ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/4026352

Evolving sinusoidal oscillators using genetic algorithms

Conference Paper · August 2003

DOI: 10.1109/EH.2003.1217647 · Source: IEEE Xplore

CITATIONS
18

READS
145

1 author:

Varun Aggarwal
Aspiring Minds
38 PUBLICATIONS **482** CITATIONS

SEE PROFILE

All content following this page was uploaded by Varun Aggarwal on 02 April 2015.

The user has requested enhancement of the downloaded file.

Evolving Sinusoidal Oscillators Using Genetic Algorithms

Varun Aggarwal
Netaji Subhas Institute of Technology, New Delhi
varun.aggarwal@vsnl.net

Abstract

In the present paper, single-opamp sinusoidal oscillators are synthesized using genetic algorithms. The motivation is to evolve new topologies of oscillators using different active building blocks (ABBs) and automate the study of their properties. A new fitness evaluation scheme by analyzing transfer function of the circuits is used and a learning scheme loosely inspired from Lamarckian search is also suggested. A new problem specific crossover operator is tested and a comparative study of different crossover operators is done. On comparison of the results of the GA with existing results, it was found that the GA rediscovered all the twelve canonic single-opamp based SFOs. Some new interesting opamp, OTRA and DDCC based topologies of oscillators are also presented. It is clearly explained how this study can be extended to other ABBs or multiple ABBs.

1. Introduction

The present work focuses on synthesizing sinusoidal oscillators using Genetic Algorithms. Traditionally, design of sinusoidal oscillators has been carried out based on intuition, inference or analysis. There is no deterministic way to synthesize sinusoidal oscillators (except by exhaustive search!) as in case of other analog synthesis problems. It was only in 1984, when an exhaustive study of oscillators was conducted by Bhattacharya and Darkani [2]. But, this study was constrained to synthesis of single-opamp oscillators with minimum elements and two capacitors, where also it proved to be very tedious. An exhaustive search for oscillators with larger number of components or more than 2 capacitors would become almost impossible manually or highly expensive algorithmically because of significant increase in the size and complexity of the search space. At the same time, authors argue that other oscillators (apart from the ones studied in [2]) may show better properties with regard to frequency distortion, frequency stability, amplitude of oscillation, frequency range, total harmonic distortion, power consumption, ease of fabrication, etc. Whether an oscillator is novel or not depends on many other aforesaid factors rather than just the number of components it uses, e.g., grounded capacitor oscillators, oscillators with all elements grounded, oscillators with effect of parasitics nullified.

In the present paper, we seek to evolve single opamp sinusoidal oscillators, where the GA run shall only be constrained by the number of elements in the oscillators. It is attempted to achieve two goals. Firstly, when 6 element oscillators are evolved, the results can be compared with the results given by Bhattacharya and Darkani [2] to test the capability of GA to identify multiple solutions. Secondly, by increasing the number of elements in the GA run, new topologies of oscillators can be found, which can then be automatically compared on basis of their properties. Furthermore, this study can be easily extended to search oscillators using other building blocks or having certain topological features. These topics are discussed in detail in Section 12.

Earlier works on oscillator synthesis using GAs [4,8,10,13] largely differ from the present work. The oscillators synthesized earlier were a different class of oscillators called Non-linear oscillators [12]. One of these studies [13] was not very successful in evolving oscillators. In other studies, the circuit population was rich in oscillator circuits and a GA was used to optimize on frequency [8, 10], amplitude [13], etc. In the present study, the aim is to design topologies of Linear Oscillators (sinusoidal oscillators) using GAs where the circuit population initially and in subsequent generations does not contain any oscillator topologies. This fact makes this problem entirely different from earlier studies.

Genetic algorithms have been used for analog circuit synthesis earlier (using simulations for fitness measurement) [5,6,9]. The present problem has a bumpy fitness landscape containing low fitness areas, high fitness plateaus and spikes, where the desired circuits reside. The scheme to evaluate fitness by analyzing the transfer function makes it different from earlier studies. For reasons discussed in Section 6, a learning technique

[1] is also implemented in the algorithm and the results are compared with the simple genetic algorithm.

Problem specific operators and techniques have been developed to improve the performance of the algorithm. These techniques may show merit with other circuit synthesis problems as well. A new crossover operator is prescribed which can be used independently and together with the learning technique. Results from these new techniques are compared with the results of the existing techniques.

The paper is organized in the following way. Section 2 gives an introduction to sinusoidal oscillators. Section 3 gives the Problem Statement and objective of the algorithm. Section 4 looks into the circuit encoding scheme used. Section 5 and 6 present the fitness evaluation of circuits. Section 7, 8 and 9 discuss the Selection scheme, Reproduction operators and Repairing Function respectively. Section 10 contains the tabulated results of experiments. Section 11 contains a study of experimental results, while Section 12 shows the final result and practical usability of this study. Finally, Section 13 concludes this study.

2. Introduction to Sinusoidal Oscillators

Sinusoidal Oscillators are analog circuits that oscillate at a fixed frequency and give a sinusoidal output. The criterion for oscillation is that the phase of the loop gain should be zero and the magnitude of the loop gain should be unity at the oscillation frequency, w. This is called Barkhausen criterion [12].

A typical sinusoidal oscillator has the following characteristic equation (CE):

$as^2 + bs + c = 0$

Condition of Oscillation (CO): $b = 0$

Frequency of Oscillation (FO): $f = 1/(2\pi * (c/a)^{1/2})$

Those oscillators, which require readjusting of both the CO and FO for changing the oscillation frequency are called SFOs (Single Frequency Oscillators), while circuits whose oscillation frequency can be changed without disturbing the CO are called VFOs (Variable Frequency Oscillators).

3. Problem Statement And Objective

Bhattacharya and Darkani have ascertained the exhaustive set of Canonic single Opamp sinusoidal SFOs and VFOs [2]. They show that canonic SFOs shall contain 2 capacitors and 4 resistors, while canonic VFOs contain 2 capacitors and 5 resistors.

Development of a Genetic Algorithm to find single-opamp oscillator topologies containing any number of elements is developed. To ascertain the canonic circuits, the maximum number of elements is kept to the likes of six or seven. The number of capacitors in the circuits is not constrained allowing the synthesized oscillators to contain any number of capacitors. This helps in studying a new class of circuits i.e. three capacitor minimum element oscillators for which a systematic study has not been carried out so far. The possible synthesis of a VFO with only 6 elements containing more than two capacitors shall provide a better oscillator than the ones studied earlier.

4. Circuit Encoding

Circuits are represented in a spice-like netlist as used by Grimbleby [6]. The topology is specified by a list of component types together with their terminal nodes. Component type includes resistor, capacitor and “empty component”. Empty component enables us to have variable number of components in a circuit, though the size of the chromosome is fixed. Inductors are not included in the given class of circuits. A tabular depiction of the encoding is shown underneath.

Element_1 node_{11} node_{12}
Element_2 node_{21} node_{22}
Element_3 node_{31} node_{32}
…

Number of elements shall determine the number of rows.
Element_i: Can be a resistor, capacitor or empty component.
Node_{i1}, node_{i2}: Depicts the nodes to which element_i is connected.

5. Fitness Evaluation

Sinusoidal oscillators have definite topologies. Such a topology results in oscillations only when the element values satisfy the CO (given in Section 2) and also yield a real-valued frequency of oscillation.

The technique to synthesize analog circuits used earlier [5,6,9] cannot be applied here. These techniques used spice simulations to assign fitness to a circuit. A genetic algorithm using transient response to assign fitness shall fail to ascertain the topology independent of the component values (i.e. if a fixed set of component values is assigned). This is so because the behavior of the oscillator categorically depends on its element values and at a particular set of values, the topology of an oscillator may not oscillate at all (giving zero, dc or non-linear output), thus misleading assignment of fitness. Circuits, which are not oscillators, also may give similar outputs. If the algorithm tries to carry out optimization (using another GA or hill climbing algorithm) on component values, the method shall again fail due to the spiked nature of the search space, where the circuit shall oscillate only at specific sets of component values. Therefore, we cannot determine whether a topology is that of an oscillator or not in the aforesaid manner.

To judge whether a given circuit is an oscillator, its CE is needed. A generalized topology for single opamp oscillator is given in Figure 1.

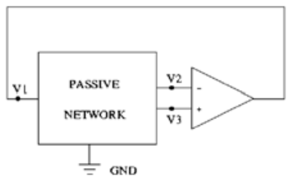

Figure 1. General topology of single opamp oscillator

Let:
$T_3(s) = V_3(s)/V_1(s)$
$T_2(s) = V_2(s)/V_1(s)$

It has been shown that the characteristic equation for such a topology [2] is given by:
$T_3(s)-T_2(s) = 0$

A symbolic analysis software developed by James Grimbleby [7] was used to find the characteristic equation. This software can find out the transfer function of any given circuit containing resistors, capacitors, inductors, the four controlled sources or ideal opamps. The evolved circuit is mapped in the topology given in Figure 2, so that its CE can be ascertained using the transfer function of the mapped topology.

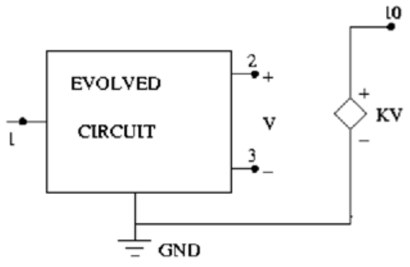

Figure 2. Topology of circuit for simulation

Let K = 1, 1:Input, 10:Output, 2,3:Opamp inputs

The transfer function of the circuit (Figure 2) is given by:
$T(s) = V_2(s)/V_1(s) - V_3(s)/V_1(s)$

The numerator of this transfer function (equated to 0) gives the CE of the circuit.

For the circuit to oscillate, the characteristic equation should only contain terms of s^2 and s^0, with term of s^1 as optional. Also, there should exist a set of values of capacitor and resistors for which the coefficient of s^1 becomes 0 and coefficient of s^2 and s^0 have the same sign (Condition a). If these conditions are fulfilled, the circuit shall oscillate.

In the present study, the following two conditions (Condition b) were checked.

1. Whether the coefficient of s^1 contains a positive and negative term so that it can be made 0.
2. Whether there is at least a single term each in coefficient of s^0 and s^2, which have the same sign.

If these conditions (Condition b) are true, circuit is accepted as an oscillator. [1] The aforesaid criterion is incomplete because even when these conditions (Condition b) are true, it may not be possible to simultaneously make the coefficient of s^1 as 0 and get same sign for coefficient of s^2 and s^0. The need for simultaneously achieving these two conditions (Condition a) sometimes also imposes constraints on the value of resistors and capacitors. Hence, a higher mathematical statement will be needed to ascertain whether a circuit is an oscillator.

Though, the argument given above is mathematically valid, it is generally not observed in practice. This statement is supported by exhaustive search for canonic opamp oscillators [2], where none of the topologies resulted in such interference between the two criteria (Condition a). In general, these criteria only impose constraints on component values though the circuit remains an oscillator. Therefore we shall use Condition b as the criterion to judge oscillators in the present study.

Using the aforesaid criterion, we can ascertain whether a circuit is an oscillator or not, but one cannot judge the fitness of a non-oscillator. All non-oscillators ideally have a fitness value of 0. Therefore, presently it seems the fitness landscape for the problem only contains spikes where the required circuits reside, whereas the fitness at all other points is ideally 0.

The following scheme was used to assign fitness to circuits and decide upon potential high fitness plateaus in the fitness landscape. (Initial fitness assigned is 1)

1. If the circuit is an invalid graph, i.e. CE has 0 terms of any coefficients, it is assigned a fitness a. ($0<a<1$)

[1] An alternative is to optimize on component values to place both roots of CE on the jw axis.

2. If the characteristic equation has n high order terms than s^2, fitness assigned is 1/(b+n). (b>1)
3. If the characteristic equation has no terms of s^2, it is assigned a fitness c. (0<c<1)
4. If the circuit has none of the above characteristics, then the following criteria are tested:
    a. If criterion 1 of Condition b is not fulfilled, the fitness of the circuit is multiplied by d. (0<d<1)
    b. If criterion of 2 of Condition b is not fulfilled, the fitness of the circuit is multiplied by e. (0<e<1)

In the above statements, a, b, c, d and e are numerical parameters which can be set to different values for GA runs. The highest fitness value 1 is that of an oscillator. Once a circuit with fitness 1 is found, the search is stopped.

6. Learning Based Fitness Function

The aforesaid fitness function transformed the fitness landscape to contain regions of low fitness, high fitness plateaus and spikes, where the oscillator reside. The high fitness plateaus are essentially circuits, which satisfy one of the criteria of Condition b. But it is arguable, whether a topology satisfying one of the criteria of condition b is similar to topology of an oscillator. The assumption that the spike identifying an oscillator lies on a high fitness plateau is surely based on intuition of an analog circuit designer.

Therefore a fitness function based on learning [1] loosely inspired by Lamarckian search is used. In this technique, rather than judging the fitness of a circuit only by its absolute fitness, change in the fitness of the circuit before and after applying genetic operators is also considered. As mentioned in [1], this technique shall encourage circuits with lower absolute fitness but showing improvement over generations and discourage circuits with higher absolute fitness but showing little or no improvement over generations.

In the present problem, this strategy makes sense because fitness should be measured as the capability of the circuit to become an oscillator rather than its absolute fitness value. The scheme shall help in smoothening the bumpy fitness landscape.

The following observations support the use of the aforesaid strategy

1. Many circuits show high fitness for many generations by fulfilling one of the criteria of Condition b, though they are unable to fulfill the second criterion and become an oscillator. It is highly probable, that these circuits lie on such high fitness plateaus, which don't contain an oscillator topology. These circuits shall dominate the population due to their highest fitness value (only lower than that of an oscillator), while it is less probable that they will evolve into an oscillator. Therefore, these circuits should be weeded out gradually.
2. Circuits with low absolute fitness are equal to circuits with high absolute fitness in respect that none of them are oscillators. Hence these circuits should also be considered if they show a tendency to become an oscillator.

Different variants of learning technique were conceived and tested. Two of them, which showed better results, are stated underneath.

In both the techniques, two records of fitness values are kept; first one being the absolute fitness calculated using Condition b and the second being the fitness which forms the basis for selection of circuits for the next population. In the first generation all circuits are assigned an equal fitness for the purpose of selection.

In the first technique, the circuit is assigned as fitness, the ratio of absolute fitness of present circuit and the circuit before application of reproduction operators. There is an additional penalty to the fitness according to the number of generations for which the circuit has consistently shown a static fitness value.

Therefore, the circuit's fitness not only depends on its current absolute fitness, but also on its absolute fitness before application of reproduction operators. If it shows improvement in absolute fitness, its fitness value is more than 1, if it becomes worse, its fitness is less than 1 and if it shows a stagnant fitness, then it is assigned fitness 1 together with a penalty (incase it shows stagnant fitness continuously). This fitness evaluation reflects the capability of the circuit to become an oscillator.

The following MATLAB code was used.

```
abs_fit= getfit();
if(abs_fit == prev_abs_fit & oper_app==1)
   no_gen = no_gen + 1;
else
   if(oper_app==1)
      no_gen=0;
   end
end
sel_fit=abs_fit/prev_abs_fit*((factor)^no_gen);
prev_abs_fit= abs_fit;
```

abs_fit: The present absolute fitness of the circuit.
prev_abs_fit: The absolute fitness of circuit before application of reproduction operator.
no_gen: Number of generations for which the circuit has continuously shown the same fitness value.

sel_fit: Fitness which guides the selection of circuits.
factor: Value of penalty factor applied when circuit continuously shows same fitness. (factor<1)
oper_app: It tells whether any reproduction operator was applied to the particular circuit.
getfit(): The function, which returns absolute fitness value according to Condition b.

In the code given above, value of oper_app is set to 1, if a reproduction operator has been applied to the circuit, otherwise it is set to 0. Penalty is imposed on a circuit only when it shows static fitness even after application of reproduction operators.

In the second technique, circuits were penalized if they showed static fitness value in the same way as done in case of the first technique. The circuits retained their absolute fitness value, together with the penalty. No other form of learning was implemented in this scheme. [2]

7. Selection Scheme

The circuits for the next population were chosen using Stochastic Universal Selection (SUS).

8. Reproduction Operators

Experiments were conducted using uniform crossover and two point crossovers. A new crossover operator was also devised. In this operator, first the elements of a circuit are sorted according to one of their connecting nodes (circuit representation contains element with two corresponding nodes), thereafter two-point crossover operator is applied [1]. The steps for this crossover are clearly shown in figure 3. As stated earlier [1], this operator seems to be more realistic as it shall exchange nearby nodes in the two circuits chosen for crossover, retaining the topological features of the remaining circuit. It can be seen as retaining some building blocks while exchanging some building blocks between the two circuits. Though, this scheme is independently applicable, it shows specific merit with the learning scheme.

[2] There can be more variants to this learning technique. Rather than using information regarding the state of circuit in the immediate previous generation, effect of more previous generations can be included. But, it has to be ascertained whether the variant of the circuit that existed some generations earlier has similarity in topological features as the present circuit [1]. Other variants of the learning technique may use different transformation functions to map the effect of fitness of the circuit in previous generations to the present one.

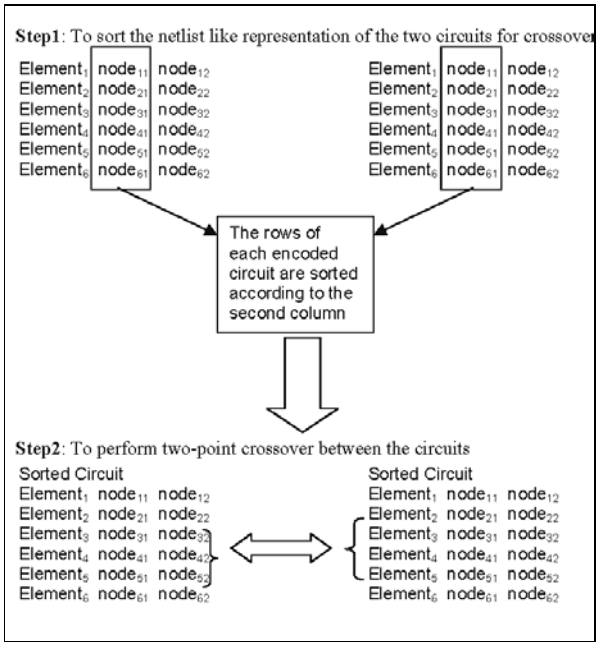
Step1: To sort the netlist like representation of the two circuits for crossover
Element$_1$ node$_{11}$ node$_{12}$
Element$_2$ node$_{21}$ node$_{22}$
Element$_3$ node$_{31}$ node$_{32}$
Element$_4$ node$_{41}$ node$_{42}$
Element$_5$ node$_{51}$ node$_{52}$
Element$_6$ node$_{61}$ node$_{62}$
The rows of each encoded circuit are sorted according to the second column
Step2: To perform two-point crossover between the circuits
Sorted Circuit
Sorted Circuit

Figure 3. A depiction of Sorting, Two-point crossover operator

Quick sort was used to sort the circuits. The sorting operator doesn't prove to be very expensive algorithmically. Once a circuit is sorted, it remains sorted in a piecewise fashion even after application of reproduction operators. For the same reason, the time to sort the circuits remains low after the initial generation.

Experiments were conducted both with and without using the mutation operator. Actually, there is implicit mutation in the repairing function used, which is discussed in Section 9. The mutation operator used replaced a given element by an open circuit.

9. Repairing Function

A repairing function was used which attempts to convert a given circuit to a valid graph. While conceptualizing the repairing function, we tried to minimize the traversal of the circuit netlist.

The repairing function attempts to remove singly connected nodes. It was only for the input node (Node 1 in Figure 2), the two outputs (inputs of opamp, Node 2,3 in Figure 2) and the ground node, that singly connected nodes were allowed. The remaining singly connected nodes were repaired either by shorting them to an existing node (or itself) or by connecting them to an existing node with either a capacitor or a resistor. Both these repairing techniques were assigned probabilities. The repairing function was prevented from falling into infinite loops and hence wasn't able to always repair the circuit.[3]

The repairing function did not check or connect unconnected independent loops in the circuit. This needed repeated traversal of the circuit file.

The basic aim of the repairing function was to connect the unconnected nodes created in a circuit as a result of crossovers and mutations, generation after generation. Replacing an element by an open circuit (Mutation operator) created considerable scope for implicit mutation through the repairing function.

10. Experiments And Results

All coding was done in MATLAB. After some experiments, the following fitness function parameters were used.[4]

a= 0.2 (Kept more than 0 to achieve considerable diversity)
b= 4.0
c= 0.3
d=0.75
e=0.75

The GA parameters used in the runs are as follows

Population size: 35 circuits
Number of elements: 7
Maximum number of generations: 100
Crossover Probability: 0.7
Mutation Probability (per element): 0.05
Mixing Ratio (for Uniform Crossover): 0.5

The results using different crossover operators over 100 runs of the algorithm are tabulated in Table 1.

Table 1. Results of GA with mutation

Crossover Operator	No. of osc. evolved	Redundant circuits per gen.	Total no. of gen.	Osc. with >2 cap.s
Uniform	**94**	8	**1786**	39
Two point	83	7	2925	29
Sort-two point	90	8	2364	38

Results of experiments without explicit mutation are tabulated in Table 2.

Table 2. Results of GA without mutation

Crossover Operator	No. of osc. evolved	Redundant circuits per gen.	Total no. of gen.	Osc. with >2 cap.s
Uniform	**93**	8	**1494**	43
Two point	84	6	2517	36
Sort-two point	89	6	2244	42

The GA parameters used in the algorithm with learning are as follows

Population size: 35 circuits
Number of elements: 7
Maximum number of generations: 100
Crossover Probability: 0.7
Mutation Probability: 0.05
Factor (for penalty): 0.9
Crossover: Sort-two point Crossover

Experimental Results are tabulated in Table 3.

Table 3. Results of GA with learning technique

Learn. Tech.	Mut	No. of osc. evolved	Redundant circuits per gen.	Total no. of gen.	Osc. with >2 cap.s
Tech. 1	Yes	**95**	11	**2664**	40
Tech. 1	No	88	10	2329	49
Tech. 2	Yes	88	8	2399	29
Tech. 2	No	**95**	8	**1612**	63

The search space was also explored using random search (with circuits initially containing 2 capacitors) together with the repairing function. In a population of 10,000 circuits (with maximum 7 elements), 3 oscillator topologies were found.

[3] The repairing function is not the primary operator, which makes the GA work. In previous works [6], four kinds of mutation operators are used, which broadly covers the changes my repairing function does to the circuit. It is shown there, that reducing the crossover probability reduces the performance of algorithm considerably, which is therefore the primary operator. In [5], once again a repairing function is used. Still, it will be worthwhile to study the effect of such repairing functions on the performance of the GA.

[4] These parameters have been set based on intuition and some initial experimental results. There is scope for tuning these parameters for even better results using some optimzation technique (e.g., another GA). Moreover, using dynamic values for these parameters to vary selective pressure during execution might be another interesting study.

11. Observations

In the present experiments, no inference as to the quality of the evolved circuit can be made as all the evolved oscillators have an absolute fitness of 1. Therefore the number of runs that converge to an oscillator and the total number of circuits analyzed during the whole run (proportional to the number of generations) are used as yardsticks to measure the performance of the algorithm.

In the first set of experiments (without learning), Uniform Crossover gives the best result. However, the crossover operator of sorting and two-point crossover gives superior results to that of two-point crossover. It is also evident that the mutation operator doesn't bring any marked improvement in convergence of the algorithm.

The result of the algorithm with learning (with crossover operator as sorting and two-point crossover) is superior to the simple genetic algorithm. The two techniques show an improvement in the number of oscillators evolved. Infact, the second technique with no explicit mutations also shows a reduction in total number of circuits analyzed drastically.

The result of random search shows that the GA with the worst performance is around 3 times better than random search in regard to the ratio of number of oscillators synthesized and number of circuits analyzed. The best performance of the GA reported in Section 10 is 6 times better than random search.

12. Result And Practical Usability

The evolved circuits were manually compared with analytical results. Some of the evolved circuits were invalid due to presence of closed loops connected at a single point of the circuit.

All the 12 SFOs found earlier [2] were rediscovered by the algorithm. The topologies of some 3-capacitor, 6 element circuits evolved are given in Appendix I. Many new 7 and 8 element circuits were also generated. The results have not been observed yet to look for VFOs.

The problems and limitations of GAs applied to this problem are the following:

1. The GA evolves same circuit topology in many runs and there is no way to eliminate this effect.
2. The GA cannot ascertain whether the set of oscillators evolved is the complete set or not.
3. There is no way to fashion the GA to only synthesize VFOs or SFOs. Though the final circuit can be ascertained as a VFO or SFO.

This work of evolution of oscillators using GAs can be used to automate the study of sinusoidal oscillators. The following steps can be followed to build a software to automatically produce oscillators with desired properties.

1. The circuit topologies synthesized by the GA should be first stripped off of redundant components, such as resistors or capacitors in parallel or series with themselves, elements connecting the input nodes of the opamp directly, elements connecting the output of the opamp to ground, etc.
2. Thereafter a revised netlist should be formed to identify and dispose off same topologies evolved by the genetic algorithm.
3. A mathematical statement should be formulated to recognize VFOs and SFOs. An approach similar to the one used by Bhattacharya and Darkani [2] can be easily mapped into an algorithm.
4. The CO and FO should be satisfied using Symbolic Analysis techniques for the desired frequency. More work needs to be done to realize this aspect.
5. Thereafter the circuit can be fed into SPICE or actual hardware. The observed frequency of oscillation, frequency distortion, frequency stability, amplitude of oscillation, frequency range, total harmonic distortion, power consumption, etc. of the oscillator can be ascertained and compared through output from SPICE or actual hardware.

In the aforesaid way, the best topology of an oscillator with desired frequency and properties can be automatically designed.

By making a few modifications in the topology of the circuit fed for simulation (Figure. 2), oscillators using other ABBs such as OTAs, CCs, multiple-opamp oscillators, etc. can be synthesized automatically. Other elements of the algorithm shall remain essentially the same. Thereafter an automated comparison of properties of all these different kind of oscillators can be conducted.

Appendix II contains detailed information about the changes to be made in the topology (of Figure 2) to extend this study to synthesize oscillators using other ABBs. Recently, there has been interest in OTRA [3,11], DDCCC [14, 15] (current-mode) based oscillators, which show advantage over opamp-based circuits. The approach presented in Appendix II was used to synthesize oscillators using these building blocks. Appendix III contains topologies of some unpublished canonic single OTRA based SFOs and a DDCCC based current mode VFO.

13. Conclusion

In the present study a Genetic Algorithm was developed to automatically synthesize oscillators. The synthesized oscillators showed approval with existing analytical results. Also, a new set of oscillators containing 3 capacitors was developed. Some new OTRA and DDCCC based oscillators were also synthesized. The practical usability of this study was established in Section 12.

A new crossover operator and learning based fitness function have been devised, which show improved results when applied to the present study. These techniques are open to statistical tests and controlled experiments. They may prove beneficial for other circuit evolution problems, which may use simulations or actual hardware for fitness evaluation.

Acknowledgement

I thank Prof. Raj Senani, AIC Lab, NSIT, Delhi for suggesting that the present problem be solved by Genetic Algorithms. I also acknowledge his constant support, encouragement and guidance throughout this work. I wish to thank Dr. Paul Layzell for reviewing this paper and providing very useful comments. I thank Dr. James Grimbleby for making available the symbolic analysis software and answering my queries. Finally, I thank Drs. C. Goh and Navid Azizi for answering my emails.

References

[1] V. Aggarwal, "Fitness evaluation based on learning for automatic analogue circuit synthesis using Genetic Algorithms," in Hypothesis Papers, Varun's *Griha* (www.geocities.com/mumukshu/fitllearn.html), September 2002.

[2] B. B. Bhattacharya, M. Darkani, "A Unified Approach to Realization of Canonic RC-Active, Single as well as Variable Frequency Oscillators using Operational Amplifiers," in *Journal of Franklin Institute* (Germany), Vol 317, No. 6, pp. 413-419, 1984.

[3] U. Cam, "A Novel Single-Resistance-Controlled Sinusoidal Oscillator Employing Single Operational Transresistance Amplifier," in *Analog Integrated Circuits and Signal Processing*, Vol. 32, pp. 183-186, August 2002.

[4] R. O. Canham and A. M. Tyrrell, "Evolved Fault tolerance in Evolvable Hardware," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 2, pp. 1267–1271, 2002.

[5] C. Goh and Y. Li, "GA Automated Design and Synthesis of Analog Circuits with Practical Constraints," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, No. 1, pp. 170-177, 2001.

[6] J. B. Grimbleby "Automatic Analogue Circuit Synthesis Using Genetic Algorithms," in *IEE Proceedings: Circuits, Devices and Systems*, Vol 147, No 6, pp. 319-323, 2000.

[7] J. B. Grimbleby, *Software for symbolic Analysis of circuits*, Available at http://www.elec.rdg.ac.uk/Staff_PostGrads/academic/jbg/Analysis.html

[8] L. Huelsbergen, E. Rietman, R. Slous "Evolving Oscillators in Silico," in *IEEE transactions on Evolutionary Computation*, Vol. 1, No. 3, pp. 197-204, September 1999.

[9] J. R. Koza, F. H Bennett, D. Andre, M. A. Keane and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," in *IEEE transactions on Evolutionary Computation*, Vol. 1, pp. 109-128, July 1997.

[10] P. Layzell and A. Thompson, "Understanding Inherent Qualities of Evolved Circuits: Evolutionary History as a Predictor of Fault Tolerance," in *Proceedings of Third Int. Conf. on Evolvable System (ICES2000)*, Vol. 1801 of LNCS, Springer, pp. 133-142, April, 2000.

[11] K. N. Salama and A. M. Soliman, "Novel oscillators using the operational transresistance amplifier," in *Microelectronics Journal*, Vol. 31, pp. 39-47, 2000.

[12] A. S. Sedra, K. C. Smith, *Microelectronic Circuits*. Oxford University Press, New York, 1982.

[13] R. S. Zebulum, M.A. Paheco, M. Vellasco, "Analog Circuits Evolution in Extrinsic and Intrinsic Mode," in *Proceedings of Second Conference of Evolvable Systems*, Vol 1478, pp. 154-165, 1998.

[14] W. Chiu, S. I. Liu, H. W. Ivan and J. J. Chen, "CMOS Differential Difference Current Conveyors and their Applications", in *IEE Proc.: Circuits, Devices, Syst.*, Vol. 143, No. 2, pp. 91-96, , 1996.

[15] S. S. Gupta and R. Senani, "Comment: CMOS Differential Difference Current Conveyors and their Applications", in *IEE Proc.: Circuits, Devices, Syst.*, Vol. 148, No. 6, pp. 335-336, 2001.

Appendix I

On observing the results manually, it was found that the algorithm has found 2 oscillator topologies, which have a total of 6 elements and 3 capacitors. Such oscillators were not considered earlier [2]. They form a part of the set of minimum element single-opamp oscillators. Their properties may show merit, which can be ascertained by non-ideal analysis or experiments. Their topologies are presented underneath.

Topology 1:

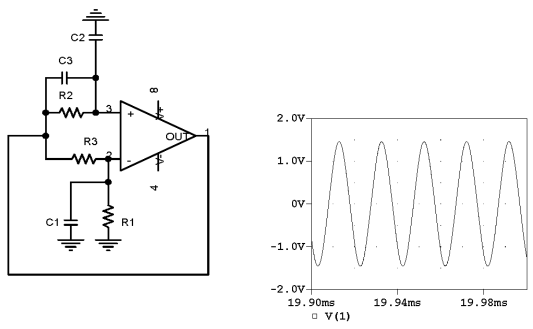

Figure 4. Topology 1, Spice Simulation Results for f = 63.66KHz

FO: $f = 1/(2\pi * (C_1C_3R_1R_2)^{1/2})$
CO: $R_3(C_1R_1 + C_3R_2) - C_2R_1R_2 = 0$

Topology 2:

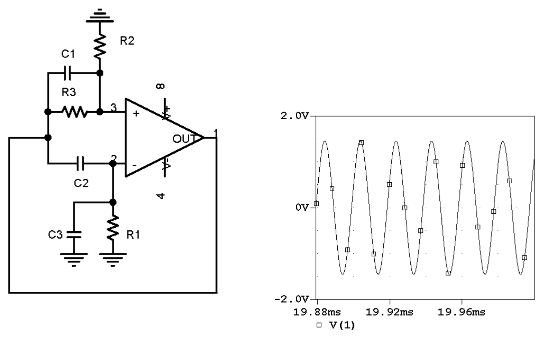

Figure 5. Topology 2, Spice Simulation Results for f = 63.66KHz

FO: $f = 1/(2\pi * (C_1C_3R_1R_3)^{1/2})$
CO: $R_2(C_1R_3 + C_3R_1) - C_2R_1R_3 = 0$

Appendix II

This section explains how the present study can be expanded to other building blocks. The basic change in the algorithm will be in the topology of the circuit fed to the software for finding the CE. For opamp oscillators, the topology of Figure 2 was used.

Opamp has a very high gain (ideally infinite) and therefore the voltages at the two inputs are ideally equal. Using this fact, we equate the voltage at the input nodes of the opamp to find the characteristic equation.

Similarly, in OTRA based circuits, the input terminals are grounded and input currents are forced to have the same value. This is used to find the CE of OTRA based oscillators. The topology given in Figure 6 is thus used.

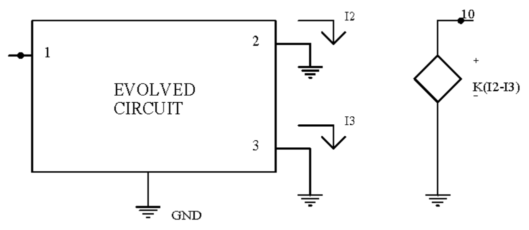

FIGURE 6. Topology to find CE of OTRA based oscillator

Let K = 1, 1:Input, 10:Output (2,3: OTRA inputs)

The transfer function of the circuit is given by:

$T(s) = I_2(s)/V_1(s) - I_3(s)/V_1(s)$;   CE: $T(s)=0$

Using similar approach, the general topology for circuits with single building blocks with infinite gain can be developed. For circuits containing active elements with finite gain or multiple building blocks, the following concept can be used.

Figure 7 depicts any general oscillator. The nodes n1, n2,…,nk (k>0) are connected to ground. The output terminal is also shown.

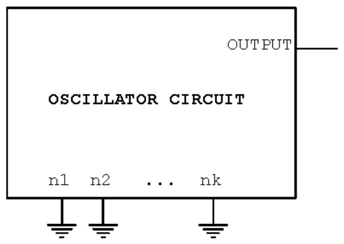

FIGURE 7. A general oscillator

The CE for the oscillator can be found in the following way. Unground one or more nodes (t<k) connected to ground terminals and connect it to a new terminal. This new terminal forms the input terminal. The transfer function of the resulting circuit is the closed loop gain. The denominator of the transfer function gives the CE. This is depicted in Figure 8.

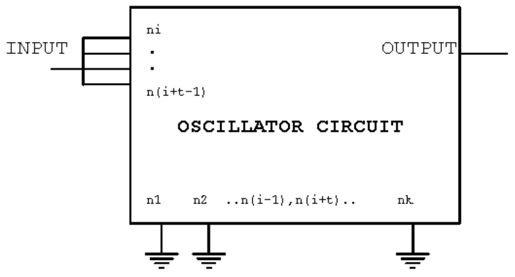

FIGURE 8. Topology to find CE

There are two ways to implement this in GA. In the first way, the genetic algorithm evolves topologies which don't have an input node and one ground node is ungrounded to form the input node for simulation (fitness evaluation and the denominator of the transfer function is the CE. In the second approach, the circuits evolved have both input and output nodes and the denominator of the transfer function is the CE. Finally, the input node is grounded, when the result is presented.

This approach can be used for oscillators containing a single finite gain building block (like OTA, CC). The topology can be built around the fixed nodes of the building block as done in case of opamp oscillators. For circuits containing multiple building blocks, the terminals of the active element will not be explicitly fixed by the GA. Therefore a higher statement of repairing function will be needed.

Appendix III

As an illustration of the ideas presented in Appendix II, some oscillator topologies using newer ABBs have been included, though spice simulations have not been carried out. Therefore, no inference regarding the stability of these oscillators can be made.

Using the modified topology of Figure 4, OTRA based oscillators were synthesized. The algorithm was able to synthesize the canonic VFO [3] and its CR transform. Apart from these and many more oscillators, the algorithm found 3 canonic SFOs. To the best of my knowledge only the first topology given underneath is published [11]. The results are being searched manually to recognize more topologies. Though topology 2 and circuit 3 are CR transforms of each other, the GA has independently synthesized them.

1: Output of OTRA; 2,3: Inputs of OTRA

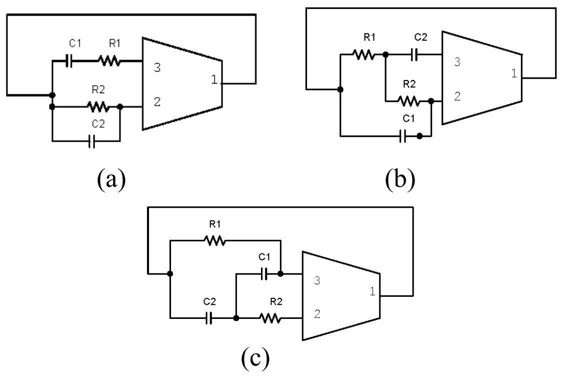

Figure 9. a, b and c depict Topology 1, 2 and 3 respectively

Topology 1
FO: $f = 1/(2\pi * (C_1C_2R_1R_2)^{1/2})$
CO: $R_2(C_1 - C_2) - C_1R_1 = 0$

Topology 2
FO: $f = 1/(2\pi * (C_1C_2R_1R_2)^{1/2})$
CO: $C_1(R_1 + R_2) - C_2R_2 = 0$

Topology 3
FO: $f = 1/(2\pi * (C_1C_2R_1R_2)^{1/2})$
CO: $C_2(R_1 - R_2) - C_1R_2 = 0$

This study applied to DDCCC [14] is near completion and results will be soon presented. A new canonic current mode VFO has been included here for illustration (Figure 10).

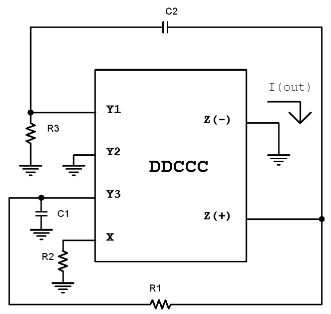

Figure 10. A DDCCC based VFO

FO: $f = 1/(2\pi * (C_1C_2(R_1R_3 - R_2(R_1+R_3))^{1/2})$
CO: $R_2(C_1 + C_2) - 2C_2R_3 = 0$

View publication stats