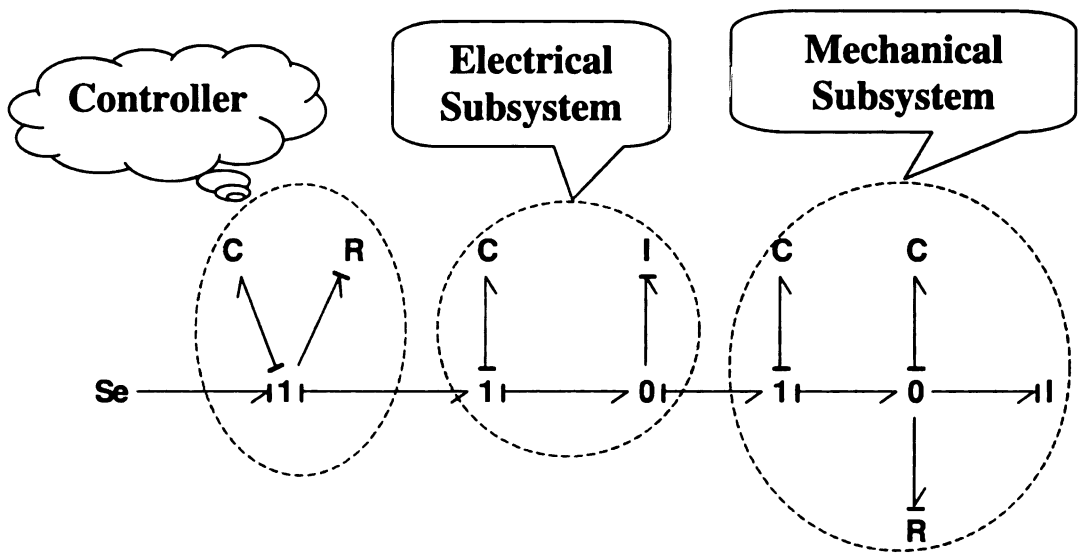


Figure 5.5 Bond graph representing a mechatronic system with mixed energy domains and a controller subsystem

5.4.2 Combining Bond Graphs and Genetic Programming

As was discussed in Chapter 3, the most common form of genetic programming (Koza [1994]) uses trees to represent the entities to be evolved. Defining a proper 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. In this work, the genotypes assembled from the function sets are *constructors* which, upon execution, specify a bond graph. In other words, when the genotype is executed, it generates the phenotype in a developmental manner. 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 those bond graphs can be interpreted as systems in different physical domains, chosen as appropriate to the circumstances. Figure 5.6 illustrates the role of bond graphs in the mappings from genotypes to phenotypes and Figure 5.7 gives a particular example in the domain of electrical circuits.

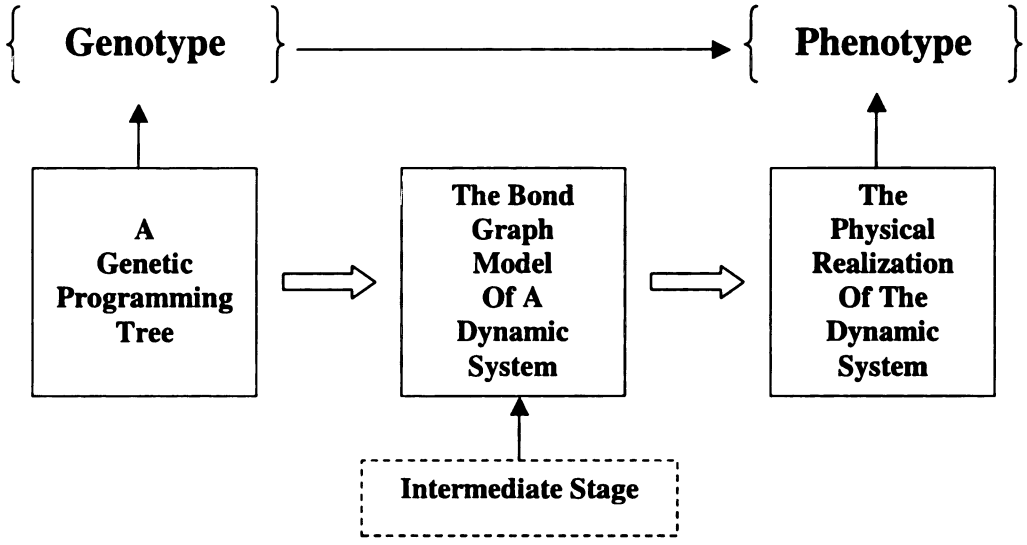


Figure 5.6 Genotype-Phenotype mapping

Figure 5.7 Example of Genotype-Phenotype Mapping in the Electrical Circuit Domain (to be continued)
Part I: The Genetic Programming Tree to generate Bond Graph models for the phenotype – the electrical circuit

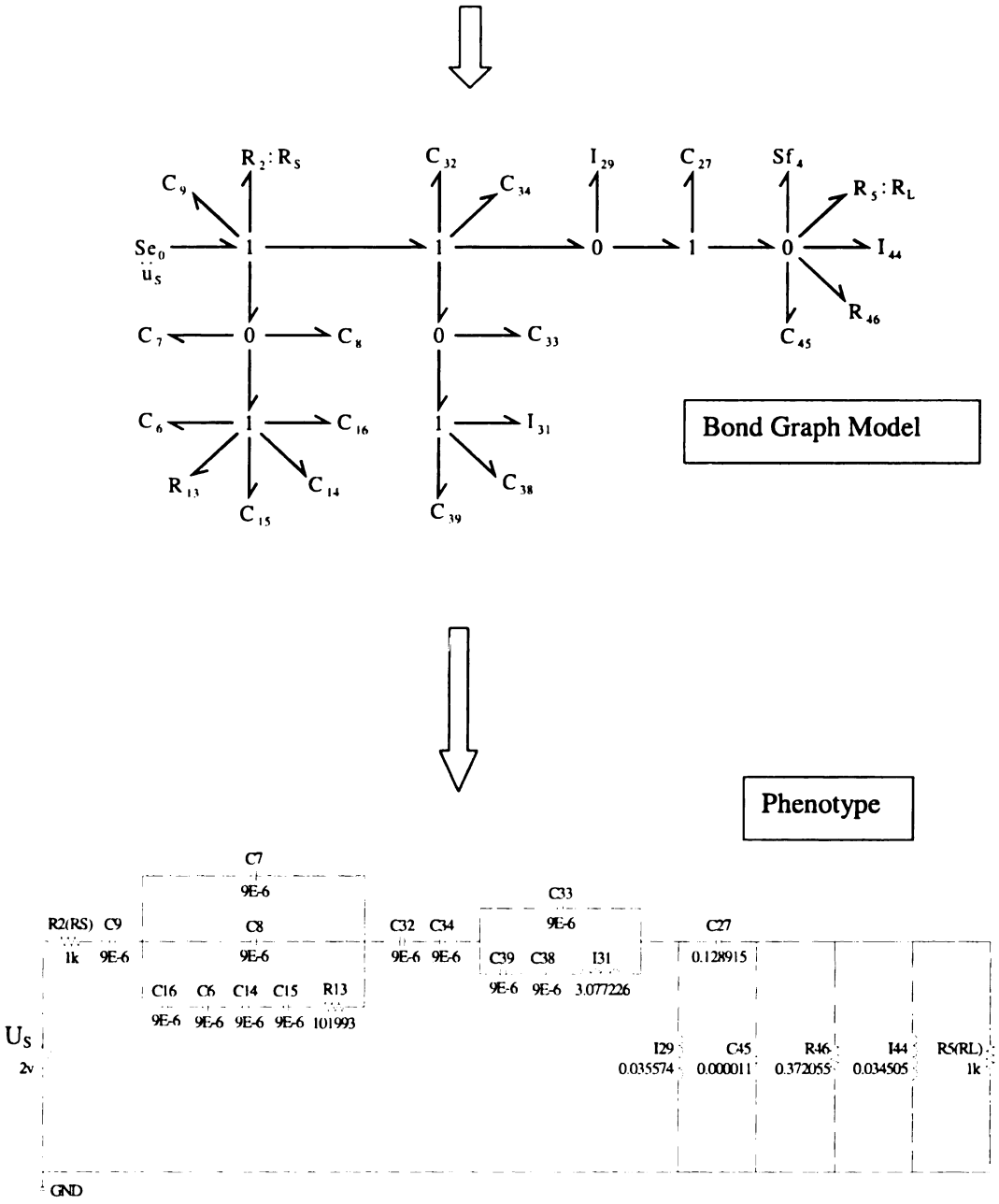


Figure 5.7 Example of Genotype-Phenotype Mapping in the Electrical Circuit Domain Part II: The Bond Graph Model Realized to the Phenotype - the Electrical Circuit Model

5.4.3 Filter Topology

Automated synthesis of an RF MEM device, a micro-mechanical bandpass filter, is used as an example in this research (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 concatenated Resonator Units (RUs) and Bridging Units (BUs) or RUs and Coupling Units (CUs). Figure 5.8 shows the layout of a typical resonator unit widely used in microsystems, along with its equivalent circuit representation and bond graph representation. Figure 5.9 and Figure 5.10 illustrates the layouts and bond graph representations of two widely accepted filter topologies, labeled I and II. Their corresponding bond graph representations are also shown.

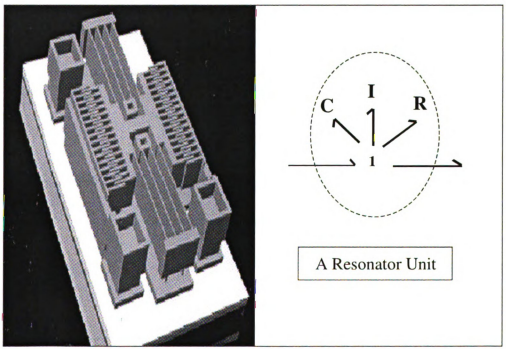


Figure 5.8 Resonator Unit and its Representations as both Bond Graph and Equivalent Circuit

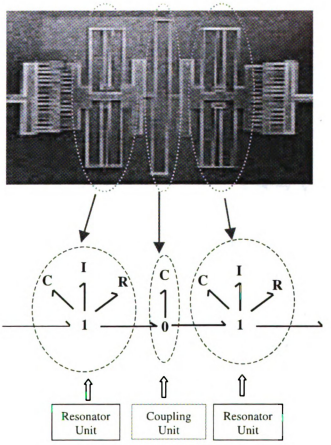


Figure 5.9 MEM filter topology I

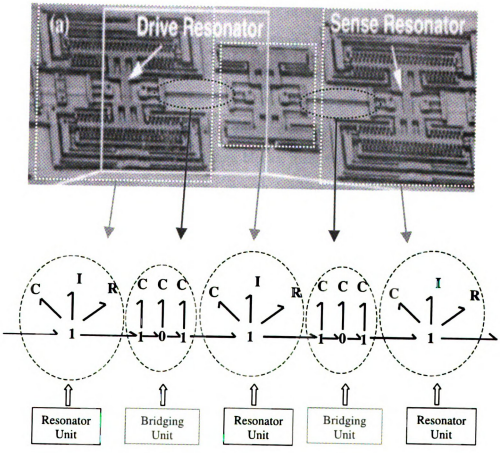


Figure 5.10 MEM Filter Topology II

5.4.4 Realizable Function Set

The most common form of genetic programming uses trees to represent the entities to be evolved. Defining of a proper 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. In this research, a basic function set and a higher-complexity, modular function set are presented and listed in Tables 5.1 and 5.2. Operators in the basic function set aim to construct primitive building blocks and assemble them into a system, while operators in the modular function set purport to utilize relatively modular and predefined building blocks composed of primitive building blocks, assembling them into a system. 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 promise for improving its search efficiency (Seo et al. [2003]). However, in this research, 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.

Examples of operators, namely insert_CU and insert_RU, are illustrated in Figures 5.11 and 5.12. Examples of basic operators are available in our earlier work (Fan et al. [2001]). Figure 5.11 explains how the insert_BU function works. A Bridging Unit (BU) is a subsystem composed of three capacitors with the same parameters, attached together with a 0-junction in the center and 1-junctions at the left and right ends. After execution of the insert_BU function, an additional modifiable site (2) appears at the rightmost newly created bond. As illustrated in Figure 5.12, 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 ERC-typed sites, (4), (5), and (6), for numerical evolution of parameters.

Table 5.1. Operators in Basic Function Set

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_J	Insert a 0-junction in a bond
insert_J	Insert a 1-junction in a bond
replace_	Replace the current element
replace_	Replace the current element
replace_	Replace the current element
+	Sum two ERCs
-	Substract two ERCs
enda	End terminal for add functions
endi	End terminal for insert
endr	End terminal for replace
erc	Ephemeral Random Constant

Table 5.2. 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
+	Sum two ERCs
-	Subtract two ERCs
endn	End terminal for add
endb	End terminal for insert
endr	End terminal for replace
erc	Ephemeral Random Constant

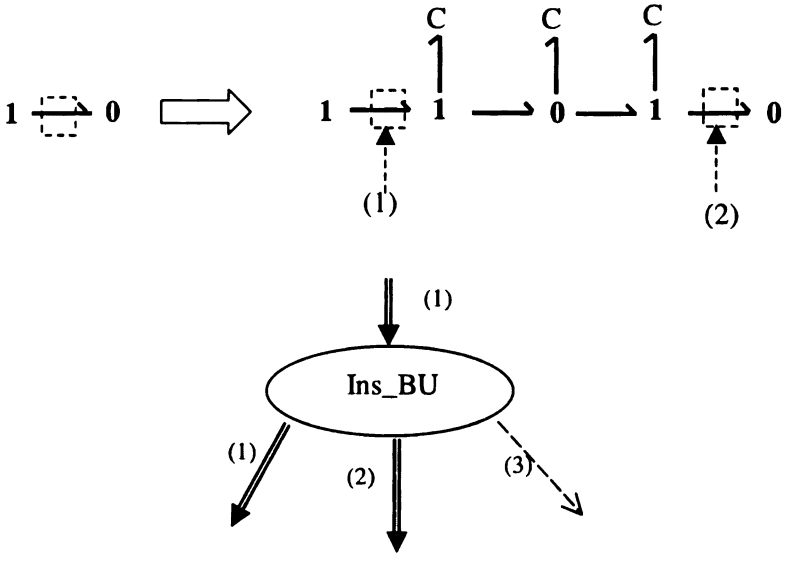


Figure 5.11 Operator to Insert Bridging Unit

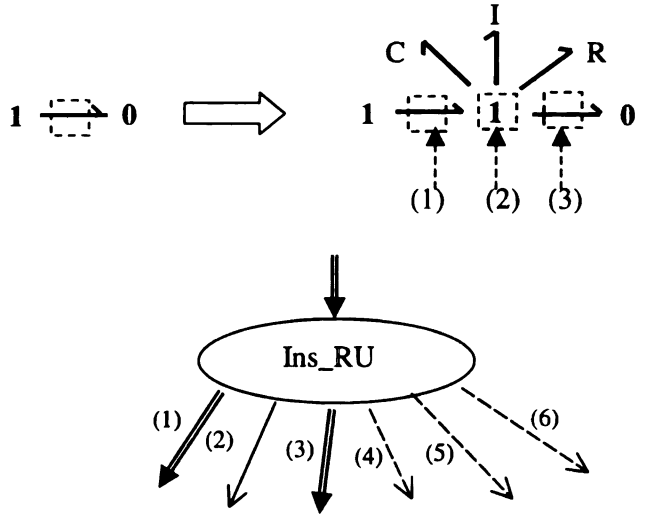


Figure 5.12 Operator to Insert Resonator Unit

BG/GP is a quite general approach to automate synthesis of multidisciplinary systems. Using a basic set of building blocks, BG/GP can perform topologically open composition of an unconstrained design. However, engineering systems in the real world are often limited by various constraints. So if BG/GP is to be used to synthesize real-world engineering systems, it must enforce those constraints.

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 belonging 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 BG/GP approach is to define a realizable function set that, throughout execution, will produce only phenotypes that can be built using existing or expected technology.

As is already known, if we analyze the system of MEM filters of (Wang and Nguyen [1999]) from a bond graph viewpoint, we find that the filters are basically composed of Resonator Units (RUs) and Coupling Units (CUs). Another popular MEM filter topology includes Resonator Units and Bridging Units (BUs). 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 resonator units.

$$\Re 1 = \{f\_tree, f\_insert\_J1, f\_insert\_RU, f\_insert\_CU, f\_add\_C, f\_add\_R, f\_add\_I\}$$

$$\Re 2 = \{f\_tree, f\_insert\_J1, f\_insert\_RU, f\_insert\_BU, f\_add\_C, f\_add\_R, f\_add\_I\}$$

5.4.5 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 bond graph shown in Figure 5.13 as our embryo.

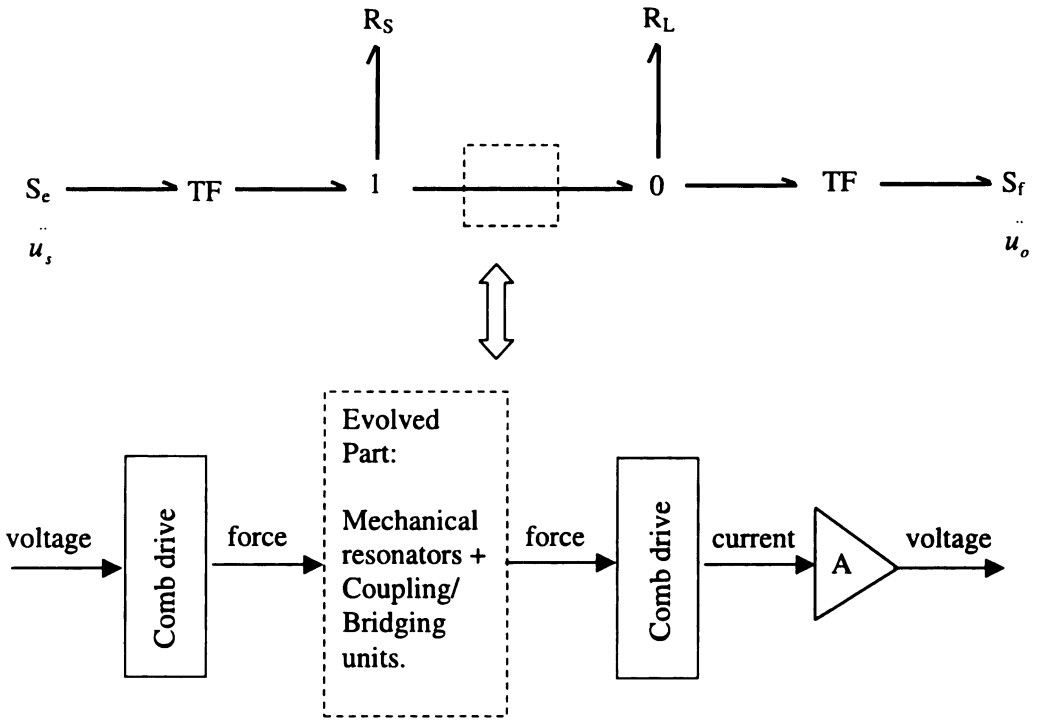


Figure 5.13 Design Embryo of the MEM Filter

5.4.6 Adaptive Fitness Function

Within the frequency range of interest, $f_{range} = [f_{\min}, f_{\max}]$, logarithmically sample 100 points. Here, f_{range} = [0.1, 1000K] Hz.

Compare the magnitudes of the frequency response at the sample points with target magnitudes, which are 1.0 within the pass frequency range of [316, 1000] Hz, and 0.0 otherwise, between 0.1 and 1000KHz.

Compute their differences and get a sum of squared differences as raw fitness, defined as $Fitness_{raw}$. If the initial raw fitness value $Fitness_{raw}^{0}$ < Threshold, change

f_{range} to $f_{range}^{*} = [f_{\min}^{*}, f_{\max}^{*}]$ Usually $f_{range}^{*} \subset f_{range}$. Repeat the above steps and obtain a new raw fitness value $Fitness_{raw}^{1}$. We obtain a final raw fitness value as sum of the two, represented by $Fitness_{raw} = Fitness_{raw}^{0} + Fitness_{raw}^{1}$.

Then normalized fitness is calculated according to:

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

The reason to use adaptive fitness evaluation is that after a GP population has reached a fairly high fitness value as a group, the differences of frequency responses of individuals need to be centered on a more constrained frequency range. In this circumstance, if there is not sufficient sampling within this much smaller frequency range, the GP may lack sufficient search pressure to push the search forward. The normalized fitness is calculated from the sampling differences between the frequency response magnitudes of the synthesized systems and the target responses. Therefore, we adaptively change and narrow the frequency range to be heavily sampled. The effect is analogous to narrowing the search window onto a smaller yet most significant area, magnifying it, and continuing to search this area with closer scrutiny.

5.4.7 Experimental Setup

We used a strongly-typed version of lilgp 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 this 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.

5.4.8 Experimental Results

The GP program obtains satisfactory results on a Pentium-IV 1GHz in 1000~1250 minutes. Experimental results show the strong topological search capability of genetic programming and feasibility of our BG/GP approach for finding realizable designs 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 the topological search capability of the BG/GP approach BG/BP shows potential for exploring more complicated topologies of future MEMS

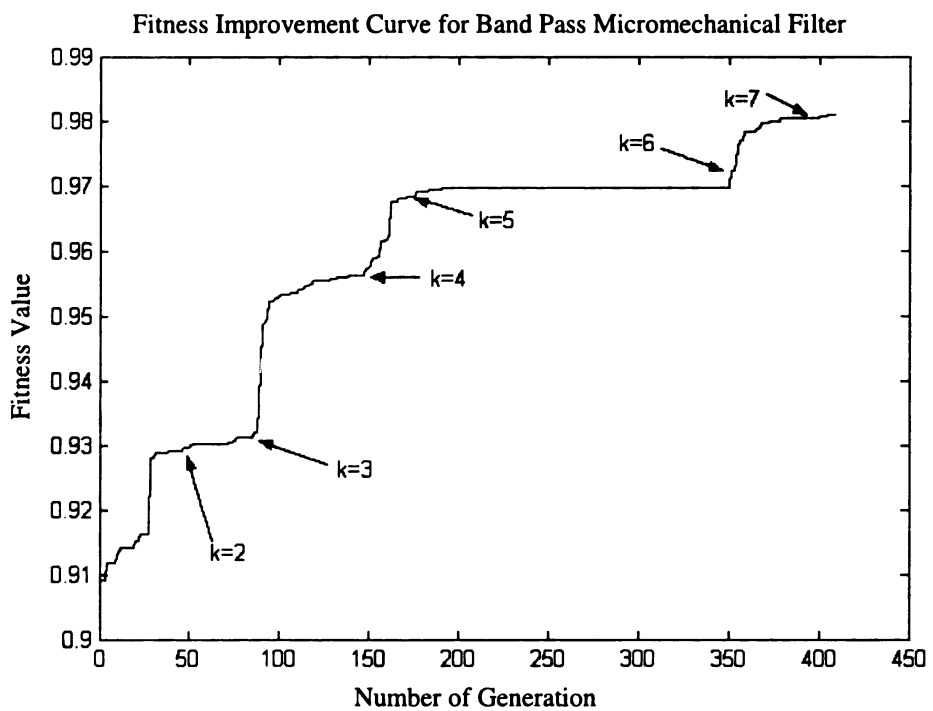


Figure 5.14 Fitness Improvement Curve

design and the ever-progressing technology frontiers of MEMS fabrication.

In Figure 5.14, 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 – unsurprising 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 are shown in Figure 5.15.

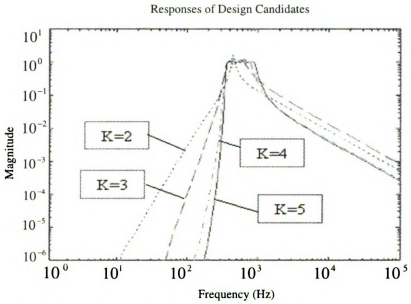


Figure 5.15 Frequency responses of a sampling of design candidates, which evolved topologies (and associated parameter sets) with larger numbers, K, of resonators as the evolution progressed. All results are from one genetic programming run of the BG/GP approach

A layout of a design candidate with four resonators and three coupling units as well as its bond graph representation is shown below in Figure 5.16. 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 BG/GP approach, it is also possible to explore novel topologies of MEM filter design. In this case, we may not necessarily 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. Figure 5.17 gives an example of a novel topology for a MEM filter design evolved using such a semi-realizable function set. An attempt to fabricate this kind of topology is being carried out at the University of California, Santa Barbara [Shaw, 2004].

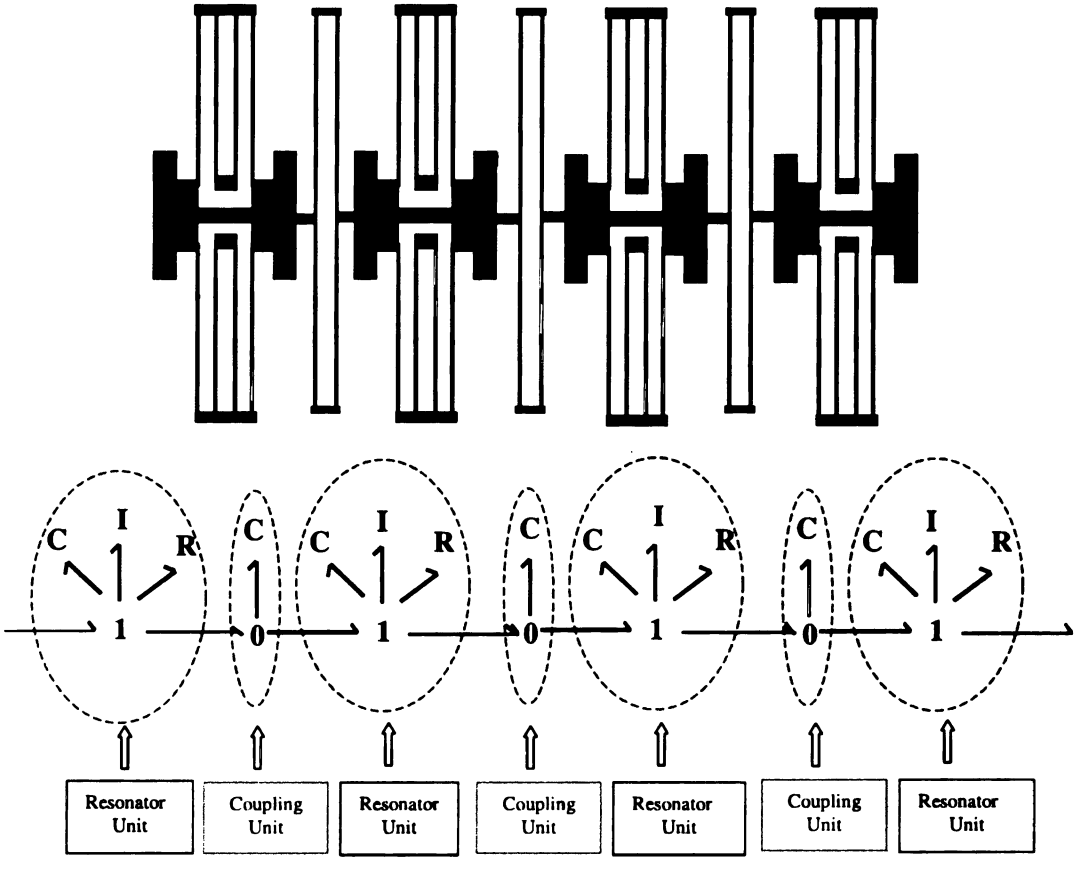


Figure 5.16 Layout and bond graph representation of a design candidate from the experiment, with four resonator units coupled by three coupling units

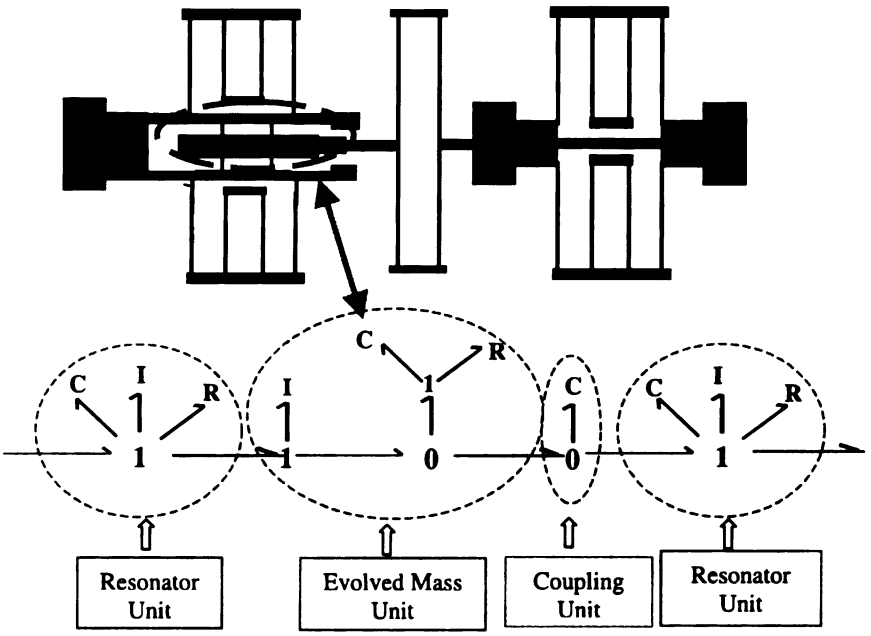


Figure 5.17 A novel topology of MEM filter and its bond graph representation as evolved by the BG/GP approach using a semi-realizable function set.

5.5 Second-Level Physical Layout Synthesis

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. Layout synthesis automatically generates valid or optimized geometric sizing parameters for cell components, which in most cases are commonly used 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 need to be satisfied, or from layout-level objectives such as minimum areas occupied. Our approach is to model this lower-level design problem as a formal constrained optimization problem, and then solve it with powerful optimization techniques, resulting in a tool that automates the design 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.

5.5.1 Formulation of Layout Synthesis as an Optimization Problem

In this research, we decided to use 14 design variables for an example cell component, a folded-flexure comb-drive microresonator fabricated in a polysilicon surface microstructural process (Fig. 5.18). Design variables and their constraints are listed as follows (Fig. 5.19) (Fedder and Mukherjee [1996]):

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

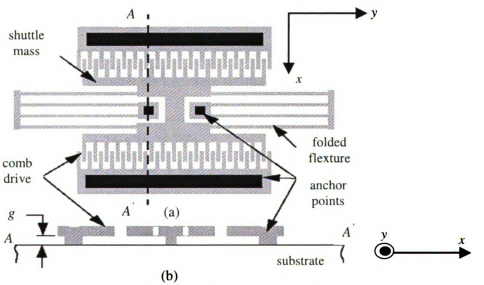


Figure 5.18 A folded-flexure comb-drive microresonator fabricated in a polysilicon surface microstructural process a) Layout; b) Cross-section A-A' (Fedder G. and Mukherjee T. [1996])

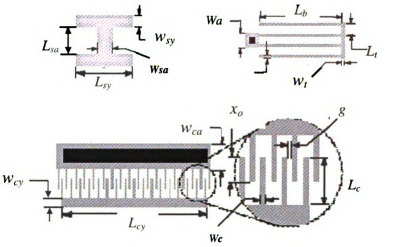


Figure 5.19 Major design variables for microresonators

Note that the first 13 design variables have units of μm. The fourteenth design variable has units of volts.

In addition, we assume $t = w_c = g = d$ in our design for simplicity. Some design variables are predefined for this technology: they are $w_{ba} = 11$, $w_{ca} = 14$, $\delta = 4$, $N = 10$.

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:

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

Among them, the first three are linear constraints, and the fourth is a nonlinear constraint because the term x_{disp} is highly nonlinear. $x_{disp} = QF_{e,x} / K_x$, where $F_{e,x} = 1.12\varepsilon_0 NV^2 t / g$, $Q = \sqrt{M_x K_x / B_x^2}$.

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

$$\begin{cases} K_x = 0.27N/m \\ Bx = 5.18\times10^{-6} kg \cdot m^2 \\ M_x = 4.0\times10^{-6} kg \end{cases}$$

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

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

where $\alpha = (W_t / W_b)^3$

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

$$M_x = M_s + \tfrac{1}{4} M_t + \tfrac{12}{35} M_b$$

where $M_s = \rho A_s$, $M_t = \rho A_t$, $M_b = \rho A_b$

$$A_s = w_{sa} L_{sa} + 2w_{sy} L_{sy}$$

$$A_t = 2w_{ca} L_{cy},$$

$$A_b = 8L_b w_b + 2w_t(2L_t + w_a + 2w_b)$$

As an alternative, we can also put reformulations of these three constraint equations into our design objectives, expressing them as differences to be minimized. In that case, we actually deal with a multi-objective constrained optimization problem. We take the objective function with the following normalized Sum of Squared Error (SSE) format:

$$\vec{f(x)} = \tfrac{1}{0.27^2}(K_x - 0.27)^2 + \tfrac{1}{(5.18\times10^{-6})^2}(B_x - 5.18\times10^{-6})^2 + \tfrac{1}{(4.0\times10^{-6})^2}(M_x - 4.0\times10^{-6})^2$$

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 or size difference

a particular design can achieve, based on specific fabrication procedures. In addition, the actual sizes of geometric shapes should be integer multiples of the feature size λ, such as $\lambda, 2\lambda, 5\lambda, 10\lambda \ldots$ etc. In this research, we set $\lambda = 0.09\ \mu m$.

While it is very difficult for many numerical optimization approaches (for example, gradient-based approaches) to include considerations of feature size constraints (Fedder and Mukherjee [1996]), 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.

5.5.2 Solving the Optimization Problem Using GA

In trying to solve constrained optimization problems using genetic algorithms or classical deterministic optimization methods, penalty function methods have been the most popular approach, because of their simplicity and ease of implementation. In this chapter, we use a special constrained GA that exploits pair-wise comparisons in a tournament selection operator to devise a penalty function approach that does not require any penalty parameter (Deb [2000]). Careful comparisons among feasible and infeasible solutions are made so as to provide a search direction towards the feasible region. Once sufficient feasible solutions are found, a niching method (along with a controlled mutation operator) is used to maintain diversity among feasible solutions. This allows a real-parameter GA's crossover operator to continuously find better feasible solutions, gradually leading the search nearer to the true optimum solution.

The parameters for setting the constrained GA are as follows:

Variable Boundaries: Rigid	Population size: 500
Total no. of generations: 100	Crossover probability: 0.9000
Mutation probability (real): 0.15	Niching parameter: 0.9000
Exponent (n for SBX): 2.00	Exponent (n for mutation): 50.00

In ten 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 5.3. It can be seen that during the ten GA runs using different seeds, the GA performs very steadily. Almost all runs achieved objective values, namely, the Normalized Squared Sum of Errors (NSSE), within the range of 1.0E-6. The mean value of NSSE is 3.4E-6, while the standard deviation of NSSE is 3.86E-6. The biggest NSSE is 1.4E-5. However, the normalized squared sum of errors of 1.4E-5 is still considered very good result. It also appears that there are many alternative and rather different ways in which parameters can be set and still produce behavior rather close to that desired.

Table 5.3 Layout parameters obtained in ten GA runs (different random seeds)

RUN NO.	1	2	3	4	5	6	7	8	9	10
$L_b(\mu m)$	261.63	261.45	261.09	262.44	262.35	260.82	261.72	261.90	262.62	259.47
$w_b(\mu m)$	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98
$L_t(\mu m)$	3.87	4.32	3.87	3.60	8.46	2.43	2.52	5.13	6.84	11.88
$w_t(\mu m)$	2.70	2.25	2.52	2.52	2.25	1.98	1.98	2.88	3.33	1.98
$L_{sy}(\mu m)$	3.69	2.88	2.07	4.41	1.98	1.98	3.60	1.98	2.79	2.79
$w_{sy}(\mu m)$	14.13	12.60	15.93	11.52	10.80	9.99	11.52	15.30	12.60	14.31
$w_{sa}(\mu m)$	18.63	18.18	10.98	11.70	11.34	11.16	10.17	11.70	14.58	10.80
$w_{cy}(\mu m)$	146.16	151.83	122.31	141.12	137.25	56.61	110.70	76.14	247.50	173.16
$L_{cy}(\mu m)$	15.66	20.79	23.85	17.37	23.85	30.69	22.68	21.96	8.91	20.79
$L_c(\mu m)$	199.26	187.29	174.06	202.41	181.89	154.71	188.19	162.09	161.91	183.60
$w_c(\mu m)$	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98	1.98
$L_{sa}(\mu m)$	2.25	2.16	2.52	2.43	2.88	1.98	2.70	2.70	6.30	2.70
$x_o(\mu m)$	10.26	96.12	24.66	34.92	10.35	14.94	30.87	20.34	25.83	4.86
$V(volt)$	66.06	70.29	75.51	64.98	72.27	85.14	69.93	81.09	81.27	71.55
Obj. Value	4E-006	3E-006	3E-006	1E-006	1E-006	1.4E-005	2E-006	2E-006	1E-006	3E-006

The Figure 5.20 shows a typical GA run with Normalized SSE vs. Generation. It is noted that the logarithmic value of NSSE reduces at a nearly linear rate in accordance to generation number. At generation 91, the NSSE reduces to the value of 1.0E-6.

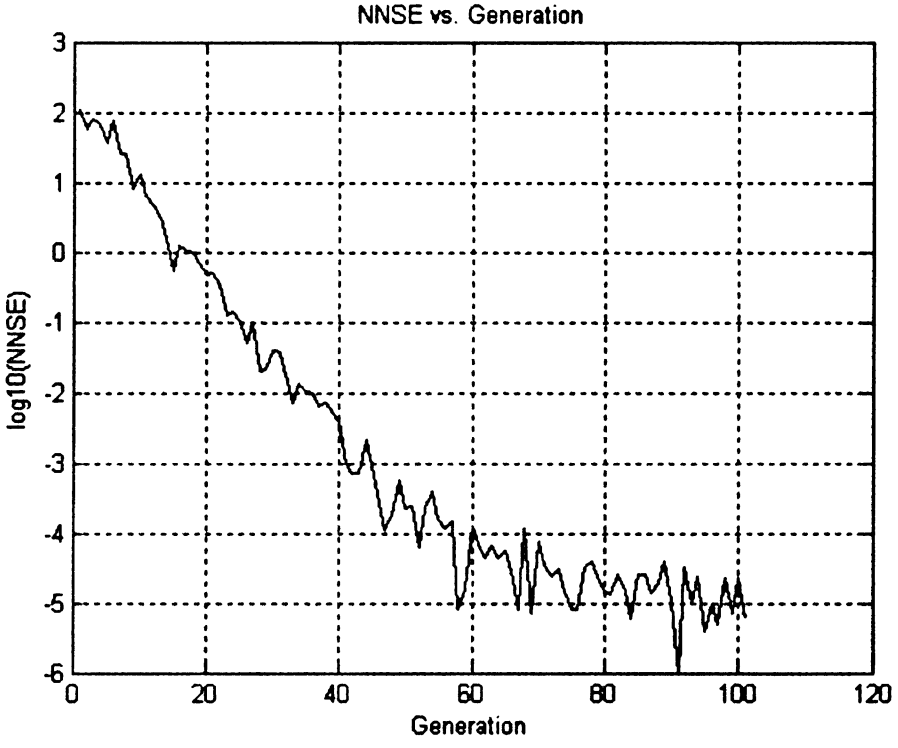


Figure 5.20 Curve of Normalized SSE vs. Generation

5.6 Conclusions

In MEMS, there are two or three levels of designs that need to be synthesized. Usually the design process must start with synthesis of a schematic design of the overall system, including topology and behavior-related parameters, 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 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.

This chapter 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 (Ma and Antonsson [2000]) (Zhou and Agogino [2001]), this is the first work reported to seek to automate 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 as 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 (Paredis et al. [2001]). 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 (Kim and Papalambros [2000]). 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.

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 behavior of each piece. It is used for verifying results of hand calculations for simple models, 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 designers from having to consider the fabrication details, allowing them to focus on the functional design of the device and system instead (Ma and Antonsson [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.

CHAPTER VI
CONCLUSIONS

6.1 Contributions

With mechatronics emerging as an independent and integrated discipline of the 21st century, the research results of this dissertation are of particular significance because it is one of the first endeavors to address the challenging issue of design automation of mechatronic systems. In this research, we have developed and applied a general framework, namely, the BG/GP approach, for automated conceptual design of mechatronics systems. The BG/GP approach combines both bond graphs as a modeling tool to unify representations of mixed-domain subsystems across different physical domains in typical mechatronic systems, and genetic programming as a strong search tool to explore the open-ended design space of mechatronic systems. We have verified the effectiveness and efficiency of the BG/GP approach through a set of case studies, including electrical passive analog filter design, mechanical typewriter redesign, and system-level synthesis of MEMS.

An interesting and instructive comparison is made between Electronic Design Automation (EDA) and Mechatronic Design Automation (MDA). Because energy and information flow between modules of mechatronic systems can be transferred through electric wires, mechatronic systems can be modularized more easily than conventional mechanical systems, and are thus more amenable to modular design automation approaches. It is believed that MDA holds great promise and may be the next big wave after EDA. In particular, micromechatronic (microelectromechanical) systems (MEMS) have the potential to be the first type of mechatronic systems that can achieve comparable

success to that achieved in Electronic Design Automation. A structured and hierarchical design methodology for MEMS is recommended and studied in this research. The preliminary results of both system-level behavioral synthesis and second level layout synthesis show that automated synthesis of MEMS is a very promising research area.

Because block diagrams could be mapped to bond graphs, bond graphs can also be used to represent designs of controllers. This feature of bond graphs is important for mechatronics research because a typical modern mechatronic system not only includes a *plant* consisting of mechanical, electrical, and/or hydraulic subsystems, etc., but also includes a critical *controller* part that regulates and coordinates movements and functionalities of various physical subsystems in the plant. It has been proved that the BG/GP approach is capable of concurrent design of both controllers and plants of mechatronic systems in a joint research project on vehicle suspension system design (Wang, Fan et al. [2004]). However, as that joint work is a major topic in the dissertation of Wang, it is not included in this dissertation.

6.2 Future work

There are many research directions to undertake in the future to extend the current BG/GP framework.

One direct enhancement is to include more complex multi-port components in the component library as the building blocks for design configurations. For example, in the current implementation of case studies in Chapter 4, basic components used to construct design candidates include 1-port C, 1-port I and 1-port R elements. These components can be generalized to multiport C-field, I-field and R-field (Karnopp, [2000]). Actually,

in Chapter 5, the modular component of coupling unit can be represented by a 2-port C-field. However, in this dissertation, multiport field is not investigated in depth. More understanding of using multiport field is underway and integration of multiport field into the BG/GP framework is the next research task of the author.

In the current BG/GP framework, we focus our research on generating conceptual designs of mechatronic systems that satisfy predefined design specifications, in case studies of Chapter 4. Detailed design, as well as design hierarchy, is discussed in Chapter 5. More work to build a composable design and simulation environment is needed so that designers can migrate among different design levels conveniently. In composable design and simulation environment, any component involved in design not only has a high-level behavioral model, but also one or more detailed physical form models (Diaz-Calderon, [1999]).

Design robustness is a very important research topic to bridge the gap between academic research results and industrial application tools. In industrial practices, the design parameters may have many more constraints than those in the academic research environment. Fabrication and measurement errors make it difficult for component parameters of a real-world product to match the design parameters exactly. In addition, changes in working environments such as temperature fluctuation and/or electromagnetic interference may easily introduce noise to the working system and make its components' equivalent parameters deviant from their designed values. Robust design (Sanchez, [1994]) aims to address the issue of making designs that are insensitive to those noise and parameter variations, and is an interesting research topic that the author is going to undertake in his future career.

To increase scalability of evolutionary synthesis, another line of research has drawn much attention recently. By augmenting experimental biology with computer models of development, biologist are building a greater understanding of how developmental process construct the staggering complexities of living organisms (Kumar and Bentley, [2003]). Taking advantage of this understanding, I expect to enhance the capability of the current evolutionary synthesis approach to reach designs that is far more complex than current evolved designs in terms of functional complexity. Related research topics include morphogenesis, cell signaling and regeneration, investigations of synthetic developmental mechanisms, and its implications in automated synthesis of engineering systems.

APPENDIX A

CAUSAL CONSTRAINTS

Fixed Causality

Fixed causality holds at a port when the equations only allow one of the two port variables to be the outgoing variable. This occurs at sources: an effort source (Se), by definition, always has its effort variable as signal output, and has the causal stroke outwards. This causality is called effort-out causality or effort causality. A flow source (S_f) clearly has a flow-out causality or flow causality.

Another situation where fixed causality occurs is at nonlinear elements, in cases in which the equations for that port cannot be inverted (for example, potentially yielding division by zero). This is possible at R, GY, TF, C and I elements. Thus, there are two reasons to impose a fixed causality:

1. There is no relationship between the port variables.

2. The equations are not invertable ('singular').

Constrained Causality

At TF, GY, 0– and 1–junctions, relationships exist between the causalities of the various ports of the element. These relations are causal constraints, since the causality of a particular port imposes the causality of the other ports. At a TF, one of the ports has effort-out causality and the other has flow-out causality. At a GY, either both ports have effort-out causality or both have flow-out causality. At a 0–junction, where all efforts are the same, exactly one bond must bring in the effort. This implies that 0–junctions always have exactly one causal stroke on junction side of their ports. The causal condition at a 1–junction is the dual of the 0-junction. The flows must sum to zero, thus exactly one bond

can have its value determined by the junction, implying that exactly one bond has the causal stroke away from the 1–junction. [Zhun, I think that what you had said was wrong, but please check that what I said is correct for bond graphs. This would represent a major error if uncorrected. Where did this language come from?]

Preferred Causality

At the storage elements, the causality determines whether an integration or differentiation with respect to time will hold. Integration has preference over differentiation in causal assignment. In the integrating form, an initial condition must be specified. Integration with respect to time is a process that can be realized physically. Differentiation is not always physically realizable, since information at future time points is needed. Another drawback of differentiation is that when the input contains a step function: the output then becomes infinite. Therefore, integrating causality is seen as the preferred causality. This implies that C–elements have effort-out causality and I–elements have flow-out causality as their preferred causal assignments.

We will present an example to illustrate this. When a voltage u is imposed on an electrical capacitor (a C–element), the current i is the result of the constitutive equation of the capacitor:

$$i = C\frac{du}{dt}$$

A differentiation is thus happening. We have a problem when the voltage instantly steps to another value, since the current required to achieve that will be infinite (the derivative of a step is infinite). This is not the case when the current is imposed on a capacitor. Now, an integral is used:

$$u = u_0 + \int i dt$$

The first case is flow-out causality (effort imposed, flow the result), and the second case is effort–out causality, which is the preferred causality. Furthermore, an effort–out causality also results in a state variable with an initial condition, u0.

In an inductor, the dual form of the C–element is used: flow-out causality will result in integral causality, and is the preferred assignment. Step changes in voltage produce integral changes in current.

Indifferent Causality

Indifferent causality is used when there are no causal constraints. At a linear R, it does not matter which of the port variables is the output (or response). Consider an electrical resistor. Imposing a current (flow) yields:

$$u = iR$$

It is also possible to impose a voltage (effort) on the linear resistor: $i = \frac{u}{R}$

There is no difference in feasibility between choosing the current as stimulus variable and the voltage as response variable, or the other way around.

In summary, the Se and Sf have fixed causalities, the C and I have preferred causalities, the TF, GY, 0 and 1 have constrained causalities, and the R has an indifferent causality (provided that the equations characterizing these basic elements are all invertable). When the equations are not invertable, a fixed causality must be used.

APPENDIX B

STATE-SPACE FORMULATION FOR BOND GRAPH MODELS

The problem of state-space formulation for bond graph models can be formulated as follows. Given a bond graph composed of elements from the basic set **{C, I, R, S_e, S_f, TF, GY, 0, 1}**, find a method of generating state-space equations of the form

$$\dot{X} = AX + BU \tag{B.1}$$

or

$$\dot{X} = \phi(X, U) \tag{B.2}$$

where

C – capacitance	I – inertance	R – dissipation
Se-- source of effort	S_f-- source of flow	TF – modulated transformer
GY – modulated gyrator	0 – zero junction	1 – one junction

A bond graph can be organized into a form consisting of storage field, loss field, source field and junction structure. The storage (energy) field is a collection of C and I elements. The loss (dissipation) field is composed of R elements. The source field is composed of source elements Se and Sf. The collection of elements from the set {TF, GY, 0, 1} forms the junction structure, which is a power-preserving multi-port subsystem. Any bond graphs composed of elements from the basic set may be organized into the form shown in the Figure B.1 describing the system division.

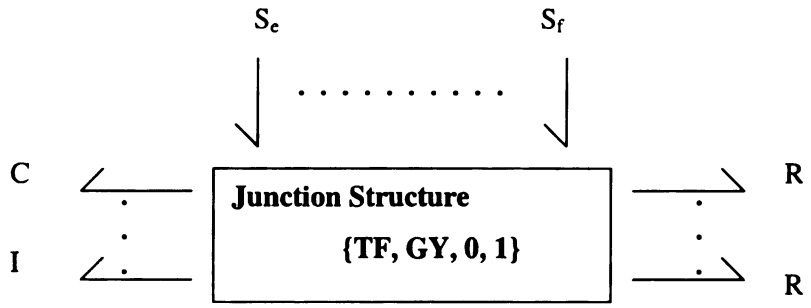


Figure B.1 Basic fields of multiport systems: acausal form

After causality is assigned to the bond graphs according to the systematic approach described above, Figure B.1 becomes Figure B.B. The graph is said to have integral causality. In particular, this means that every C-field port and every I-field port is as shown in Figure B.B. According to causality, Figure B.2 identifies for the port of each characteristic field the input and output variables, namely, loss, storage and source. An R port can have either e in and f out, or the reverse, depending upon causality. C and I ports are always defined as shown. The variable x in the storage field is the true energy variable, and its derivative dx/dt is taken as input, with the co-energy variable z as output. The outputs of the source field are the independent driving functions u (e for Se, f for S_f), and the inputs to the source elements are the complementary bond variables v.

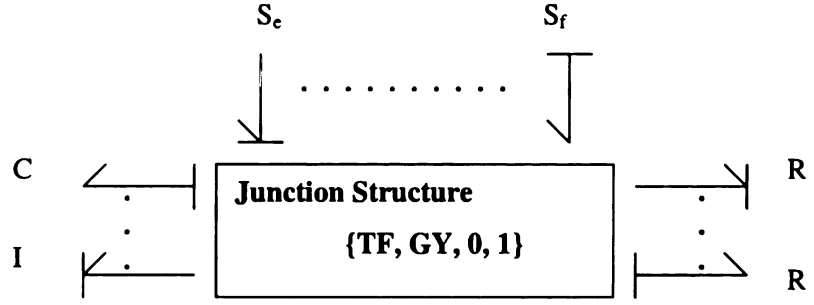


Figure B.2 Symbolic form for integration causality

Based on the definitions given for each field port in Figure B.2, the entire system may be represented in causal form as shown in Figure B.3. Each of the arrows represents a vector of variables, and the vector sets are paired according to the field types.

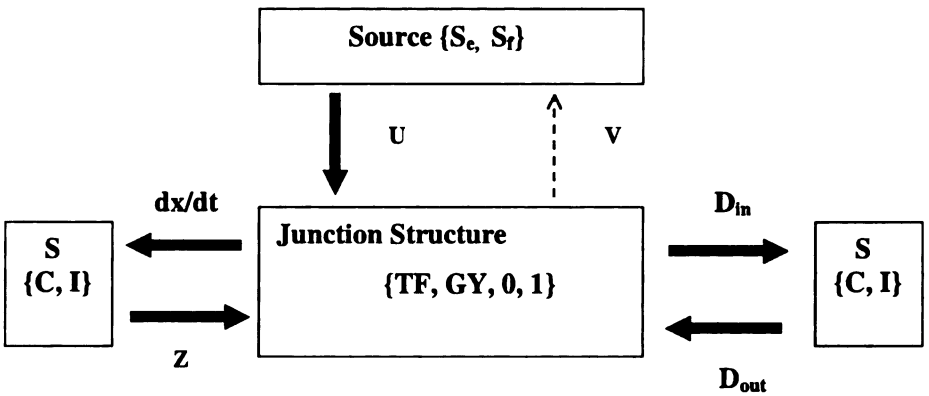


Figure B.3 Significant vectors for systems having integration causality

Then, the linear field equations in standard form in the dissipation field can be given by $D_{out} = L \cdot D_{in}$, (B.3)

For the case of storage field, we have $Z = S \cdot X$ (B.4)

The junction structure yields expressions for the dx/dt and D_{in} vectors in terms of the inputs to the junction structure, namely, Z, D_{out}, and U. Provided the elements TF and GY all have constant modulus, we have

$$\dot{X} = J_{SS}Z + J_{SL}D_{out} + J_{SU}U \quad \text{(B.5)}$$

$$D_{in} = J_{LS}Z + J_{LL}D_{out} + J_{LU}U \quad \text{(B.6)}$$

where J matrices are the constraints imposed by the junction structure between sets of ports. Reduction of the four equations (B.3) through (B.6) to a single state-space equation of the desired form may be accomplished quite directly. Substituting (B.4) into (B.5), we get

$$\dot{X} = J_{SS}Z + J_{SL}LD_{in} + J_{SU}U \quad \text{(B.7)}$$

Then substituting (B.6) into (B.7), we obtain

$$\dot{X} = J_{SS}Z + J_{SL}L(J_{LS}Z + J_{LL}D_{out} + J_{LU}U) + J_{SU}U \tag{B.8}$$

Equation (B.4) and (B.6) may be combined and solved to give

$$D_{out} = L(I - J_{LL}L)^{-1}J_{LS}Z + L(I - J_{LL}L)^{-1}J_{LL}U \tag{B.9}$$

Substituting (B.3) into (B.9), we get

$$D_{out} = L(I - J_{LL}L)^{-1}J_{LS}SX + L(I - J_{LL}L)^{-1}J_{LL}U \tag{B.10}$$

Substituting (B.10) into (B.8), we get

$$\dot{X} = [J_{SS}S + J_{SL}L(I - J_{LL}L)^{-1}J_{LS}S]X + [J_{SU} + J_{SL}L(I - J_{LL}L)^{-1}J_{LU}]U \tag{B.11}$$

This can be written as

$$\dot{X} = AX + BU \tag{B.12}$$

where

$$A = [J_{SS} + J_{SL}L(I - J_{LL}L)^{-1}J_{LS}]S \tag{B.13}$$

$$B = [J_{SU} + J_{SL}L(I - J_{LL}L)^{-1}J_{LU}] \tag{B.14}$$

BIBLIOGRAPHY

F. Broenink, [1999], "Introduction to physical systems modelling with bond graphs," *In SiE Whitebook on Simulation methodologies*, http://www.rt.el.utwente.nl/bnk/papers/BondGraphsV2.pdf.

Amerongen, J. van and P.C. Breedveld, [2003], "Modelling of physical systems for the design and control of mechatronic systems (IFAC Professional Brief)," *Annual Reviews in Control 27, Elsevier Ltd., ISBN S1367-5788*, pp 87–117

M. I. Campbell, J. Cagan and K. Kotovsky, [1999] "A-Design: An Agent-Based Approach to Conceptual Design in a Dyanmic Environment," *Research in Engineering Design*, vol. 11, pp.172-192.

J. M. Cabanells, J. Feléz, [1999], "Dynamic Systems Optimization Based on Pseudo Bond Graph," *1999 International Conference on Bond Graph Modeling and Simulation*, pp.50-55.

Antonio Diaz-Calderon, [2000], *A Composable Simulation Environment to Support the Design of Mechatronic Systems*, PhD Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University.

S. Carlson-Skalak, M. D. White, Y. Teng, [1998] "Using Evolutionary Algorithm for Catalog Design," *Research in Engineering Design*, vol 10, pp. 63-83.

Chakrabarti, T. P. Bligh, [1996a], "An Approach to Functional Synthesis of Solutions in Mechanical Conceptual Design. Part I: Kind Synthesis," *Research in Engineering Design*, vol. 8, pp.52-62.

Chakrabarti, T. P. Bligh, [1996b], "An Approach to Functional Synthesis of Solutions in Mechanical Conceptual Design. Part I: Spatial Configuration," *Research in Engineering Design*, vol. 8, pp.116-124.

Chakrabarti, T. P. Bligh, [1994], "An Approach to Functional Synthesis of Solutions in Mechanical Conceptual Design. Part I: Introduction and Knowledge Representation," *Research in Engineering Design*, vol. 6, pp.127-141.

E. Coelingh, T. J. A. de Vries, J. V. Amerongen, [1998], "Automated Performance Assessment of Mechatronic Motion Systems During the Conceptual Design Stage," *Proceedings of the 3nd International Conference on Advanced Mechatroncis, Okayama, Japan*, pp.472-477

B. Danielson, J. Foster and D. Frincke, [1998], "GABSys: Using Genetic Algorithms to Breed a Combustion Engine," *Proc. of IEEE Conf. on Evolutionary Computation*, pp. 259-264.

Deb K., [2003], *Multi-Objective Optimization Using Evolutionary Algorithms*, Chichester, UK: Wiley Publisher

Deb K., "An efficient constraint handling method for genetic algorithms", *Comput. Methods Appl. Mech. Engrg.*, Vol. 186, (2000) 311-338

D. Eby, R. C. Averill, W. Punch, E. D. Goodman [1998], "Evaluation of Injection Island GA Performance on Flywheel Design Optimization," *Proceedings, Third Conference on Adaptive Computing in Design and Manufacturing*, Plymouth, England, Springer Verlag, pp.121-136.

Z. Fan, K. Seo, R. C. Rosenberg, J. Hu, E. D. Goodman, [2003], "System-Level Synthesis of MEMS via Genetic Programming and Bond Graphs", *Proc. 2003 Genetic and Evolutionary Computing Conference, Chicago, Springer, Lecture Notes in Computer Science*, 2058-2071

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

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

Fedder, G.K. and Q. Jing, [1999], "A Hierarchical Circuit-Level Design Methodology for Microelectromechanical Systems", *IEEE Transactions on Circuits and Systems II (TCAS)*, vol. 46, no. 10, pp. 1309-1315.

Fedder G. and Mukherjee T., [1996], "Physical Design for Surface-Micromachined MEMS", *Proceedings of the Fifth ACM/SIGDA Physical Design Workshop*, April, pp. 53-60.

J. S. Gero, [1996], Computers and Creative Design, in M. Tan and R. Teh (eds), The Global Design Studio, National University of Singarpo, pp. 11-19

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

E. D. Goodman, R. C. Averill, W. F. Punch, D. J. Eby, [1997a], "Parallel Genetic Algorithms in Optimization of Composite Structures," *Proc. Second World Conference on Soft Computing (WSC2)*, Springer Verlag, pp. 199-208.

E. D. Goodman, [1996], *An Introduction to GALOPPS*, GARAGe Technical Report #96-07-01, Michigan State University.

J. B. Grimbleby [2000] Automatic analogue circuit synthesis using genetic algorithms. *IEE Proc. – Circuits Devices Systems.*319-323.

M. Heinrich, W. E. Jeungst, [1996], " Resource Base Paradigm for the Configuring of Technical Systems from Modular Components," *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference,* Irvine, CA, August, pp.18-22.

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

J. Hu, E. D. Goodman, K. Seo, M. Pei, [2002] Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002,* New York, 772-779.

S. P. Hoover, J. R. Rinderle, [1989], "A Synthesis Strategy for Mechanical Devices," *Research in Engineering Design*, vol 1, pp.87-103.

D. C. Karnopp, D. L. Margolis, R. C. Rosenberg, [2000] *System Dynamics, A Unified Approach, 3rd Ed.,* John Wiley & Sons.

Kim, H.M., Michelena, N.F., Papalambros, P.Y., and Jiang, T., [2000], "Target Cascading in Optimal System Design," *Proceedings of the 2000 ASME Design Automation Conference*, DAC-14265, Baltimore, Maryland, USA.

S. Kota, C. L. Lee, [1993], "General Framework for Configuration Design: Part I – Methodology," *Journal of Engineering Design*, vol. 4, no. 4, pp.277-289.

J. R. Koza, F. H. Bennet, D. Andre, M. A. Keane, [1999a], *Genetic Programming III, Darwinian Invention and Problem Solving*, Morgan Kaufmann Publishers.

J. R. Koza et al., [1999b], "Automatic Creation of Both the Topology and Parameters for a Robust Controller by Means of Genetic Programming," *Proceedings of the 1999 IEEE International Symposium on Intelligent Control, Intelligent Systems, and Semiotics*. Piscataway, NJ: IEEE. pp.344-352.

J. R. Koza, F. H. Bennet, D. Andre, M. A. Keane, F. Dunlap, [1997a], "Automate Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, pp.109-128.

J. R. Koza, D. Andre, F. H. Bennet, M. A. Keane, [1997b], "Evolution Using Genetic Programming of a Low-Distortion 96 Decibel Operational Amplifier," *Proceedings of the 1997 ACM Symposium on Applied Computing*, San Jose, California, pp.207-216.

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

J. R. Koza, [1992], *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press.

Kumar, S. and Bentley, P. J. [2003], *On Growth, Form and Computers*. Academic Press, London.

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

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

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

C. J. J. Paredis, A. Diaz-Calderon, R. Sinha, and P. K. Khosla, [2001], "Composable Models for Simulation-Based Design", *Engineering with Computers* 17, pp. 112-128.

D. R. Prabhu, [1989], "Synthesis of Systems from Specifications Containing Orientations and Positions Associated with Flow Variables", *Proc. 1989 Design Automation Conference*, Montreal, Canada

W. Punch, [1998], *lil-gp 1.1 User's Manual*, Technical Report, Genetic Algorithms Research and Algorithms Group, Michigan State University.

W. Punch, R. C. Averill, E.D. Goodman, S.-C. Lin, Y. Ding [1995], "Design Using Genetic Algorithms – Some Results for Laminated Composite Structures," *IEEE Expert*, vol 10 (1), pp. 42-49.

M. Raymer, W. Punch, E. Goodman, and L. Kuhn [1996], "Genetic Programming for Improved Data Mining –Application to the Biochemistry of Protein Interactions," *Proc. First Genetic Programming Conference*, Stanford University, pp. 375-380.

R. C. Redfield, [1999], "Bond Graphs in Dynamic Systems Designs: Concepts for a Continuously Variable Transmission," *1999 International Conference on Bond Graph Modeling and Simulation*, pp. 225-230.

Y. Reich, [1995], "A Critical Review of General Design Theory," *Research in Engineering Design*, vol. 7, pp.1-18.

D. W. Rosen, T. J. Peters, [1996], "The Role of Topology in Engineering Design Research," *Research in Engineering Design*, vol. 8, pp.81-98.

R. C. Rosenberg, [1996a], *The ENPORT User's Manual*, Rosencode Associates, Inc.

R. C. Rosenberg, M. K. Hales, and M. Minor, [1996b], "Engineering Icons for Multidisciplinary Systems," *Proc.ASME IMECE* 1996, DSC-V.58, pp.665-672.

R. C. Rosenberg and Y-y. Wang , [1993a], "Multiport Subsystems," *Proc. 1993 IEEE International Conference on Systems, Man and Cybernetics*, Le Touquet, France.

R. C. Rosenberg, [1993b], "Reflections on Engineering Systems and Bond Graphs," *ASME Trans. J. Dynamic Systems, Measurements and Control*, V.115, pp.242-251.

R. C. Rosenberg, J. Whitesell, and J. Reid, [1992], "Extendible Simulation Software for Dynamic Systems," *SIMULATION*, 58:3, pp.175-183.

R. C. Rosenberg, [1971], "State-Space Formulation for Bond Graph Models of Multiport Systems," *ASME Trans. J. Dynamic Systems, Measurements and Control*, V.93, pp.35-40.

Susan M. Sanchez, [1994], "A Robust Design Tutorial", *Proceedings of the 26th conference on Winter simulation*, pp. 106-113.

K. Seo, Z. Fan, J. Hu, E. Goodman, R. Rosenberg, [2003], "Toward an Automated Design Method for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming," *Mechatronics*, 13 (8-9), pp: 851-885

J. E. Sharpe, R. H. Bracewell, [1995]. " The Use of Bond Graph Reasoning for the Design of Interdisciplinary Schemes," *1995 International Conference on Bond Graph Modeling and Simulation*, pp. 116-121.

J. L. Stein, L. S. Louca, [1995], " A Component-based Modeling Approach for System Design: Theory and Implementation," *1995 International Conference on Bond Graph Modeling and Simulation*, pp.109-115.

S. Shaw, Michigan State University, October, 2003, *pers. comm.*

E. Tay, W. Flowers and J. Barrus, [1998], "Automated Generation and Analysis of Dynamic System Designs," *Research in Engineering Design*, vol 10, pp. 15-29.

Vargas-Hernandez N., J. Shah, Z. Lacroix, [2003], "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*, pp. 255-261.

G. Wang, E. D. Goodman, W. Punch, [1997b], "Toward the Optimization of a Class of Blackbox Optimization Algorithms," *Proc. IEEE Internat. Conf. on Tools for Artif. Intell.*, pp. 348-356.

Jiachuan Wang, Zhun Fan, Janis P. Terpenny, and Erik D. Goodman, [2004], "Knowledge Interaction with Genetic Programming in Mechatronic Systems Design Using Bond Graphs," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation* (to appear).

Wang, J. and Terpenny, J., [2003], "Interactive Evolutionary Solution Synthesis in Fuzzy Set-based Preliminary Engineering Design", *Special Issue on Soft Computing in Manufacturing, Journal of Intelligent Manufacturing*, Vol. 14. pp. 153-167

D. E. Whitney, [1996], "Why Mechanical Design Cannot be like VLSI Design," *Research in Engineering Design*, vol 8, pp. 125-138.

K. Youcef-Toumi, Y. Ye, A. Glaviano, P. Andrson, [1999], "Automated Zero Dynamics Derivation from Bond Graph Models," *1999 International Conference on Bond Graph Modeling and Simulation*, pp. 39-44.

N. Zhou, B. Zhu, A.M. Agogino, K.S.J. Pister, [2001], "Evolutionary Synthesis of MEMS (Microelectronic Mechanical Systems) Design". Proceedings of ANNIE 2001, Intelligent Engineering Systems through Artificial Neural Networks, Volume 11, ASME Press, pp. 197-202

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

AUTHOR'S PUBLICATION

Journal Papers

1. **Z. Fan**, K. Seo, R. Rosenberg, J. Hu, E. Goodman, A Novel Evolutionary Engineering Design Approach for Mixed-Domain Systems. *Journal of Engineering Optimization* (in press), 2004
2. Jiachuan Wang, **Zhun Fan**, Janis P. Terpenny, and Erik 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, Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation* (to appear), 2004
3. J. Hu, E. D. Goodman, K. Seo, **Z. Fan**, R. C. Rosenberg, The Hierarchical Fair Competition (HFC) Model for Continuing Evolutionary Algorithms, *Journal of Evolutionary Computation* (to appear), 2004.
4. K. Seo, **Z. Fan**, J. Hu, E. Goodman, R. Rosenberg, Toward an Automated Design Method for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming. *Mechatronics* 13 (8-9), 2003, pp: 851-885
5. K. Seo, J. Hu, **Z. Fan,** E. D. Goodman, and R. C. Rosenberg, Automated Design Approaches for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming, *The International Journal of Computers, Systems and Signals*, vol.3, no.1, pp.55-70, 2002.

Conference Papers

1. **Z. Fan,** Jiachuan Wang, E. Goodman, Ronald Rosenberg, Kisung Seo, Jianjun Hu, Hierarchical Evolutionary Synthesis of MEMS, *IEEE Congress on Evolutionary Computation*, CEC2004 (to appear)
2. **Z. Fan,** K. Seo, R. Rosenberg, J. Hu, E. Goodman, System-Level Synthesis of MEMS via Genetic Programming and Bond Graphs, *Proc. 2003 Genetic and Evolutionary Computing Conference*, Chicago, Springer, Lecture Notes in Computer Science, July, 2003, pp. 2058-2071, **Best-Paper Award Finalist**.

3. K. Seo, **Z. Fan,** J. Hu, E. Goodman, and R. Rosenberg, Dense and Switched Modular Primitives for Bond Graph Model Design, *Proc. 2003 Genetic and Evolutionary Computing Conference,* Chicago, Springer, Lecture Notes in Computer Science, July, 2003, pp. 1764-1775.
4. **Z. Fan,** K. Seo, R. Rosenberg, J. Hu, E. Goodman, Computational Synthesis of Multi-Domain Systems – Application in MEMS, *Proceedings of the 2003 AAAI Spring Symposium - Computational Synthesis: From Basic Building Blocks to High Level Functionality*, Stanford, California, March, 24-26, 2003, pp. 59-66.
5. Erik D. Goodman, Kisung Seo, **Zhun Fan**, Jianjun Hu, Ronald C. Rosenberg, Automated Design of Mechatronic Systems: Novel Search Methods and Modular Primitives to Enable Real-World Applications, *Service and Manufacturing Grantees and Research Conference Proceedings*, Edited by R.G. Reddy, The University of Alabama, Tuscaloosa, AL 35487, USA, 2003, pp 120-138.
6. J. Hu, E. D. Goodman, K. Seo, **Z. Fan,** R. C. Rosenberg, HEMO: A Sustainable Multi-Objective Evolutionary Optimization Framework. *Proc. 2003 Genetic and Evolutionary Computing Conference, Chicago, Springer, Lecture Notes in Computer Science,* July, 2003, pp. 1029-1040.
7. **Z. Fan,** K. Seo, R. Rosenberg, J. Hu, E. Goodman, Exploring Multiple Design Topologies using Genetic Programming and Bond Graphs, accepted for full presentation by *2002 Genetic and Evolutionary Computation Conference*, ISGEC Press, New York, July, 2002, pp. 1073-1080
8. J. Hu, K. Seo, S. Li, **Z. Fan,** R. C. Rosenberg, E. D. Goodman, Structure Fitness Sharing (SFS) for Evolutionary Design by Genetic Programming, *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO-2002, New York, July, 2002, pp. 780-787.
9. **Z. Fan,** J. Hu, K. Seo, E. Goodman, R. Rosenberg, and B. Zhang, Bond Graph Representation and GP for Automated Analog Filter Design, *2001 Genetic and Evolutionary Computation Conference Late-Breaking Papers*, ISGEC Press, San Francisco, 2001, pp. 81-86.