GAPSYS: A GA-based Tool for Automated Passive Analog Circuit Synthesis

Angan Das and Ranga Vemuri
Department of Electrical and Computer Engineering, University of Cincinnati, Cincinnati, OH 45221-0030, USA
Email: {dasan, ranga}@ece.uc.edu

Abstract— **This paper presents GAPSYS - a genetic algorithm based automated circuit synthesis tool for passive analog circuits. It describes the procedure for developing both the circuit topology and the component values for a passive analog circuit comprising of R, L and C components from a given set of specifications. The novelty of the work pertains to the component value assignment procedure for the initial set of circuits and the crossover techniques employed. Experiments conducted on two low-pass filter design benchmarks demonstrate the effectiveness of GAPSYS as a synthesis tool.**

I. INTRODUCTION

Circuit synthesis is defined as the process of constructing a circuit or network that satisfies the desired performance specifications. This essentially comprises of two sequential steps viz. Topology selection and Circuit sizing. While the former depends heavily on expert analog designers, the latter suffers from the drawbacks of either a high setup effort (for knowledge based methodology) or that of a three-fold trade-off between speed, accuracy and global convergence (for numerical optimization based methodology). Hence, there is a need for a flexible, automated circuit synthesis framework that can produce both the topology and component values, but with minimal design expertise and reasonable computational effort. This kind of objective is well achieved through evolutionary search techniques like Genetic Algorithms (GA) and Genetic Programming (GP). Both of them evolve and maintain a set of possible solutions in a generation. Previous generations give birth to newer generations till the design goals are fulfilled.

Previous work in this context has been done by Koza *et al.* [1] and Lohn *et al.* [2], followed by Zebulum *et al.* [3], Grimbleby *et al.* [4], Goh *et al.* [5] and Chang *et al.* [6]. The common drawback in all of these works is that the initial set of circuits (i.e. those circuits belonging to the first generation) have been built by connecting components having totally random values. This introduces a huge initial gap between the desired and obtained performance metrics. It indirectly increases the subsequent number of generations required to gradually decrease this gap and produce a fully compliant circuit. Furthermore, to the best of our knowledge, in case of GAs, the crossover procedures adopted till date exchanges a handful of random elements between two parent circuits. This kind of crossover produces a lot of structurally incorrect or *pathological* circuits which cannot be simulated by a standard circuit simulator like SPICE. All of the above factors contribute to the delayed convergence for the GA.

This work was supported in part by National Science Foundation under award number CCF-0429717 and in part by Ohio Board of Regents PhD Enhancement Program.

In this paper, we present a GA-based synthesis framework, GAPSYS, for simultaneous topology generation and component sizing for passive analog circuits. Here, we alleviate all of the above shortcomings. First of all, in the first generation of circuits, we assign values to certain components through a deterministic technique, rather than all of them being random. It is ensured though that minimal design knowledge is required in the process. Secondly, the crossover procedure involves exchange of proper and well-defined subcircuits between two participating parent circuits. This maintains the structural integrity of the offspring circuits produced.

The rest of the paper is organized as follows. Section II discusses the basic formulation underlying GAPSYS. The initial circuit formulation and crossover techniques are described in details. Section III demonstrates the effectiveness of the tool through two low-pass filter design benchmarks. The conclusion is finally drawn in Section IV.

II. GAPSYS: BASIC FORMULATION

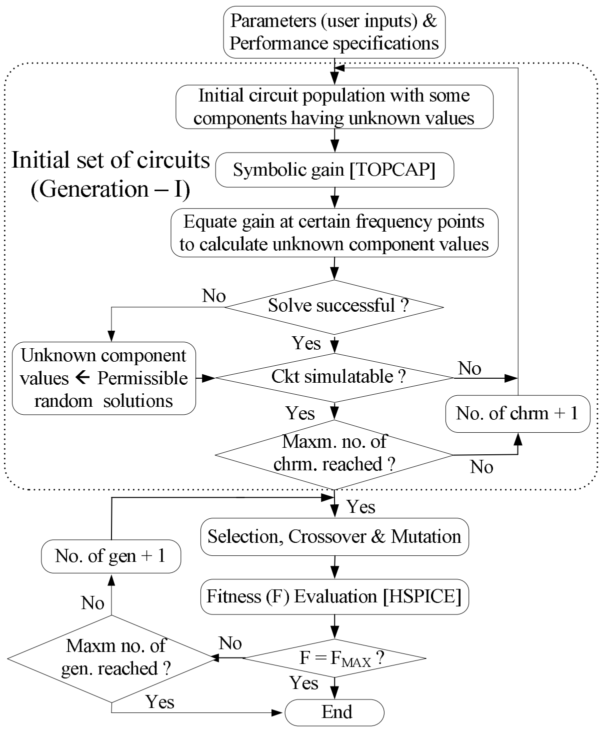
The GA-based synthesis approach is shown in Fig. 1. The user provides certain control parameters along with the required specifications. The GA builds an initial set of solutions, called chromosomes, that collectively form the first generation. Each of these chromosomes are evaluated for their fitness, which measures their degree of compliancy to the target specifications. The selection procedure selects prospective parent circuits for reproduction (crossover and mutation) of the present generation in order to produce future generations of chromosomes. This process continues either till the target is met, or until the maximum number of generations is reached, whichever occurs earlier. The individual aspects of GAPSYS are detailed in the following subsections.

A. Embryonic Circuit

The process of generating circuits by means of genetic algorithm comprises of evolving a circuit and placing it in a predefined template or framework, the *embryonic circuit*. The single-input single-output embryonic circuit [1] is shown in Fig. 2. The nodes $1, 2$ and 3 form the gateway of connection for the evolved circuit to the template circuit.

B. Circuit Representation

One of the most important aspects of any GA-based search technique is the representation of a solution. Here, a gene represents an element of a circuit and several such genes combine to form a chromosome, representing the complete circuit. The information carried by each gene comprises of:



Parameters (user inputs) & Performance specifications
Initial circuit population with some components having unknown values
Initial set of circuits (Generation – I)
Symbolic gain [TOPCAP]
Equate gain at certain frequency points to calculate unknown component values
Solve successful ?
No
Yes
Unknown component values ← Permissible random solutions
Ckt simulatable ?
No
Yes
No. of chrm + 1
Maxm. no. of chrm. reached ?
No
Yes
Selection, Crossover & Mutation
No. of gen + 1
Fitness (F) Evaluation [HSPICE]
No
Maxm no. of gen. reached ?
No
F = F_MAX ?
Yes
Yes
End

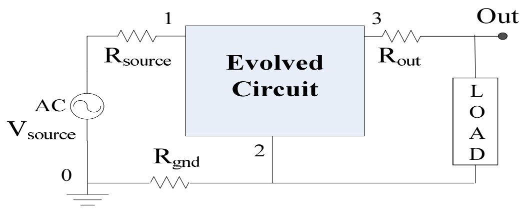
Fig. 1. GAPSYS architecture

1) *Type of element:* The three kinds of elements possible and their respective integer representations are: $0 \rightarrow$ Resistor, $1 \rightarrow$ Inductor and $2 \rightarrow$ Capacitor.
2) *Connectivity nodes:* The two circuit nodes between which the R, L and C components are connected.
3) *Value:* The numerical value of the element X lies within the range [$\mathrm{X}_{\mathrm{min}}$, $\mathrm{X}_{\mathrm{max}}$] where $\mathrm{X} \in \{\mathrm{R, L, C}\}$.

The length of the chromosome is made variable to assign flexibility to the evolvable circuit sizes. An example representation of the chromosome for parent circuit P_1 of Fig. 4 is shown in Fig. 3. It is to be noted that we do not encode $\mathrm{R}_{\mathrm{source}}$, LOAD, $\mathrm{R}_{\mathrm{gnd}}$ and $\mathrm{R}_{\mathrm{out}}$ since they belong to the common template contained in all circuits.

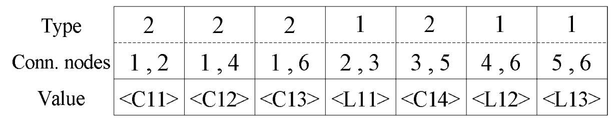
C. Initial Circuit Formulation

Most of the previous works [1, 2] greatly stress on evolving circuits with almost nil design knowledge. But the fact remains that some kind of design knowledge, however minimal, always helps in minimizing the unwanted search space for evolutionary circuits [7]. In GAPSYS, we utilize this fact



$\mathrm{V}_{\mathrm{source}}$: Source voltage, $\mathrm{R}_{\mathrm{source}}$: Finite source resistance,
$\mathrm{R}_{\mathrm{gnd}}$: Connectivity to ground, $\mathrm{R}_{\mathrm{out}}$: Connectivity to output

Fig. 2. Embryonic Circuit



Type	2	2	2	1	2	1	1
Conn. nodes	1 , 2	1 , 4	1 , 6	2 , 3	3 , 5	4 , 6	5 , 6
Value	<C11>	<C12>	<C13>	<L11>	<C14>	<L12>	<L13>

<X> → Value of element X

Fig. 3. Chromosome representing the parent circuit P_1 of Fig. 4

while assigning values to the components building the set of circuits belonging to the first generation. It is achieved by a two-step process. First, the symbolic gain of the circuit ($V_{\mathrm{out}}/V_{\mathrm{in}}$) is computed in terms of the component values as parameters. The gain expression obtained is then equated to certain desired values at specific frequency points in order to solve the unknown parameters. The procedure is schematically shown in Fig. 1. The process is repeated for non-simulatable circuits till we get a structurally correct, simulatable circuit. This ensures that all the circuits in the initial generation are *non-pathological* circuits representing valid circuit graphs.

The procedure outlined above has its own limitations like occasional failure to solve multiple equations and production of invalid solutions, apart from the computational burden involved herein. But it certainly proves effective if one judiciously chooses the number of unknown parameters involved and takes help of a good numeric solver. In extreme cases where the procedure fails to produce valid solutions, the unknowns are assigned random values in order to prevent the termination of the evolutionary process.

GAPSYS uses TOPCAP [8] as the symbolic simulator owing to its speed and ease of implementation. MATLAB® is used as the computational engine and Synopsys® HSPICE as the circuit simulator for fitness evaluation.

D. Crossover

The crossover operation selects certain portions individually from two participating parent circuits and swaps them to from two new offspring circuits. In this work, a novel crossover technique is developed comprising of the following features:

- The group of elements extracted out of a parent circuit (say P) forms a specific subcircuit (say S).
- Empty pockets are created in the two participating parent circuits (P_1 and P_2) when corresponding subcircuits (S_1 and S_2) are extracted out of them. The extraction is done in such a way that S_1 exactly fits into the empty pocket of P_2 and S_2 into that of P_1. This leads to the formation of two structurally perfect offspring circuits.

The crossover process is illustrated with an example shown in Fig. 4. The operation is performed through the following steps in sequence (explained with the help of Fig. 4).

Formation of S_1 from P_1

1) Start_node (node 6) is chosen randomly.
2) Branches connected to start_node form the subcircuit S_1. The corresponding nodes are designated as stitch_nodes (here nodes $1, 4$ and 5).
3) The subcircuit S_1 comprises of the portion encompassed by the connectivity_nodes (stitch_nodes + embryo_nodes) and internal_nodes. Embryo_nodes and internal_nodes are not applicable for S_1 in this example.

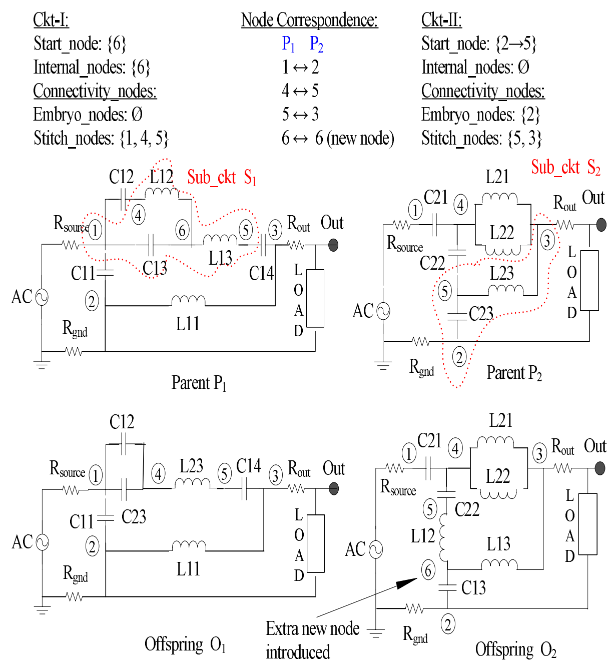


Fig. 4. Example illustrating the process of crossover

Formation of S_2 from P_2

4) Same as step-1 (node 2).
5) Same as step-2 (node 5).
6) For each stitch_node selected in step-5, it is checked if the total number of connectivity nodes of S_2 is equal to that of S_1 (no. of connectivity nodes should be same).
7) When condition is met, jump to step-11.
8) If condition is not met, start_node is *put* (status changed) either into the queue of embryo_nodes (if start_node $\in \{0, 1, 2, 3\}$) or else to that of internal_nodes (as is here). Then, we *get* the new start_node from the stitch_nodes queue (new start_node is node 5) and go back to step-5. Also, we consider only those branches that have not already been chosen for S_2.
9) If condition is not met even after all nodes have been explored, S_1 and S_2 comprise of a single element chosen randomly from their parent circuits (not applicable here).
10) Same as step-3 (S_2 is shown with dotted lines).

Offspring circuit formation

11) Node correspondence is established between S_1 and S_2. Offspring circuits to be formed may differ from their parent circuits in terms of number of nodes (extra node (node 6) for O_2).
12) Offspring circuit O_1 = remaining$(P_1) \cup S_2$
13) Offspring circuit O_2 = remaining$(P_2) \cup S_1$

E. Mutation

Mutation is a genetic operator used to avoid any local minima, that may slow or even stop evolution. It is achieved by preventing the population of chromosomes from becoming too similar to each other. The mutation process used here comprises of six different types viz. series, parallel, open circuit, short circuit, change value of element and change type of element [5].

F. Fitness

The fitness metric determines how well the generated circuit meets the given performance specifications. The user specifications for a passive analog circuit (like a filter) are generally given in terms of its frequency response [9]. Hence, the fitness evaluation for the circuit comprises of the following steps:

- Evaluate the circuit at various frequency points.
- Penalize the circuit with varying degrees of penalty where it fails to comply with the desired values.

Quantitatively, over the N frequency points chosen for evaluating fitness, the fitness (F) is given by the absolute weighted deviation as shown in the expression below.

$$F = \sum_{j=1}^{N} W_j(\delta_j) * \delta_j, \quad \delta_j = \| P_{obtained}|_{w_j} - P_{expected}|_{w_j} \|$$

W, P, δ and w denotes the weightage of penalty, performance measure, difference between obtained and target performance measures, and frequency respectively. The obtained fitness is then normalized to a value between 0 to 1 with the fully compliant circuit having a fitness of unity.

III. Experiments and Results

Filters (especially LC filters) are the most well-defined passive analog circuits having a wide range of applications [9]. Two such filters of increasing design difficulty level are chosen as benchmarks for our test experiments. Table. I lists the specifications for the filters along with the results obtained by GAPSYS. The GA is coded in C++. It is to be noted that the the circuits reported here are those that are actually obtained. Further reduction in the size of the circuits is possible through combining similar types of components connected serially or parallely, into one single component.

A. 3rd order Butterworth Low-pass Filter (Filter-1)

This example, taken from some of the previous works [2, 5], is considered to be a difficult task to accomplish. Results show that GAPSYS generates the required circuit in the 2900th $(100*29)$ design explored, compared to 20000, the best among the prior works [5]. Two initial unknown component values are solved and the procedure succeeded for 81 circuits out of 100. The average fitness for the first generation of circuits with and without the component value assignment procedure are 0.000635 and 0.000061 respectively. The former clearly provides a $10X$ improvement. The fully compliant circuit, shown in Fig. 5(a) is obtained in generation 29. The frequency response given in Fig. 5(b) has a dont' care band in the range $925 - 3200$Hz. Fig. 5(c) shows progressively increasing average and best fitness curves.

B. 5th order Elliptic Low-pass Filter (Filter-2)

The specifications for this filter are more stringent than that of the first. The transition band occurs in the range of $1-2$kHz. The size of each population is 200 and the final circuit evolves in generation 142. Here, we solve for three initial unknowns and the initial fitness improvement obtained is 12.5X. The final circuit evolved, frequency response and fitness curves are shown in Fig. 6.

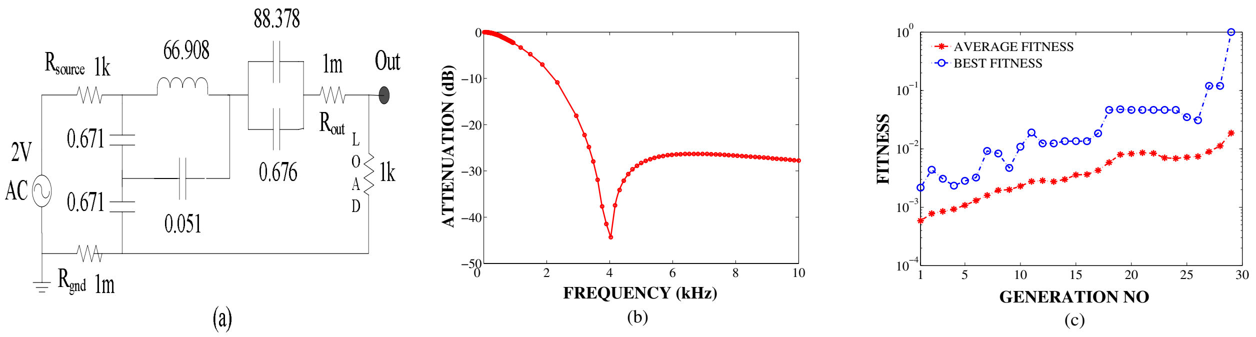


Fig. 5. (a) Circuit (units are Ω, mH and μF); (b) Frequency response; (c) Fitness curves for the evolved 3rd order Butterworth Low-pass filter

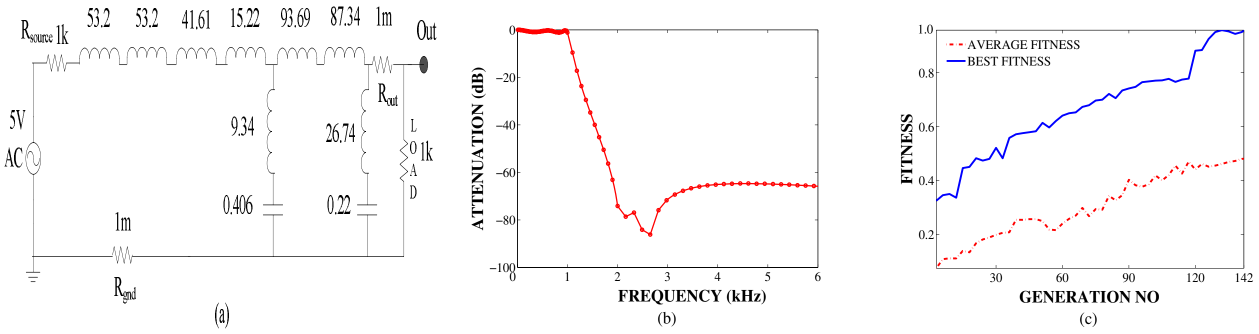


Fig. 6. (a) Circuit (units are Ω, mH and μF); (b) Frequency response; (c) Fitness curves for the evolved 5th order Elliptic Low-pass filter

TABLE I
GAPSYS-EVOLVED FILTER CIRCUITS

PARAMETERS	**FILTER-1**	**FILTER-2**
Performance specifications:		
Passband edge freq. in kHz (w_p)	0.925	1
Stopband edge freq. in kHz (w_s)	3.2	2
Passband attn. in dB (K_p)	3.01	1.11
Stopband attn. in dB (K_s)	22.0	64.7
GA parameters:		
Size of population	100	200
Selection type	Tournament	Tournament
Selection size	2	2
Crossover rate	0.7	0.8
Mutation probability	0.05	0.1
Inductance in H (Min, Max)	$1\mu, 1$	$10\mu, 1$
Capacitance in F (Min, Max)	1p, 1μ	$1p, 1\mu$
W for within 10 % deviation	1	1
W for $>$ 10 % deviation	10	15
Results:		
No. of generations required	29	142
Avg. % of faulty off. ckts / gen	7	10
Initial component value assignment method:		
No. of unknowns	2	3
% Success in solving	81	72
% Initial fitness improvement	$10X$	$12.5X$

IV. Conclusion

We have presented GAPSYS, a tool to synthesize passive analog circuits, based on genetic algorithms. The initial component value assignment procedure increases the fitness level for the first set of circuits. Further, the crossover method employing sub-circuit exchange minimizes the production of *pathological* circuits in subsequent generations. GAPSYS has been successful in evolving two low-pass filter configurations.

References

[1] J.R.Koza, F. H. B. III, D. Andre, and M. A. Keane, "Automated WYWIWYG design of both the topology and component values of electrical circuits using genetic programming," in *Proc. First Annual Conference*, Stanford University, CA, July 1996, pp. 123–131.

[2] J. D. Lohn and S. P. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. Evol. Comput.*, vol. 3, no. 3, pp. 205–219, Sept. 1999.

[3] R. S. Zebulum, M. A. Pacheco, and M. Vellasco, "Comparison of different evolutionary methodologies applied to electronic filter design," in *IEEE Int. Conf. on Evol. Comput.*, May 1998, pp. 434–439.

[4] J. B. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," in *IEE Proc. - Circuits, Devices, Systems*, vol. 147, no. 6, Dec. 2000, pp. 319–323.

[5] C. Goh and Y. Li, "GA automated design and synthesis of analog circuits with practical constraints," in *Proc. 2001 Congress on Evol. Comput.*, vol. 1, no. 1, May 2001, pp. 170–177.

[6] S.-J. Chang, H.-S. Hou, and Y.-K. Su, "Automated passive filter synthesis using a novel tree representation and genetic programming," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 93–100, Feb. 2006.

[7] J. R. Koza, L. W. Jones, M. A. Keane, M. J. Streeter, and S. H. Al-Sakran, "Towards industrial strength automated design of analog electrical circuits by means of genetic programming," in *Genetic Programming Theory Practice Workshop*, Univ. of Michigan, Ann Arbor, May 2004.

[8] J. Grimbleby. TOPCAP: Symbolic analysis tool. [Online]. Available: http://www.elec.reading.ac.uk/people/J.Grimbleby/Analysis.html

[9] L. C. Huelsman, *Active and Passive Analog Filter Design: An Introduction*. McGraw-Hill, Inc., 1993.