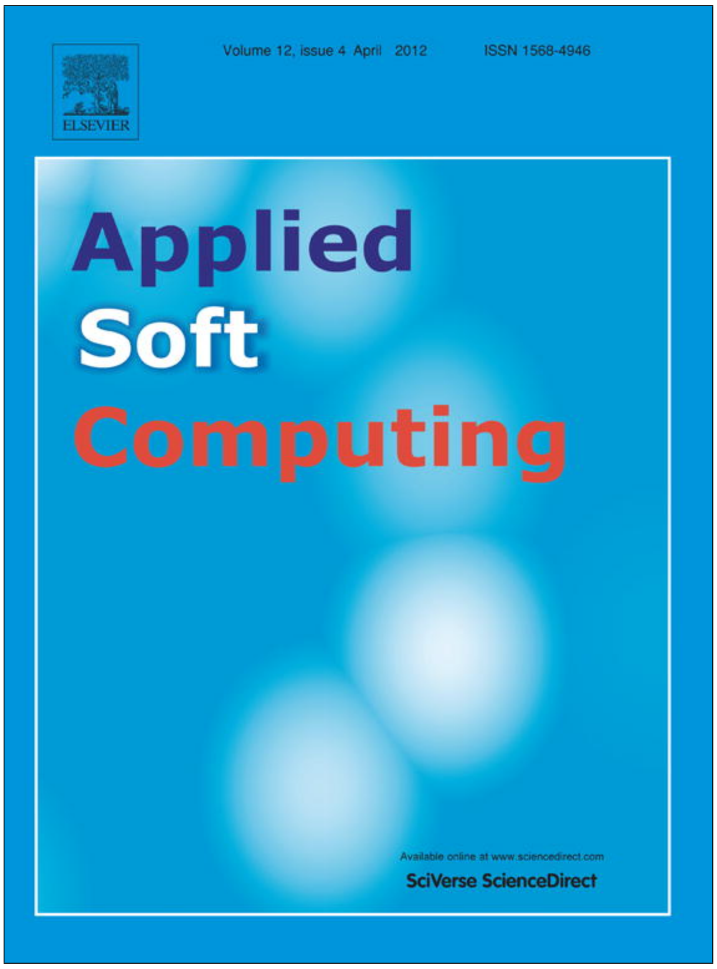

Volume 12, issue 4 April 2012
ISSN 1568-4946
ELSEVIER
Applied Soft Computing
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Applied Soft Computing 12 (2012) 1309–1321

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Automated synthesis of multiple analog circuits using evolutionary computation for redundancy-based fault-tolerance

Kyung-Joong Kim[a,*], Sung-Bae Cho[b]

[a] Department of Computer Engineering, Sejong University, Seoul, Republic of Korea
[b] Department of Computer Science, Yonsei University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 23 May 2011
Received in revised form 28 October 2011
Accepted 7 December 2011
Available online 17 December 2011

Keywords:
Analog circuits
Fault-tolerance
Redundancy
Evolutionary computation
Filters

ABSTRACT

Analog circuits are one of the most important parts of modern electronic systems and the failure of electronic hardware presents a critical threat to the completion of modern aircraft, spacecraft, and robot missions. Compared to digital circuits, designing fault-tolerant analog circuits is a difficult and knowledge-intensive task. A simple but powerful method for robustness is a redundancy approach to use multiple circuits instead of single one. For example, if component failures occur, other redundant components can replace the functions of broken parts and the system can still work. However, there are several research issues to make the redundant system automatically. In this paper, we used evolutionary computation to generate multiple analog circuits automatically and then we combined the solutions to generate robust outputs. Evolutionary computation is a natural way to produce multiple redundant solutions because it is a population-based search. Experimental results on the evolution of the low-pass, high-pass and band-stop filters show that the combination of multiple evolved analog circuits produces results that are more robust than those of the best single circuit.

1. Introduction

The robustness of the system is one of the important issues in knowledge-based systems [1] and there are several ways to make them to be fault-tolerant [2]. First of all, the most widely used method is to prepare redundant modules which replace the original one in case of failure [3]. Although this is simple to be implemented, the same redundant modules can be weak to the same problem which gives damage to the original one. Also, they need additional modules to detect the malfunction of each module and switch them. Secondly, if the faults are known a priori, we can attempt to design specialized systems robust to the problems [4]. However, this often requires much expert knowledge and computational power to design them because of its high complexity of the design problems. Also, they are weak to the unknown problems. Finally, unlike the two passive approaches, researchers try to build active fault-tolerant systems which reconfigure themselves when faults occur [5]. Although this is a promising approach, they are in the early stage of development.

Analog circuits are one of the fundamental parts of modern electro-mechanic systems. Although many analog electronic functions have been replaced with digital equivalents, there still exists a need to use analog circuits [6]. Analog circuit is still used to convert speech signals to digital signals, sensor signals are inputted to microprocessors, and digital outputs are converted to analog signals. Moreover, although we call it as digital systems, all electronic circuits are ultimately analog circuits [14].

The presence of robustness is vital when working with analog circuits because there is usually a possibility of component failure [6]. Physical damage, manufacturing faults, aging, radiation, temperature changes and power surges are possible reasons for such failures. If the system is not fault-tolerant, there is a high possibility of radical performance degradation. Even worse, analog circuits have been widely used for the autonomous systems working without the intervention of human operators in remote and hazardous environments. In those cases, the failure of components results in the significant loss of systems. The purpose of fault-tolerant analog circuits is to maintain functioning even when these kinds of component failures are experienced.

Because designing analog circuits is a difficult, knowledge-oriented process which is not possible without training or experience, there have been a lot of works about the automation of the process, especially with evolutionary computation [7–9]. Evolutionary computation includes a set of methodologies mimicking the natural evolution phenomena: genetic programming [10,11], genetic algorithm [12], and evolution strategies [13]. Koza et al. attempted to evolve analog circuits using genetic programming (GP) based on minimal information about problems such as the number of inputs and outputs [14]. Lohn and Colombano used

* Corresponding author. Tel.: +82 2 3408 3838.
E-mail addresses: kimkj@sejong.ac.kr (K.-J. Kim), sbcho@cs.yonsei.ac.kr (S.-B. Cho).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.12.002

linear representation with a genetic algorithm to evolve filter circuits which is relatively simpler than GP [18].

Although there have been several works on designing fault-tolerant analog circuits with evolutionary computation, they have focused on making a robust single circuit instead of exploiting redundancy. Recently, Hu et al. evolved fault-tolerant filters using genetic programming. These filters are robust to the changing parameters of each component [16]. Hollinger and Gwaltney also evolved fault-tolerant circuits for controlling robots using a genetic algorithm (GA) [19]. In this work, they evolved a circuit that is robust to component removal. Kim et al. evolved analog circuits robust to partial short or disconnection damage to the components and tested them physically [4].

In this paper, we argue that the evolutionary computation can be useful to make robust analog circuits with redundancy. Redundancy is the key concept of fault-tolerant analog circuits because, in general, redundant parts can replace or complement original damaged parts. Evolutionary computation is a population-based search method and maintains multiple redundant solutions. Since it is possible to arrange redundant parts with different architectures with evolutionary computation before system manufacturing, it is expected that the original parts and redundant parts not fail simultaneously.

The redundancy is not a new method but, there are several research topics to be solved for the use of the redundancy in analog circuits. Is it possible to design redundant analog circuits automatically? If possible, how can you choose some of them and combine them? To the best of our knowledge, there is no work addressing those issues in the field of analog circuits.

Evolutionary computation finds multiple redundant analog circuits and they are combined using weighted summing circuits (averaging the outputs of multiple circuits) having an output which is the average of the outputs of each circuit. Because of these redundant circuits, a fault in one circuit can be recovered from the other circuits' normal outputs. In this method, the key point is to prepare multiple redundant circuits with different properties. If the members of the combined circuits are identical, they can suffer from similar failures and the system cannot realize the benefits of synergism. The easiest way to maintain diversity in the population is to use tournament-based selection [20] which prevents few individuals from dominating. In this paper, we used the tournament selection method with mutation-only evolution which allows only one offspring per parent. In addition to the single population approach (increasing the diversity within single population), we propose a method to use multiple populations expecting higher diversity than the single one.

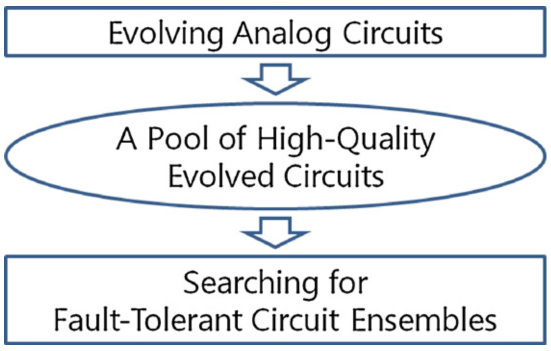

Evolving Analog Circuits
A Pool of High-Quality Evolved Circuits
Searching for Fault-Tolerant Circuit Ensembles

Fig. 1. Overview of the proposed method. At first, the evolutionary algorithm searches for multiple redundant analog circuits. The next step is to find the best ensemble of the circuits in terms of fault-tolerant property.

Following the recent work [4], we have used evolutionary computation to evolve filters, and multiple circuits are combined for robustness (Fig. 1). The fault-tolerance levels of both evolved and combined circuits are also tested by removing each component. At each time, only one component is removed from the analog circuits.

The rest of this paper is organized as follows. Section 2 describes the background including the research on evolving normal and fault-tolerant analog circuits. Section 3 applies the selective ensemble approach to the evolution process and presents the multi-population approach. Section 4 describes the experimental results and analysis.

2. Related works

2.1. Fault-tolerant evolved digital circuits

Hartmann and Haddow show that the artificial evolution is able to automatically generate digital circuit designs robust to noise and faults [36]. They report that the evolved multiplier and adder circuits show a graceful degradation to noise and failures. In the work, they address the possibility to combine the evolved circuits with traditional fault-tolerant schemes (classical redundancy techniques). The evolved circuits also provide valuable insight on the design of fault-tolerant digital circuits.

Schnier and Yao propose a method to evolve diverse fault-tolerant digital circuits using fitness sharing method [31]. In their work, they used negative correlation approach to make individuals

Table 1
Summary of related works on fault-tolerant analog circuits designed by evolutionary computation.

Reference	Fault types	Methods	Tasks
Kim et al. [4]	Partial short and disconnection of one component	ES	Low-pass, high-pass, band-pass, band-stop filters
Kim and Cho [23]	One component removal	ES	Low-pass filter
Hu et al. [16]	Parameter variation	GP	Low-pass, high-pass filters
Keymeulen et al. [24]	Open/close switch	GA	Analog multiplier, xnor gate
Zebulum et al. [25]	One component removal	GA	Compensator
Hollinger and Gwaltney [19]	One component removal, actuator failure	GA	Controller for piezoelectric pipe-crawling robot
Zebulum et al. [26]	Extreme low temperature	GA	Half-wave rectifier, nor gate, controllable oscillator
Ando and Iba [27]	Parameter variation	GA	Band elimination
Layzell and Thompson [28]	One transistor removal	GA	Inverter, amplifier, oscillator
Liu and He [34]	Resistor stuck open capacitor stuck short	GA	Low-pass filter
The proposed	One component removal	ES	Low-pass filter, high-pass filter

ES, evolution strategy; GP, genetic programming.

of population as diverse as possible. After then, they try to combine multiple digital circuits evolved to build fault-tolerant circuits. In the digital circuit evolution, each circuit is represented as a grid (7×7) of basic node functions (and, or, xor, nand and nor). They used majority voting mechanism to combine the output of multiple circuits.

Greenwood and Joshi compare the population-based and correlation-based methods to evolve fault-tolerant digital circuitry [35]. In their work, the test case is a 2×3 binary multiplier circuit. The two methods are evaluated by inserting stuck-at faults at selected locations. A gate output is always one (stuck-at one) or zero (stuck-at zero). They report that the correlation-based method produces the small-size ensemble.

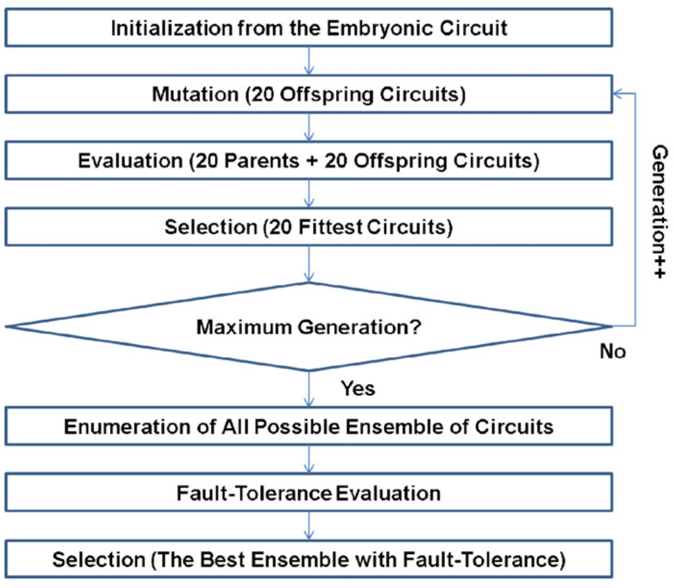
2.2. *Fault-tolerant evolved analog circuits*

Evolutionary computation can be used to design fault-tolerant analog circuits automatically but requires high computational cost (Table 1). It starts from randomly generated analog circuits and tests the fitness of each circuit based on the robustness to known faults or damages. Based on the survival of the fittest in natural evolution, the circuits with high fitness allow to producing more offspring than others. In addition to the selection operation, there are crossover and mutation operations to generate new offspring. It continues this process until they converge or predefined number of generations. Although this is a feasible approach, its computational cost is very high. In the fitness evaluation, it requires multiple simulations of candidate circuits removing one component each time.

There are several types of faults simulated in evolutionary fault-tolerant analog circuit research. They can be classified into two groups: Internal and external faults. In case of internal faults, there are parameter variation, component removal, and switching problems. Although more than one component can be failed at the same time, most works assumed that only one component can be removed from the circuit at each time because of computational cost. Unlike other works, Kim et al. deal with partial short and disconnection damage to components [4]. External faults are related to the problems outside of the analog circuits. It includes damage to the system (sensor, actuator and interface) and environmental conditions (temperature and air pressure).

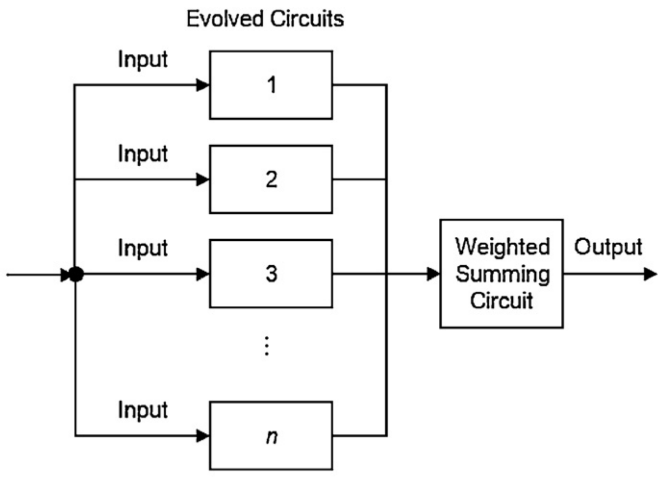
Hu et al. compared three representative approaches to evolve fault-tolerant analog circuits using genetic programming [16]. They are evolving analog filters using GP without incorporating a robustness criterion in the fitness function, applying real parameter genetic algorithm to tune the parameters of evolved filters for robustness, and incorporating robustness criteria in the fitness function. They report that open-ended topology search by GP with robustness criterion in the fitness function is stronger to parameter perturbations than that with parameter tuning alone. Hollinger and Gwaltney evolved fault-tolerant analog circuits for a piezoelectric pipe-crawling robot [19]. The goodness of the circuit with respect to the robustness is measured by each circuit simulated with one component removal. The fitness value is the average of the multiple simulations.

Compared to the digital circuit evolution, the analog circuit evolution is quite different. Unlike the basic node of digital circuit, the analog component outputs real values (voltage level). Also, it is not common to represent analog circuit in a grid. The negative correlation idea is possible because they defined their circuit in the grid. In this representation, it is possible to map one component in circuit A to another component in circuit B if they are placed in the sample coordination. Based on this assumption, it is possible to define the dissimilarity between two analog circuits. In the context of analog circuits, it is not trivial to map components in different circuits. Liu and He apply the negative correlation approach to the evolution of fault-tolerant analog circuit ensemble [34]. In their work, they evaluate the correlation-penalty term based on the outputs of analog circuits. Table 2 summarizes the difference between the work and our proposed method. For example, they use different algorithm to



Initialization from the Embryonic Circuit
Mutation (20 Offspring Circuits)
Evaluation (20 Parents + 20 Offspring Circuits)
Selection (20 Fittest Circuits)
Maximum Generation?
No
Yes
Generation++
Enumeration of All Possible Ensemble of Circuits
Fault-Tolerance Evaluation
Selection (The Best Ensemble with Fault-Tolerance)

(a) Overall procedure



(b) Ensemble implementation

Fig. 2. Overview of the proposed method.

Table 2
The comparison with Liu and He [34].

	Liu and He [34]	The proposed
Evolutionary algorithm	Genetic algorithm + negative correlation	Evolution strategy
Single/multi-population	Multi-population	Single and multi-population
Ensemble formation	The combination of the best circuit from each run	Searching for the best fault-tolerant ensemble (selective ensemble)
Combination	Analog redundancy model	A weighted summing circuit
Fault types	Resistor stuck open, capacitor stuck short	Component removal

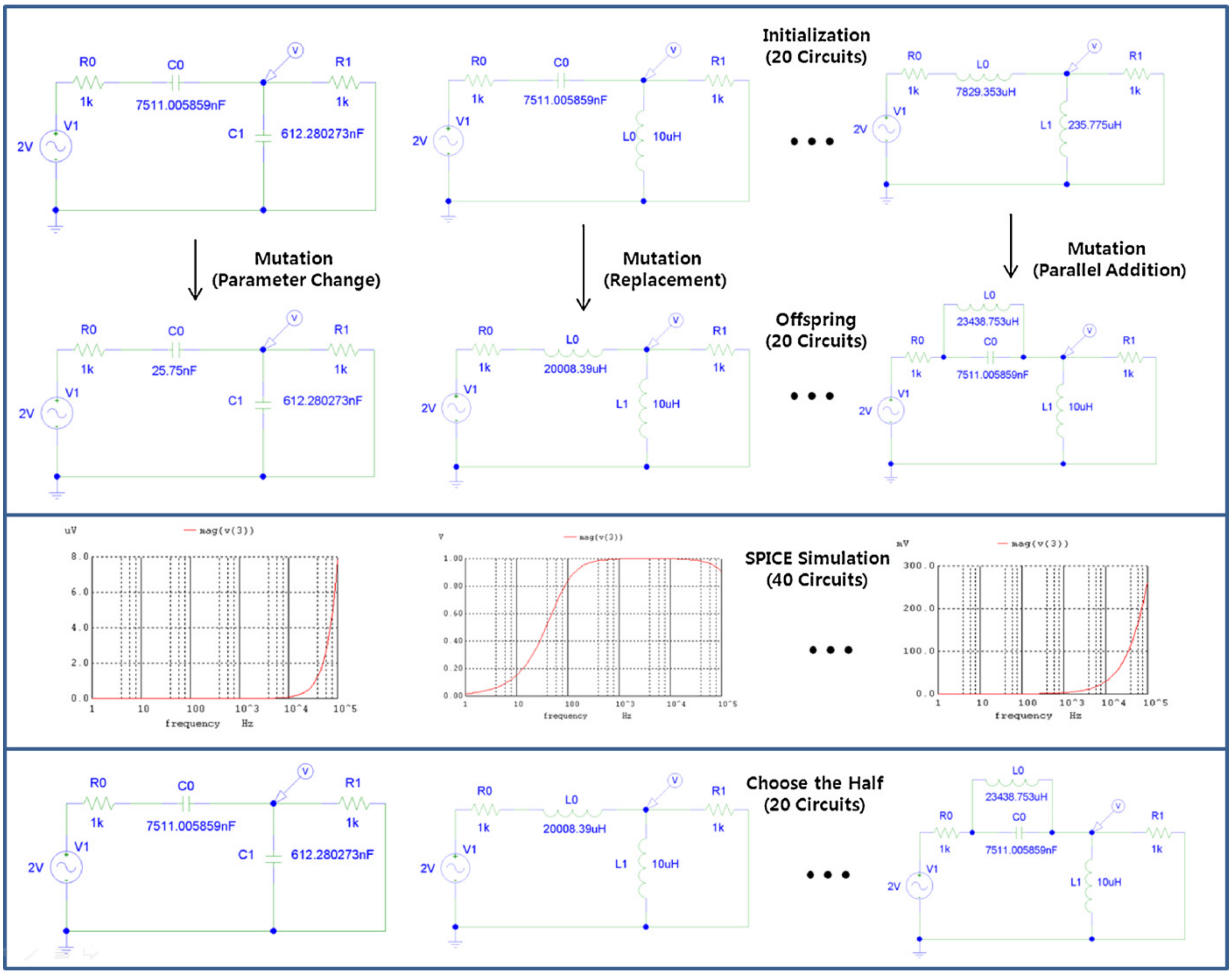


Fig. 3. Illustration of evolutionary process (initialization, mutation, evaluation and selection).

generate basic analog circuits, adopt differnent strategy to choose members for the ensemble and combine them in a different way.

Recently, Kim et al. introduced the concept to use evolutionary computation to generate multiple redundant analog circuits and combine them to increase fault-tolerance [23]. In their work, they did not incorporate the robustness of circuits in their fitness function of evolution. This enables the quick convergence of evolution and the discovery of high-quality analog circuits. After the evolution, the next step is to find the best ensemble of evolved circuits that is robust to faults. In this paper, we extended the previous work by applying our method to several other tasks and improving the ensemble search process.

3. Proposed method

The basic idea of the proposed method involved using multiple redundant circuits with the summing circuit. Each circuit performed the same function and all are active. The circuits are evolved using evolutionary computation and it is assumed that they all had different architectures. Fig. 2 shows an overview of the proposed method. Important issues included the number of evolved circuits for the ensemble system, the way of evolving each member circuit, and the implementation of the summing circuit.

Among all possible ensembles of the *P* circuits (*P* is population size), the one with the highest fault-tolerance is selected. The summing circuit is implemented by using an operational amplifier component. In this paper, there are 20 evolved circuits in the last generation of the evolution. If we choose 3 circuits from them for the ensemble, there are ${}_{20}C_3$ (1140) possibilities. The best ensemble is chosen from the exhaustive search of the 1140 possibilities based on the fault-tolerance level of each ensemble.

3.1. *Evolving analog circuits*

Because the analog circuits have numerous topologies with real-valued parameters, it is not a trivial task to evolve them. Genetic programming has been widely used to evolve analog circuits for computational circuits [14], CMOS amplifier [15], and filters [16]. Because of the complexity of GP, there are several researchers to evolve analog circuits using genetic algorithm [17–19]. Especially, Kim et al. and Gho and Li considered the practical implementation of evolved circuits from the simulation [4,17]. Recently, there are new algorithms to design analog circuits by two-layer genetic programming [21] and immune programming [22].

Genetic programming encodes the operations that are applied to the embryonic circuits which is a starting point of the circuit development. Although GP has been successful to evolve analog

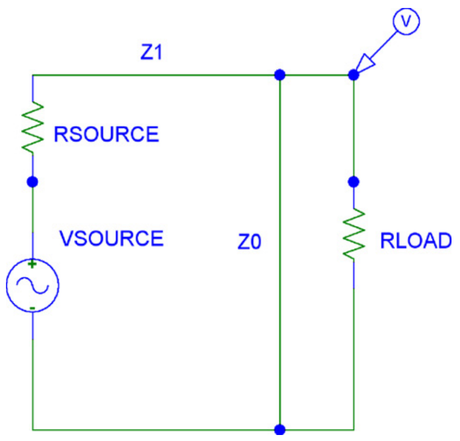


Fig. 4. An embryonic circuit for filter evolution (Z0 and Z1 are modifiable wire).

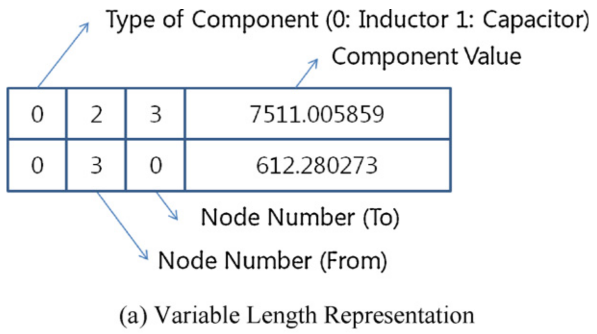
circuits for many different tasks, they require large population size (for example, 32,000) and implementation is not easy. There are many approaches to use genetic algorithm to evolve analog circuits [18]. Because they use linear representation of analog circuit, it is relatively simple to implement. However, the population size is still large (18,000).

In this work, we adopted the evolution strategy approach that uses only mutation to produce offspring with relatively small population size (20) [4]. It has been successfully used to evolve analog circuits that are fault-tolerable to partial short and disconnection damage. Also, they can generate analog circuits that are transferable to physical circuits with real components. Unlike GA with roulette wheel selection, it adopts tournament-based selection methods to increase the diversity of population. Because it uses only mutations, it is relatively easy to implement and the representation of circuit is simple (see Fig. 3).

Step (1): Initialization: Initially, P analog circuits are randomly generated. P is a population size. The embryonic circuit refers to the starting point of the circuit development and it has voltage source and fixed resistors (Fig. 4). Each modifiable wire is replaced with one new component. There are two types of components: inductor (L) and capacitor (C). Their type and parameter values are randomly determined. Fig. 5 shows an example of variable length representation of the analog circuit.

Step (2): Mutations (new P offspring): One component is randomly selected and one of eight different mutations is applied to the component. The mutations are parameter change, type change, parallel addition of a different type component, serial addition of a different type component, component deletion, ground setting, replacement, and adding a component. Mutations are as follows (see Fig. 6).

- Parameter change: The component's value is assigned as a new randomly chosen value.
- Type change: The component type is swapped to a different one randomly.
- Parallel addition of a different type component: A new component (with a different type) is added in parallel configuration to the component. The type and value of the new component is randomly chosen.
- Serial addition of a different type component: Same as above except the addition in serial configuration.
- Component deletion: The component is removed from the circuit.
- Ground setting: The component is connected to the ground.
- Replacement: The component is replaced with a new component (possibly of the same type).
- Adding a component: A new component bridges between two randomly chosen wires (not identical wire).


Type of Component (0: Inductor 1: Capacitor)
Component Value
Node Number (To)
Node Number (From)
(a) Variable Length Representation

0	2	3	7511.005859
0	3	0	612.280273

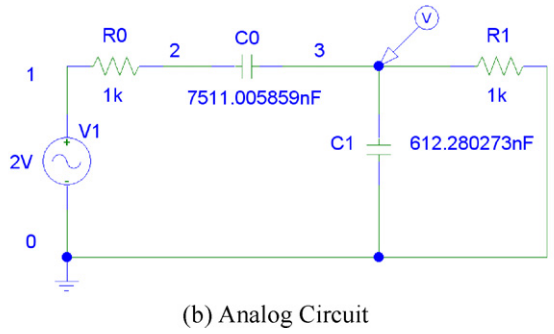


Fig. 5. Representation of individual (analog circuit) in evolution strategy (numbers assigned to the wire are node number).

Step (3): Circuit simplification: It combines identical components in a serial or parallel configuration into a single component. It maintains the circuit size as small as possible.

Step (4): Fitness evaluation with a spice simulator [29] (fitness is defined based on the similarity between actual and desired outputs). From 1 Hz to 100 kHz, the SPICE simulator performed an AC small signal analysis. The frequency area is divided into five decades and each decade is further divided into 20 parts (on a logarithmic scale). Finally, the simulator checked the voltage of 101 points (61 points below 1 kHz, 5 points between 1 kHz and 2 kHz, and 35 points above 2 kHz). The fitness is calculated as follows.

$$Fitness = \frac{1}{Error} \quad Error = \sum_{i=0}^{100} W(d(f_i), f_i) \times d(f_i)$$

The fitness value is summed over 101 points. In the above equation, f_i represents the frequency of the ith point, d represents the difference between the target and observed values at the frequency f_i, and W represents the weight for the difference at the frequency f_i (based on [14]). If circuits could not be simulated in the SPICE program, the fitness of these circuits is 0.

Step (5): Selection (P individuals from 2P pool): The best P circuits are selected from 2P individuals (parents + offspring).

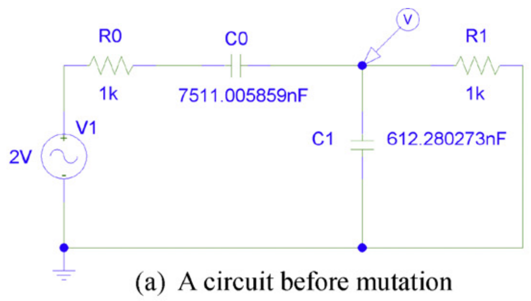
Step (6): Termination: It stops when the number of generation is larger than the maximum number of generation.

3.2. Ensemble search for fault-tolerance

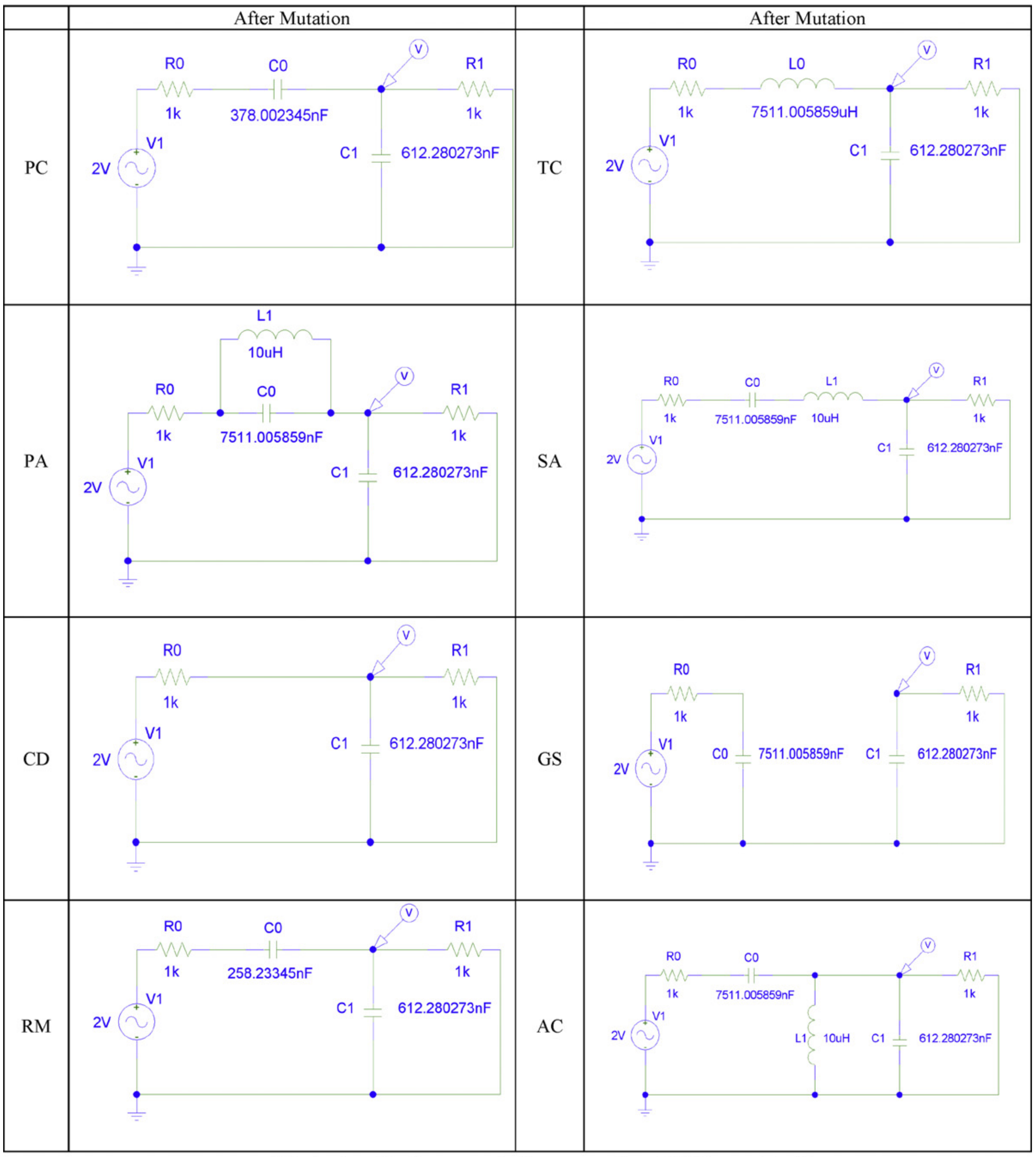
The circuits are combined using weighted summing circuits. Fig. 7 shows a weighted summing circuit [30]. Input voltages are defined as $v_1, v_2, \ldots, v_n$. The output voltage v_0 is defined as follows.

$$v_0 = -\left(\frac{R_f}{R_1}v_1 + \frac{R_f}{R_2}v_2 + \ldots + \frac{R_f}{R_n}v_n\right)$$

The weights of each input voltage are adjusted using the resistors. If the resistor for each input voltage is the same as R_f, the



(a) A circuit before mutation


After Mutation
After Mutation
PC
R0 1k C0 378.002345nF R1 1k V1 2V C1 612.280273nF
TC
R0 1k L0 7511.005859uH R1 1k V1 2V C1 612.280273nF
PA
L1 10uH R0 1k C0 7511.005859nF R1 1k V1 2V C1 612.280273nF
SA
R0 1k C0 7511.005859nF L1 10uH R1 1k V1 2V C1 612.280273nF
CD
R0 1k R1 1k V1 2V C1 612.280273nF
GS
R0 1k R1 1k V1 2V C0 7511.005859nF C1 612.280273nF
RM
R0 1k C0 258.23345nF R1 1k V1 2V C1 612.280273nF
AC
R0 1k C0 7511.005859nF R1 1k V1 2V L1 10uH C1 612.280273nF

(b) A circuit after a mutation operation (C0 is mutated)

Fig. 6. Mutation examples (PC, parameter change; PA, parallel addition; CD, component deletion; RM, replacement; TC, type change, SA, serial addition; GS, ground setting, and AC, adding a component).

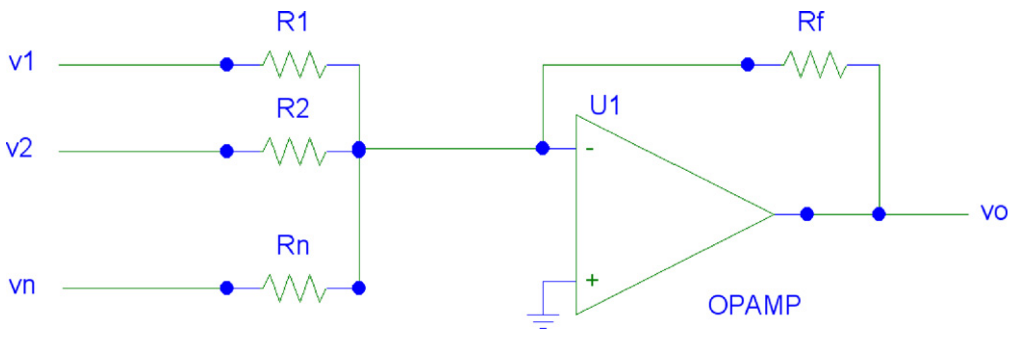


Fig. 7. A weighted summing circuit.

weight is 1. If the weight is 1 for all input voltages, it is the average of the input voltages. In this work, the average method is used.

It searches for the best fault-tolerant ensembles of circuits from the last generation. Among the P circuits, we choose 3 circuits for an ensemble and there are ${}_PC_3$ possibilities. The exhaustive search is used to find the best fault-tolerant ensemble from them. The fault-tolerance level of each ensemble circuit is represented by ft. The number of components (except the fixed resistors and voltage source) of the ensemble circuit is defined as M. The error of the circuit by removing the ith component is defined as $error(i)$. Removing a component results in an open circuit.

$$ft\_avg = \frac{1}{(1/M)\sum_{i=1}^{M} error(i)} \times 100$$

Although the fault-tolerance measure is reasonable, it requires huge number of simulations to find the best ensemble. For example, if the average number of components of ensemble is M, we need ${}_nC_3M$ simulations (see Fig. 8).

In this paper, we proposed new definition of fault-tolerance based on worst case analysis to accelerate the ensemble search (Fig. 9). The fault-tolerance is defined only based on the worst performance of the ensemble circuit.

$$ft\_worst = \frac{1}{\max_{i=1,\ldots,M}(error(i))} \times 100$$

Instead of simulating all component failure cases, if there is no possibility that the ensemble can defeat the current best one, it skips the remaining testing. In this way, we can speed up the ensemble search process.

In this paper, we proposed multi-population approach to use the results of multiple runs as a source of the fault-tolerant ensemble search (Fig. 10). It is expected that the circuits from multiple runs are structurally more different than those from the single run. First of all, we repeat the evolutionary run multiple times. From the results of the evolution, we choose one best circuit from each run based on their fitness. For example, if the number of runs is thirty,

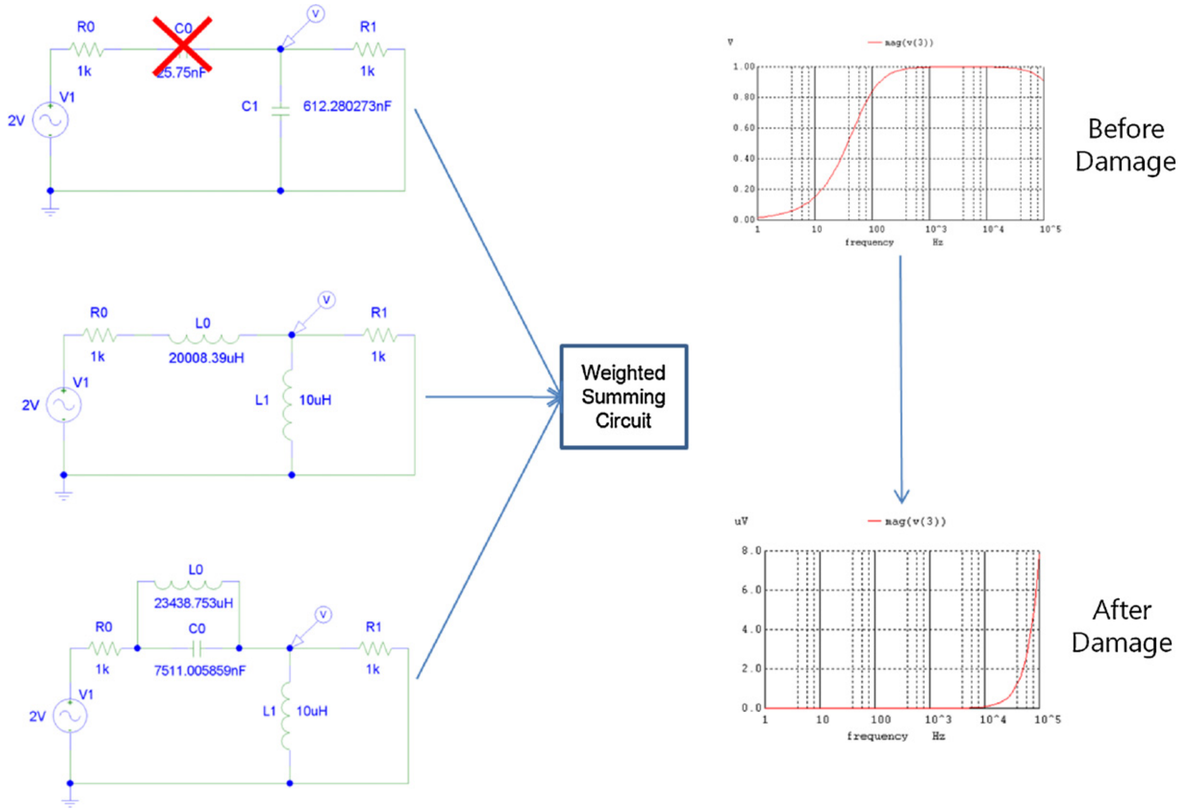


Fig. 8. An example of fault-tolerance evaluation of an ensemble (at each time, one component is removed from the ensemble and performance is measured. After testing all removals, the fault-tolerance of the ensemble is measured based on the average (*ft_avg*) or worst (*ft_worst*) case analysis.).

```
// Find an ensemble with the highest $ft\_worst$
// $ft\_worst\_global$ : The current maximum $ft\_worst$
$ft\_worst\_global = 0$
For each ${}_nC_3$ possible ensemble circuits {
    if(identical circuits in the ensemble exist) continue;
$ft\_worst = 0$
    For each component from the circuits in the ensemble{
        Remove ith Component and Do Simulation Using Spice
        if ($ft\_worst < \frac{1}{error(i)} \times 100$)  $ft\_worst = \frac{1}{error(i)} \times 100$
    if ($ft\_worst < ft\_worst\_global$) Break Loop                // Speed Up
    }
    if ($ft\_worst > ft\_worst\_global$) $ft\_worst\_global = ft\_worst$ }
```

Fig. 9. A pseudo code of the accelerated ensemble search.

there are thirty analog circuits. From the thirty analog circuits, the ensemble search algorithm finds the best fault-tolerant ensemble.

4. Experimental results and discussion

4.1. Evolving analog circuits

The initial circuits are generated randomly and their structures are dependent on the architecture of the embryonic circuit. In this paper, the embryonic circuit contained two modifiable wires and fitness is evaluated using a SPICE simulator. The developed circuit is converted into NETLIST, an input file for the SPICE program (Fig. 11). The targets of evolution are low-pass, and high-pass filters with an one-input, one-output circuit composed of capacitors and inductors. Table 3 summarizes the desired outputs for the three different tasks. There are several parameters for these experiments (Table 4).

In this work, we have used WinSpice 1.05.07 by OuseTech (downloadable as a binary executable from their website). Our program generated an input file for the WinSpice and executed it using a "system" function in C language. Each input file describes circuit's topology and component's values together with analysis commands for the SPICE. As a result, the SPICE produced an output file containing the output voltage of an analog circuit over several different input frequencies ranged from 1 Hz to 100 kHz. If the output response of the circuit is similar to the one that we are looking for, it gets high fitness value (reward) from the search algorithm. For example, if the purpose of our work is to find a low-pass filter circuit, the desired output might be 1 V in low frequency

Table 3
Desired output response of three different filters.

	Desired outputs		
	From	To	Output
Low-pass filter	0	1 kHz	1 V
	1 kHz	2 kHz	Don't care
	2 kHz	∞	0 V
High-pass filter	0	1 kHz	0 V
	1 kHz	2 kHz	Don't care
	2 kHz	∞	1 V
Band-stop filter	0	100 Hz	1 V
	100 Hz	10 KHz	0 V
	10 KHz	∞	1 V

Table 4
Parameters of experimentations (node means any point on a circuit where two or more circuit elements meet).

Embryonic circuit	Fixed: VSOURCE, RSOURCE, RLOAD modifiable: Z0 and Z1 wires
Voltage source	2 V
RSOURCE and RLOAD	1 kΩ
SPICE simulator	WinSpice (1.05.07) by OuseTech [29] file I/O interface in C++
Population size	20
Maximum generation	300
Node number	0–10
Component value	Inductor: 0.1–10^5, capacitor: 1–10^5
# of runs	30

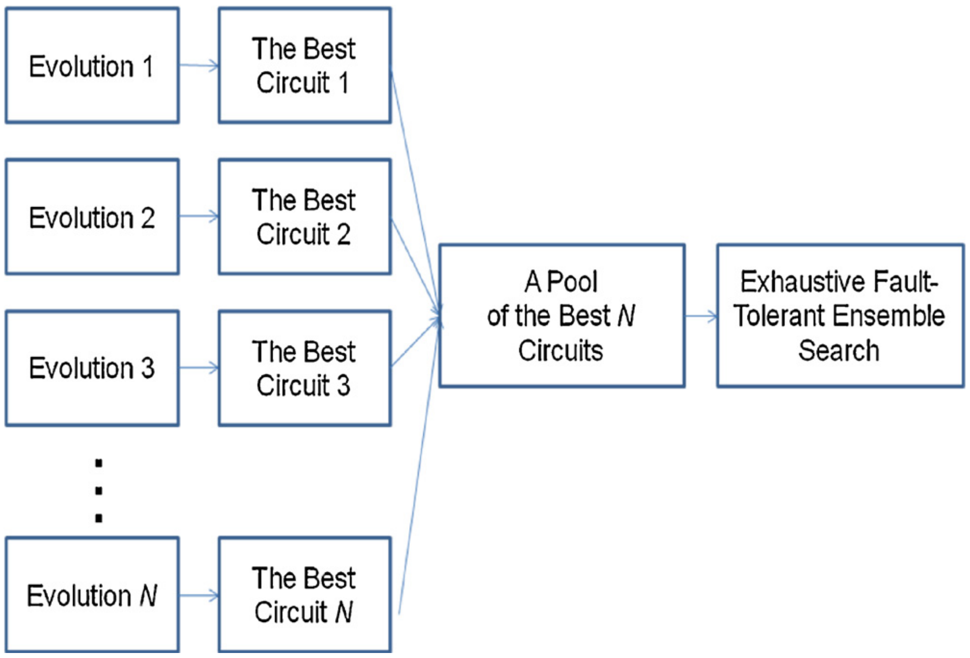
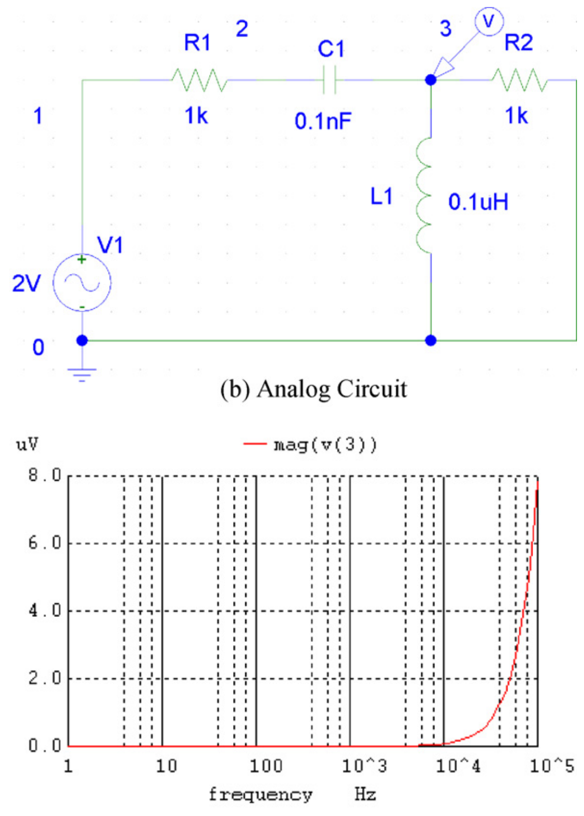


Fig. 10. Multi-population approach.

```
LOW-PASS FILTER
VS 0 1 AC 2.0
R1 1 2 1.0K
R2 3 0 1.0K
C1 2 3 0.1nF
L1 3 0 0.1uH
.AC DEC 20 1 100K
.PRINT AC V(3)
.PROBE
.END
```



(a) NETLIST (c) Output Response

Fig. 11. An example of analog circuit represented as NETLIST and its output response (in the NETLIST, "R1 1 2 1.0K" means that the component R1 is located between node 1 and node 2 with 1.0 K value).

area and 0 V in high frequency area. The desired output is a target/ideal response which we are expecting from our analog circuits automatically designed.

For comparison, fault-tolerant evolution (FT evolution) is used. In the evolution, the fitness function is *ft_worst*. Because it requires many simulations for each circuit to get the fitness value, it is much more time consuming than the normal evolution. Unlike the normal evolution, the FT evolution can be trapped into local optima in the early stage of evolution if it uses the same embryonic circuit. For example, if a component in Z1 wire is removed (Fig. 4), the *ft_worst* might be the lowest one. This is the same situation for all circuits in the early stage of evolution because it starts from an empty circuit and growing the circuit continuously. To solve this problem, a new embryonic circuit is proposed for the FT evolution (Fig. 12). It has two parallel wires between the R0 and the R1.

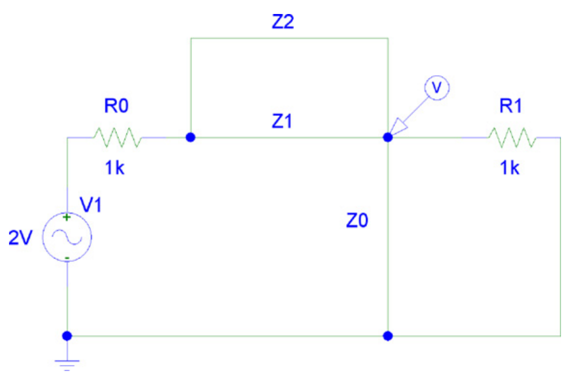


Fig. 12. An embryonic circuit for FT evolution.

The results of the evolution are as follows. Fig. 13 shows the progress of the evolution showing the error of the best circuit over generations. Although the high-pass filter's desired output is just opposite to that of low-pass filter. The progress pattern is a bit different for the two tasks. For low-pass filter, it gradually improved but the high-pass filter showed a bit radical improvement. Fig. 14 shows the best circuit's diagram and output response. They have relatively small number of components (8–10 components) to do their tasks.

4.2. Ensemble search for fault-tolerance

The most important thing in our approach is to maintain different analog circuits in the last generation. This minimizes the probability that one or more circuits fail simultaneously in the ensemble. Fig. 15 shows the change of diversity (the number of unique analog circuit in the population) over generation. It is natural that the diversity of population goes down because of the domination of successful solutions. The tournament selection prohibits quick dominance of the most successful one unlike roulette-wheel selection. By ignoring the ensemble with identical members, its search space can be reduced. We compared NETLIST of two circuits to decide whether they are the same circuit or not.

Table 5 summarizes the error and fault-tolerance of both approaches. The ensemble showed an acceptable fitness performance and the best for the fault-tolerance measure. The fault-tolerance level showed significant improvement compared to the single circuit. With regard to the fault-tolerance level, every ensemble with four circuits performed better than the single circuit. In the fault-tolerance evolution, the fault-tolerance level of

Table 5
Summary of statistics (average of 30 runs, standard error).

	Single best circuit	Ensemble (4 circuits)	Multi-population ensemble (4 circuits)	FT evolution[a]
Low-pass filter				
Error (↓)	4.896 ± 0.445	4.877 ± 0.448	**3.401**	113.352 ± 9.301
ft_avg (↑)	0.949 ± 0.088	5.935 ± 0.698	**11.982**	1.174 ± 0.075
ft_worst (↑)	0.136 ± 0.011	0.532 ± 0.044	0.642	**0.932 ± 0.043**
High-pass filter				
Error (↓)	5.376 ± 0.923	5.403 ± 0.924	**1.368**	80.634 ± 8.733
ft_avg (↑)	1.357 ± 0.140	6.776 ± 0.728	**9.408**	1.641 ± 0.044
ft_worst (↑)	0.324 ± 0.049	1.263 ± 0.147	1.118	**1.620 ± 0.046**
Band-stop filter				
Error (↓)	122.109 ± 2.892	122.589 ± 2.834	**99.992**	226.771 ± 12.083
ft_avg (↑)	0.322 ± 0.028	0.649 ± 0.022	**0.924**	0.437 ± 0.020
ft_worst (↑)	0.171 ± 0.014	0.468 ± 0.009	**0.538**	0.388 ± 0.018

The best obtained result for each measure is highlighted in bold font.

[a] The fitness function in the FT evolution is similar to the one used in [19]. It simulates each circuit once with each circuit component removed. Worst case analysis is used to measure the robustness of the circuit from these runs.

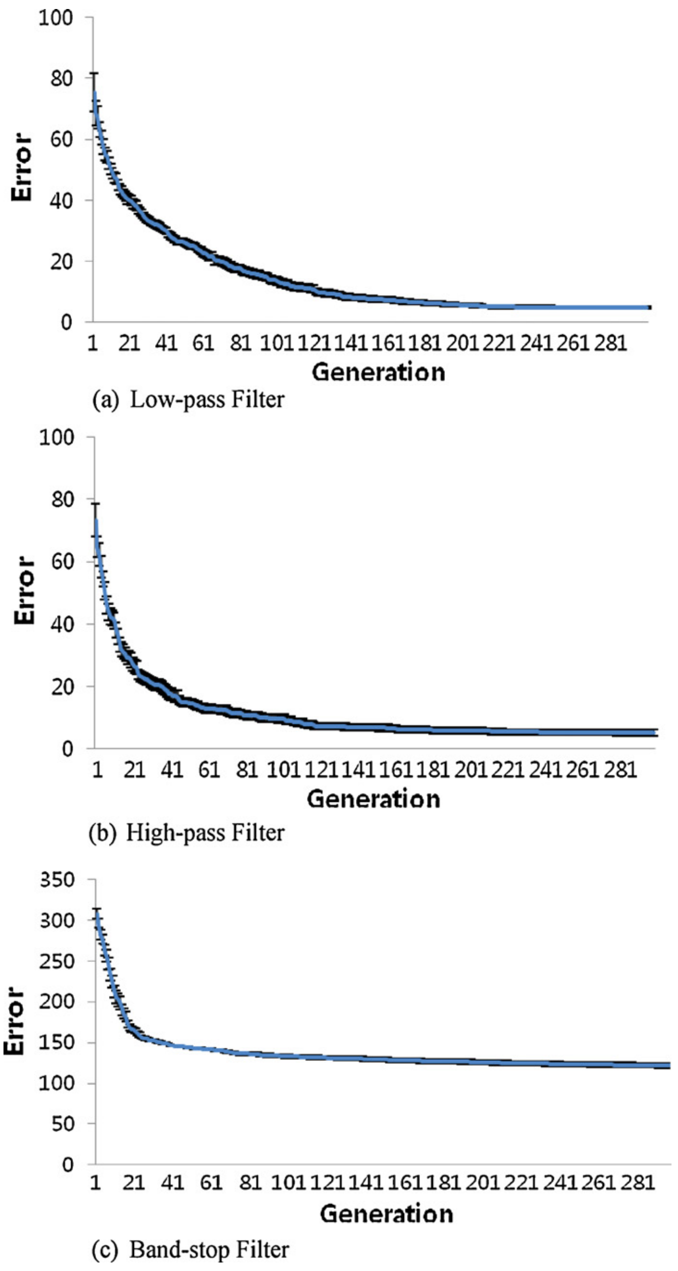


Fig. 13. The error of the best circuits over generation (averaged over 30 runs, standard error bar is included).

each circuit is used as a fitness function. It showed that the fault-tolerance evolution fails to produce successful analog circuits.

In case of error and *ft_avg*, multi-population ensemble is the best among several alternatives. The ensemble approach always outperforms the single best circuit. FT evolution produces the best *ft_worst* circuits for low-pass and high-pass filters. However, the circuits from the FT evolution show poor performance if there is no damage.

Table 6 shows the acceleration of exhaustive ensemble search. By ignoring the ensembles with the identical analog circuits, it significantly reduces the search space. Also, it is possible to minimize the number of evaluations by introducing the *ft_worst* measure. Gain ratio is about 15–30. In case of ensembles with 2 members (low-pass filter), the new acceleration method needs 40 s to find the best ensemble, however, the method without the acceleration records 382 s. The new method is about 10 times faster than the previous one.

Table 7 shows the comparison with negatively correlated ensemble of evolved analog circuits proposed by Liu and He [34]. In their approach, the original fitness of each circuit is modified by adding the negatively correlated penalty (the $\lambda = 0.5$). The best circuit from each run of negatively correlated evolution is combined to produce the final outcome. The evolved circuits from the approach are good when there is no fault. However, their averaged fault-tolerance value is smaller than the one from our approach.

4.3. Discussion

Digital circuit's output is binary and it is possible to combine their outputs using majority voting method. However, this is not

Table 6
Comparison of computational cost.

	# of SPICE simulation without acceleration (A)	# of SPICE simulation with acceleration (B)	Gain (A/B)
Low-pass filter	23,402 ± 9467	1604 ± 416	14.58
High-pass filter	36,438 ± 10,257	2431 ± 778	14.98
Band-stop filter	93,574 ± 27,606	2778 ± 703	33.68

Table 7
Comparison with previous work [34] (low-pass filter).

	Ensemble (4 circuits)	Multi-population ensemble (4 circuits)	Negatively correlated ensemble [34] (4 circuits)
Error (↓)	4.877 ± 0.448	**3.401**	3.608
ft_avg (↑)	5.935 ± 0.698	**11.982**	3.396
ft_worst (↑)	0.532 ± 0.044	**0.642**	**0.642**

The best obtained result for each measure is highlighted in bold font.

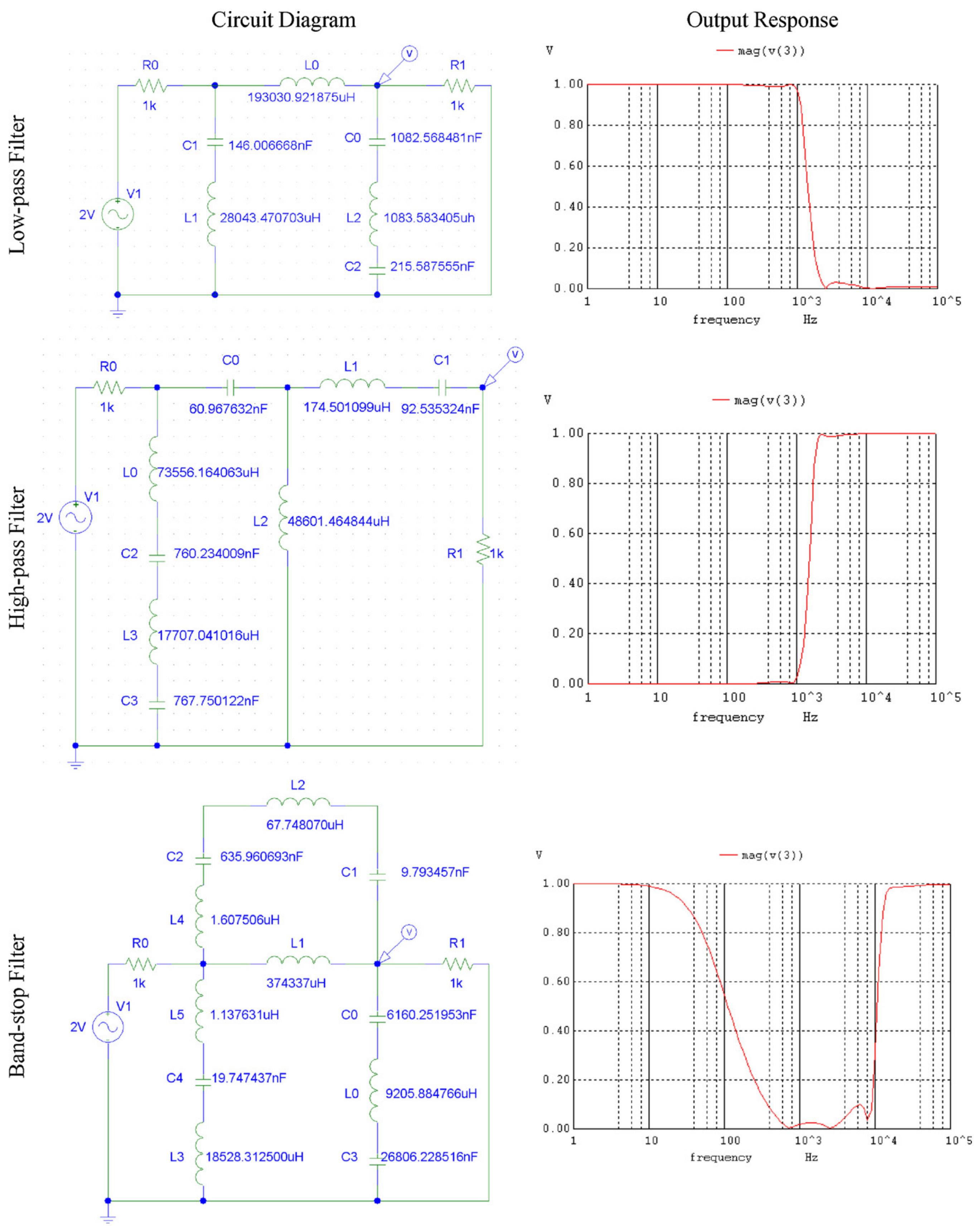
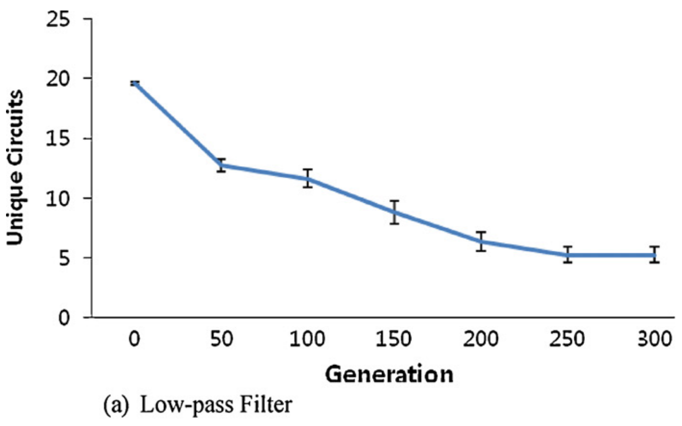
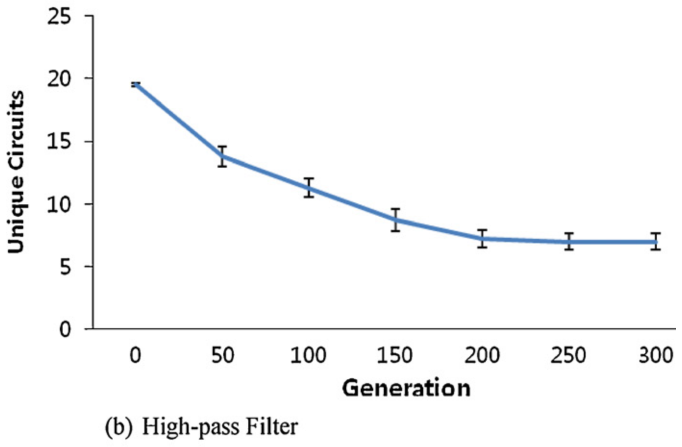


Fig. 14. Circuit diagram and output response of the best circuits.





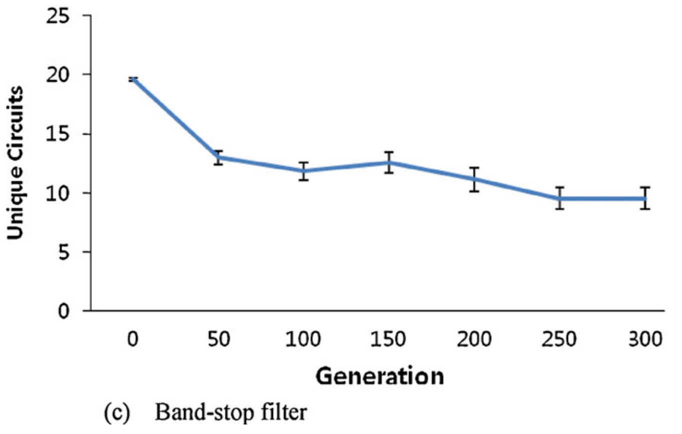


Fig. 15. The diversity of population goes down over generation.

easy for analog circuits. It is the reason that we choose the weighted sum method to combine multiple analog circuits.

In this work, we use the evolutionary algorithm with eight mutations and tournament selection. Although this mechanism can generate different analog circuits, the difference between their parents and the children is not big because only mutation operator is used. It is interesting to invent crossover operations for analog circuits.

The size of ensemble is one of the important parameters in our experimentations. In our work, we use four members for ensembles. It is based on the initial investigation on low-pass filters (Fig. 16). If possible, it is better to minimize the ensemble size because the number of components in the ensemble is proportionate to the size of ensemble.

Optimizing analog circuit is not a trivial problem because it has to optimize both topology and values of components. Because the genetic algorithm maintains multiple candidates at the same time, it has strong global search capability. For comparison, we implemented a simple greedy-style optimization algorithm using the eight mutation operators proposed in this paper. At first, one random circuit is generated. If the number of components in the random circuit is two, there are total 16 possible mutations (two components $\times$ eight mutations). Among them, we choose the mutation which improves the performance best. This process continues until the maximum number of SPICE simulation reaches. For comparison, we ran the greedy-style algorithm for the low-pass filter with the same computational constraints. The error rate of the final circuit by the greedy algorithm is 19.565914 and about five times worse than the evolutionary algorithms used in this paper.

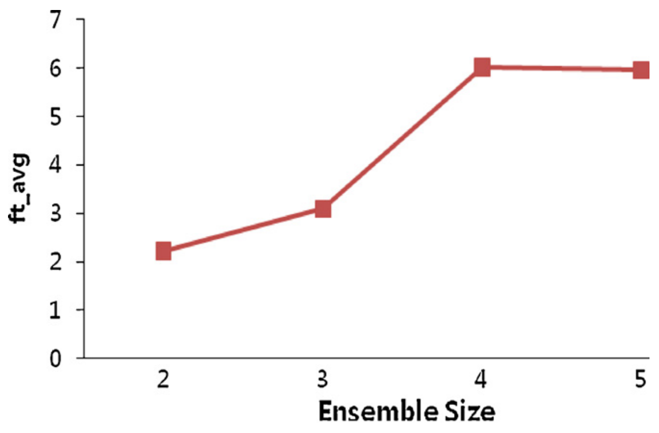


Fig. 16. The relationship between the ensemble size and the fault-tolerance (low-pass filter).

5. Conclusions and future works

In this work, we have proposed an ensemble of evolved circuits with a weighted summing circuit. It is shown that the combination of the analog circuits evolved performed well when removing components. Furthermore, the ensemble also performed quite well when there is no component failure. In general, this will allow designers to make fault-tolerant redundant circuits automatically. For diversity, tournament selection is used [32]. Because we use the exhaustive search for ensemble optimization, time complexity is one of the important issues. In this paper, we enhanced the search speed by ignoring ensembles with identical members and adopting fault-tolerance based on the worst case analysis.

Although we can generate different analog circuits with the tournament selection and mutation operators, their difference is not big as expected. It is more convenient and natural to use speciation and niching methods for evolution [33]. For this purpose, in future work we will devise a method for measuring the similarities between two circuits in the phenotype and genotype levels. Also, the weights of each redundant circuit can be adjusted according to characteristic of the circuit.

Acknowledgements

This research was supported by Basic Science Research Program and the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0012876, 2010-0018948). This work was supported by Prof. Hod Lipson (Cornell University).

References

[1] P. Groot, A.T. Teije, F.V. Harmelen, A quantitative analysis of the robustness of knowledge-based systems through degradation studies, Knowledge and Information Systems 7 (2005) 224–245.
[2] R. Isermann, Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance, Springer, 2006.
[3] R.J. Patton, Fault detection and diagnosis in aerospace systems using analytical redundancy, Computing & Control Engineering 2 (1991) 127–136.
[4] K.J. Kim, A. Wong, H. Lipson, Automated synthesis of resilient and tamper-evident analog circuits without a single point of failure, Genetic Programming and Evolvable Machines 11 (2009) 35–59.
[5] J. Bongard, V. Zykov, H. Lipson, Resilient machines through continuous self-modeling, Science 314 (2006) 1118–1121.
[6] P. Kabisatpathy, A. Barua, Fault Diagnosis of Analog Integrated Circuits, Springer, 2005.
[7] A.E. Akadi, A. Amine, A.E. Ouardighi, D. Aboutajdine, A two-stage gene selection scheme utilizing MRMR filter and GA wrapper, Knowledge and Information Systems 26 (2011) 487–500.
[8] W. Song, S.C. Park, Latent semantic analysis for vector space expansion and fuzzy logic-based genetic clustering, Knowledge and Information Systems 22 (2010) 347–369.
[9] J. Martinez-Gil, F. Aldana-Montes, Evaluation of two approaches to solve the ontology meta-matching problem, Knowledge and Information Systems 26 (2011) 225–247.
[10] P. Kouchakpour, A. Zaknich, T. Braunl, A survey and taxonomy of performance improvement of canonical genetic programming, Knowledge and Information Systems 21 (2009) 1–39.
[11] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[12] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[13] T. Back, F. Hoffmeister, H.P. Schwefel, A survey of evolutionary strategies, in: Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 2–9.
[14] J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, F. Dunlap, Automated synthesis of analog electrical circuits by means of genetic programming, IEEE Transactions on Evolutionary Computation 1 (1997) 109–128.
[15] T. Sripramong, C. Toumazou, The invention of CMOS amplifiers using genetic programming and current-flow analysis, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 21 (2002) 1237–1252.
[16] J. Hu, X. Zhong, E.D. Goodman, Open-ended robust design of analog filters using genetic programming, Proceedings of the 2005 Conference on Genetic and Evolutionary Computation 2 (2005) 1619–1626.
[17] C. Goh, Y. Li, GA automated design and synthesis of analog circuits with practical constraints, Proceedings of the 2001 Congress on Evolutionary Computation 1 (2001) 170–177.
[18] J.D. Lohn, S. Colombano, Automated analog circuit synthesis using a linear representation, Lecture Notes in Computer Science 1478 (1998) 125–133.
[19] G.A. Hollinger, D.A. Gwaltney, Evolutionary design of fault-tolerant analog control for a piezoelectric pipe-crawling robot, in: Proceedings of the 2006 Conference on Genetic and Evolutionary Computation, 2006, pp. 761–768.
[20] T. Back, D.B. Fogel, Z. Michalewicz, Evolutionary Computation 1: Basic Algorithms and Operators, Taylor & Francis, 2000.
[21] F. Wang, Y. Li, L. Li, K. Li, Automated analog circuit design using two-layer genetic programming, Applied Mathematics and Computation 185 (2007) 1087–1097.
[22] A. Ciccazzo, P. Conca, G. Nicosia, G. Stracquadanio, An advanced clonal selection algorithm with Ad-hoc network-based hypermutation operators for synthesis of topology and sizing of analog electrical circuits, Lecture Notes in Computer Science 5132 (2008) 60–70.
[23] K.J. Kim, S.B. Cho, Combining multiple evolved analog circuits for robust evolvable hardware, in: 10th International Conference on Intelligent and Data Engineering and Automated Learning (IDEAL), 2009, pp. 359–367.
[24] D. Keymeulen, R.S. Zebulum, Y. Jin, A. Stoica, Fault-tolerant evolvable hardware using field-programmable transistor arrays, IEEE Transactions on Reliability 49 (2000) 305–316.
[25] R.S. Zebulum, M.A. Pacheco, M. Vellasco, H.T. Sinohara, Evolvable hardware: on the automatic synthesis of analog control systems, Proceedings of IEEE Aerospace Conference 5 (2000) 451–463.
[26] R.S. Zebulum, A. Stoica, D. Keymeulen, L. Sekanina, R. Ramesham, X. Guo, Evolvable hardware systems at extreme low temperatures, Lecture Notes in Computer Science 3637 (2005) 37–45.
[27] S. Ando, H. Iba, Analog circuit design with variable length chromosomes, Proceedings of the 2000 Congress on Evolutionary Computation 2 (2000) 994–1001.
[28] P. Layzell, A. Thompson, Understanding inherent qualities of evolved circuits: evolutionary history as a predictor of fault tolerance, Lecture Notes in Computer Science 1801 (2000) 133–144 (2000).
[29] WinSpice, http://www.winspice.com/.
[30] A.S. Sedra, K.C. Smith, Microelectronic Circuits, fifth ed., Oxford University Press, 2004.
[31] T. Schnier, X. Yao, Using negative correlation to evolve fault-tolerant circuits, in: Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware, 2003, pp. 35–46.
[32] B.L. Miller, D.E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex Systems 9 (1995) 193–212.
[33] B. Sareni, L. Krahenbuhl, Fitness sharing and niching methods revisited, IEEE Transactions on Evolutionary Computation 2 (1998) 97–105.
[34] M. Liu, J. He, Negatively correlated redundancy circuits evolution: a novel way of robust analog circuit synthesizing, in: Proceedings of the Third International Workshop on Advanced Computational Intelligence (2010), 2010, pp. 496–501.
[35] G. Greenwood, M. Joshi, Evolving fault tolerant digital circuitry: comparing population-based and correlation-based methods, IEEE Congress on Evolutionary Computation (2009) 2796–2801.
[36] M. Hartmann, P.C. Haddow, Evolution of fault-tolerant and noise-robust digital designs, IEE Proceedings Computers and Digital Techniques 151 (2004) 287–294.