Evolution of Active Filter Circuits Design Using Genetic Programming

Ogri J. Ushie[1], Maysam F. Abbod[2], and Evans C. Ashigwuike[3]

[1,2] Electronic and Computer Engineering, Brunel University, London
[1] Physic Department, University of Calabar, Nigeria.
[3] Electrical and Electronic Engineering Department, University of Abuja, Nigeria.
[1]ogri.ushie@unical.edu.ng
[2]ushjames@yahoo.com

Abstract: **This research seeks to introduce genetic folding (GF), modified symbolic circuit analysis in Matlab (MSCAM), and automatically simulated Netlist into existing genetic programming (GP) which is a new contribution of this paper. It enhances the development of independent Matlab toolbox for the evolution of active filter circuits. The active filter circuit evolution, especially when operational amplifiers are involved as components, is of the first kind in circuit evolution. The research uses only one software package instead of combining PSpice and Matlab in electronic circuit simulation as in existing GP. This saves the elapsed time for moving the simulation between the two platforms and reduces the cost of subscription. The evolving circuit from GP/F using Matlab simulation is automatically transformed into a symbolic Netlist. The Netlist is fed into MSCAM; where MSCAM uses it to generate matrices for the simulation. The matrices enhance frequency response analysis of four different active filter circuits (low-pass, high-pass, band-pass, band-stop of active filter circuits). Results presented proved the algorithm's efficiency regarding design wise. The research also provided an alternative method of using GP/F for the evolution of active filter circuit, especially when operational amplifier is involved as a component.**

Keywords: – modified symbolic circuit analysis in Matlab, genetic folding, genetic programming and automatically simulated netlist

1.0 Introduction

Analogue circuit design is very important for interfacing to the outside world. In this work, genetic programming/folding **(**GP/F) is used to evolve active filter circuits specified by the user. Examples are given where the attenuation sharpness and cut-off frequency are used as the objectives function for designing the filter. Filter network is an indispensable part of a modern communication system and electronic. The active filter is constructed from capacitor, resistor and operational amplifier as an active source. The active filters eliminate bulky coils and expensive nature of passive filters [1, 2]. Operational transducer amplifier and a detailed step-by-step method is used in the design of active filter [3]. The traditional method of filter design requires mathematical computation.

Evolvable Hardware (EH) is a research area in the evolutionary algorithm (EA) used in electronic circuit simulation without a manual engineering design. Some of the EH usage in an electronic circuit simulation are discussed by different researchers [4-6], while simulation related to filter design is presented in [7]. In addition, some automatic designs of analogue circuits are described in [8-10]. Genetic algorithm (GA) application to optimise component values selections is illustrated in [11]. Lohn et al. [12] developed an evolvable hardware simulation system which is able to automatically design analogue circuits using a parallel GA. Vural et al. [13] propose three evolutionary algorithms namely: harmony search (HS), differential evolution (DE) and artificial bee colony algorithm (ABCA). These evolutionary algorithms are used to minimise CMOS differential amplifier area. In addition, Vural and Yildirim [14, 15] applied the same circuit using PSO in analogue active filter for components selection and they also use PSO, ABCA and GA in which their performances are evaluated respectively.

A basic introductory documentation to genetic programming is discussed in [16]. The use of GP and current – flow analysis for discovery of CMOS amplifier is introduced in [17]. Matlab toolbox for GP and its algorithm is presented in [18, 19]. Hou et al. [20] illustrated GP based on the tree structure for passive filter design. Chang et al. [21] also applied the same approach as that of Hou et al. Chang et al. claimed that their technique is better in efficiency compared to the traditional technique and faster than the previous work. Also, Senn et al. [22] used tree

representation technique which combined two-port theory and GP for analogue circuit synthesis is illustrated. Koza et al. [23] used GP for automatic design of analogue electronic circuits. In addition, Peng et al. [24] use genetic programming and bond graph (GPBG) in electronic circuit design with active components which is an extension of their previous publication on passive component design. Many researchers have worked on the use of an evolutionary algorithm to evolve passive filters, but little has been done in the area of active filters especially operational as part of active filter components.

2.0 Methodology

The detailed procedure involved in this research is illustrated in the flowchart shown in Figure 1. The generated population is calculated to know individually evolved circuit's fitness. If the evolved circuit fulfils the objective function with zero or less than zero error, the circuit is the desired circuit, otherwise generation starts. The processes regenerate until zero or less than zero error is achieved or the objective function is satisfied. Further detailed process involved is illuminated in the flowchart below.

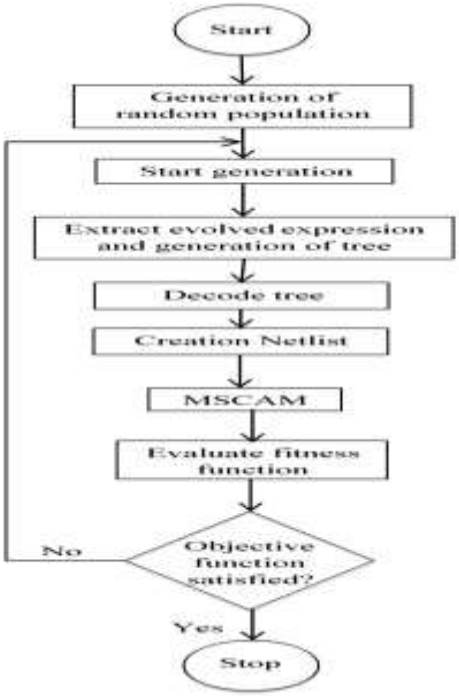

Figure 1: The GP algorithm

2.1 Genetic Programming

The aim of GP is to write a computer code, specify all the parameters required in coding and it has the potential to solve a defined problem. GP can solve a problem that can be measured and compared regarding fitness. GP differs from GA, in that: GP represents its elements with variable length of tree structures, while GA represents its elements with a fixed length of strings. Tree structure can create neural networks, determine parallel computer programmes and designs for analogue electric circuits. The TS can solve problems with an arbitrary size and complexity, which is the opposite of GA with fixed-length. In GP, a random population is created and each individual is evaluated. Individual with the highest fitness is picked to perform crossover, mutation or reproduction, with other individuals to create other individuals for next generation.

GP algorithm in addition to GF, automatically generated symbolic netlist and MSCAM are applied to evolve the low, high, band pass and band stop active filter circuits. Also, the algorithm is tested with small signal analysis circuit of a common collector, common emitter and FET transistor. Using the first circuit for illustration of the detailed procedure involved in the method as:

(a) *Initialisation*

Ten crossover points are taken for swapping and crossover = 0.90, Length of parameters = 8191, mutation = 0.10, population size (PS) = 100 and length of chromosome = 24573. Randomly created population of matrix size = length of chromosome by *PS*. These initial values give the best solutions.

(b) *Coding of circuit's components*

The chromosome is divided into bit group of three, and each group is transformed to its equivalent decimal interpreted as:

'0' for series part (+)

'1' for parallel part (|)

'2' for capacitor (X)

'3' represents inductor (Y)

'>3' for operational amplifier (Z)

The voltage is fixed before the netlist formation to reduce length of chromosome.

2.1.3 *Tree creation*

Operators (+ and |) and terminals (Op amp, capacitor and inductor) defined in Section *3.1.2*. are used to create tree randomly that may be grow or full method:

2.1.1 Mutation

Mutation point is picked in one individual subtree and exchanged its subtree with a randomly generated subtree. In this work, the mutation rate of 0.1 is used.

2.1.5 Crossover: Ten Crossover points in both parents are picked and the subtrees are swapped. It creates

offsprings that are different from their parents. The crossover rate used is 0.9.

2.1 Genetic Folding

GF is one of the classes of an evolutionary algorithm that the genes are structurally organised in order of linear digits separated by dots. In this work, the GF is applied to illustrate how the chromosome is structurally linked from beginning to end so that the circuit can be extracted to generate the netlist. For illustration, Figure 2 shows active fourth order low pass filter GP representations, whereas its GF representation is shown in Table1.

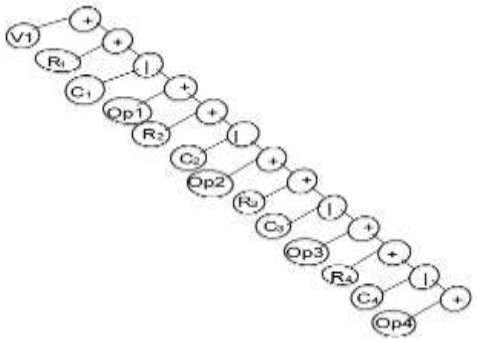

Figure 2: Tree representations active fourth order low pass filter

The tree structure expression is interpreted (from top to down and from left to right), where + for series and | for parallel:

$$+V_1+R_1|C_1+Op_1+R_2|C_2+Op_2+R_3|C_3+Op_3+R_4|C_4+Op_4 \quad (1)$$

The expression in equation (1) is read as follows: the first series operator is a two operands operator with two values (V1, series). The V1 is a terminal. The second series operator is a two operands operator with values (R1, parallel). The R1 is a terminal. The first parallel operator is a two operand operator with values (C1, series). The C1 is a terminal. The third series operator is a two operands operator with values (OP1, series) and so on. The expression is represented using GF in Table 1. Each element is given a position number in order as in the first row, the elements are in the second row and the elements are folded over their complementary position genes in the third row.

Table 1: The GF representation.

1	**2**	**3**	**6**	**7**	**14**	**15**	**30**	**31**	**62**	**63**	**126**	**127**	**254**
+	R_1	|	C_1	+	OP_1	+	R_2	|	C_2	+	OP_2	+	R_3
2.3	0.2	6.7	0.6	14.15	0.14	30.31	0.30	62.63	0.62	126.127	0.126	254.255	0.254

255	**510**	**511**	**1022**	**1023**	**2046**	**2047**	**4094**	**4095**	**8190**
|	C_3	+	OP_3	+	R_4	|	C_4	+	OP_4
510.511	0.510	1022.1023	0.1022	2046.2047	0.2046	4094.4095	0.4094	8190.0	0.8190

V_1 and the first series (+) are fixed to reduce the length of chromosome and therefore not included in Table 1. The GF starts with the series operator in position 1 and terminates with element *'OP_4'* in position 8190. The first series operator has the R_1 terminal in position 2 (terminal always end a branch) and parallel operator in position 3. The parallel operator in position 3 has terminals C_1 position 6 and series operator in position 7. The series operator in position 7 has terminals OP_1 position 14 and series operator in position 15 and so on. The terminals are represented using their indices position. GF is summarised with the following points:

1. The chromosome arrangement comprises of float string in the gene and the position of the gene.
2. The gene arrangement is left child *(LC)* side separated by dot and right child (*RC)* side.
3. The dot mean and.

4. The operator with two operands has *LC* and *RC*.
5. The operator with one operand has *LC* and 0 in the *RC*.
6. The terminals have 0 in *LC* and value in the *RC*.

2.1 Creation of Netlist

To enable evaluation: how evolving circuits have performed in the population regarding the desired circuit, the evolving circuits are extracted and transformed into a symbolic netlist. Variables representation in the Matlab programme are single variable as defined in Section 2.1.2 before they are encoded again to individual component types. The same component type is encoded with different subscripts to differentiate them if they have more than one component in the circuit. The encoding is done symbolically. Using one component type (resistor) for illustration, all resistors are represented by Y. Assuming there are four Y (4 resistors), they are replaced by [‘a-d’] so that if an element is picked and it is ‘a’ it is assigned R1, if another element is picked and it is ‘b’ it is assigned R2, and so on. The evolving circuits in form of a tree structure are described as: operand terminate a branch (capacitor, inductor and op amp), whereas operator continues the TS (parallel or series part). The TS is read from left to right and from top to bottom. The branches after the operand are replaced by ‘0’. Likewise, the branches after the ‘0’ are replaced by ‘0’ so that all the branches after the operands are replaced by ‘0’ up to the maximum length of TS. All the ‘0’ components are then discarded to leave only the evolving circuit.

The method of stack separation evaluation is then used to arrange the elements as it is connected. The series elements are numbered from 0 to the highest number while the parallel elements are all numbered 0 since all are grounded. The components labels are differentiated with subscript from 1 to the last e. g. 4 resistors in a circuit are numbered as R1 R2 R3 R4. The netlist is thus formed as: if an element is picked; it is between the first node number and the second node number. It is vital to note that, the series components are always connected to the next node number that is not zero. For example, if the extract from the evolving circuit above:

$$+V+R|C+Z+R|C+Z+R|C+Z+R|C+Z \quad (2)$$

In equation (2) *Z* represent op amp and substituting the values of series and parallel form equation (3) as:

$$0V1R0C2Z3R0C4Z5R0C6Z7R0C8Z \quad (3)$$

The symbolic netlist formation is as follows: It begins with the element name, accompanied by node1, node2 and accompanied by component value. If a circuit has op amp as component(s), netlist starts with it, numbered from one to the last. The formation proceeds thus: op amp name, followed by its output node number, inverting node number and non-inverting node number

OAmp1 3 2 3

OAmp2 5 4 5

OAmp3 7 6 7

OAmp4 9 8 9

V 0 1 component value

R1 1 2 component value

R2 3 4 component value

R3 5 6 component value

R4 7 8 component value

C1 0 2 component value

C2 0 4 component value

C3 0 6 component value

C4 0 8 component value

2.2 Modified Symbolic Circuit Analysis in Matlab

Gielen and Sansen [25] illustrated how symbolic simulation is useful while creating a large part of analytical prototype automatically in a circuit needed to size that circuit in system design automation. MSCAM is a Matlab toolbox that uses netlist from PSpice or netlist generated from simulation described in Section 2.3 to generate matrices. These matrices can be used for circuit parameters analyses or optimization. Detailed implementation of MSCAM in a programme is explained in [26, 27]. In this research, the MSCAM is utilised to take in automatically simulated netlist from Matlab simulation that is symbolic, transform it to symbolic matrices before substituting their real values (using the eval command in Matlab) to get frequency response and then compare it to the set frequency response specified in the objective function. The process continues until the set frequency is achieved.

2.3 Objective Function Specifications for the Active Fourth-Order Low Pass Filter

Voltage gain (frequency response) is applied to analyse the fourth-order active filter circuit.

a. The frequency range between 1 Hz to 1 MHz is specified for the circuit with a cut-off frequency of 70 KHz. The impedance of each operand node is evaluated using 3 parameters namely: given frequency, component type and component value.
b. Every operand node returns impedance upward. The operator node executes the corresponding arithmetic (parallel and series) to achieve its impedance after receiving impedance from its branches and the process continues till the circuit impedance is computed.
c. The source voltage is then divided by the circuit impedance to get the current flowing in the tree. Starting from the source, the current that flows in the series node is equal to the current that flows in from the source. While in the parallel node, the current that flows in is divided inversely proportional to the branches' impedance and continues until the node end.
d. The node voltage is got by multiplication of impedance by the current, and the voltage gain is got by dividing the voltage across the specified output node by source voltage. The process is demonstrated mathematically below.

The symbolic matrices *A* and *B* (equation (1) and equation (2) respectively), which are automatically generated from MSCAM is used to formulate objective function as:

$$A = [(V_S \div R_{111}); \quad 0; \quad 0; \quad 0; \quad 0; \quad 0; \quad 0; \quad 0] \tag{4}$$

$$B = \begin{bmatrix} b_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & b_{32} & b_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{54} & b_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{65} & b_{66} & b_{67} & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{76} & b_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & b_{87} & b_{88} \end{bmatrix} \tag{5}$$

where $b_{11} = j \times \omega \times C_1 + R_{111}$

$b_{21} = -A_v$

$b_{22} = A_v + 1 + (1 \div R_2)$

$b_{23} = -1 \div R_2$

$b_{32} = -1 \div R_2$

$b_{33} = j \times \omega \times C_2 + R_2$

$b_{43} = -A_v$

$b_{44} = A_v + 1 + (1 \div R_3)$

$b_{45} = -1 \div R_3$

$b_{54} = -1 \div R_3$

$b_{55} = j \times \omega \times C_3 + R_3$

$b_{65} = -A_v$

$b_{66} = A_v + 1 + (1 \div R_4)$

$b_{67} = -1 \div R_4$

$b_{76} = -1 \div R_4$

$b_{77} = j \times \omega \times C_4 + R_4$

$b_{87} = -A_v$

$b_{88} = A_v + 1$

$$C = B^{-1} \times A \tag{6}$$

where *C* has unknown voltages across all the nodes to be calculated. C_n is the voltage across the last node (n) being evaluated to get its frequency response. The eval command in Matlab is used to substitute the values of variables in the automatically generated symbolic matrices (matrices A and B). The command (logspace) in Matlab is used to span the frequency range over 50 intervals for both evolving and targeted circuit. The difference between the roots means square value of targeted frequency response and GP/F evolving circuit frequency response is the error. The relationship is presented in equation (7):

$$W = rms(f_{11} - f_{22}) \tag{7}$$

where *W* is the error, f_{11} is the targeted frequency response and f_{22} is the GP evolved frequency response.

3.0 Results and Discussion

Four different active filter circuits are used to illustrate the algorithm's efficiency. The algorithm successfully evolves all the 4 circuits with zero or less than zero error. Results presented are the same as the objective function frequency response.

3.1 Fourth-order active low pass filter circuit

Fourth order active low pass filter is used to illustrate the intermediate stages in the circuit's evolution. The 7th iteration GP evolved circuit is in Figure 3. The frequency response curve of the evolved circuit after 7th iteration is indicated with red colour and the PSpice circuit simulation of the original circuit is shown with the black colour all in Figure 4. The original circuit specification is with cut-off frequencies of 67.9 kHz while the GP/F evolved circuit is with cut-off frequencies of 93.3 kHz. The algorithm evolved the circuit with an error of 0.0342 and a gain of 1.

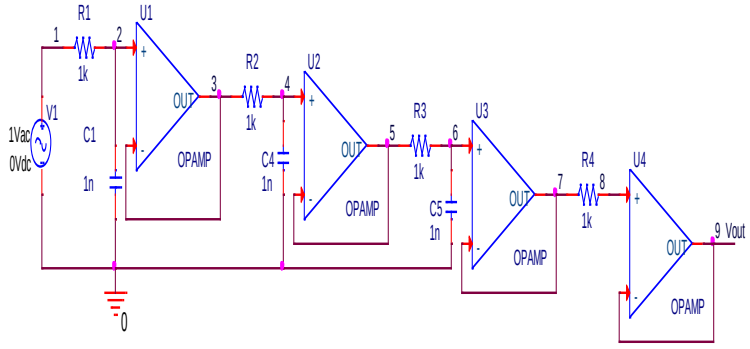

Figure 3: *7th iteration GP/F evolved circuit for the active fourth order low pass filter*

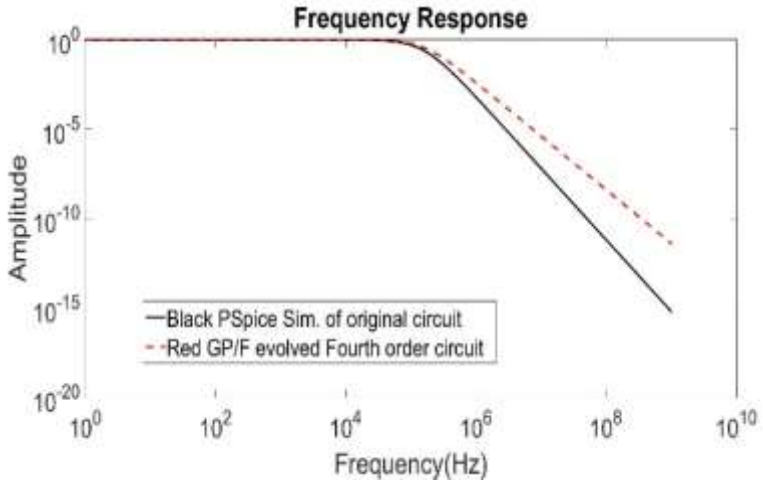

Figure 4: *7th iteration GP/F evolved circuit for the active fourth order low pass filter*

The algorithm evolved the GP/F desired or expected circuit (fourth order active low pass filter) after 18th iterations shown in Fig 5 and its tree structure is in Figure 2. It takes thirty-six minutes to evolve the circuit after eighteen iterations. The MSCAM frequency response of the evolved circuit is shown with the black colour whereas the PSpice circuit simulation of the 18th iteration is shown with the red colour all in Figure 6. The original circuit specification and the GP/F evolved circuit are with cut-off frequencies of 67.9 kHz. The algorithm evolved the circuit with an error of 8.6921E-10 and a gain of 1.

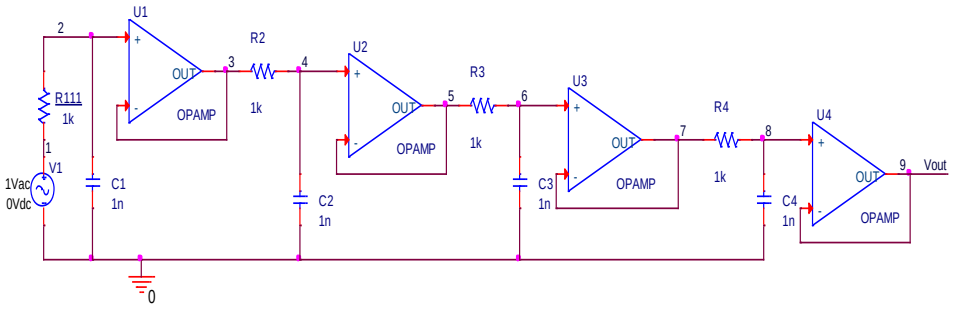

Figure 5: GP/F evolved circuit for the active fourth order low pass filter

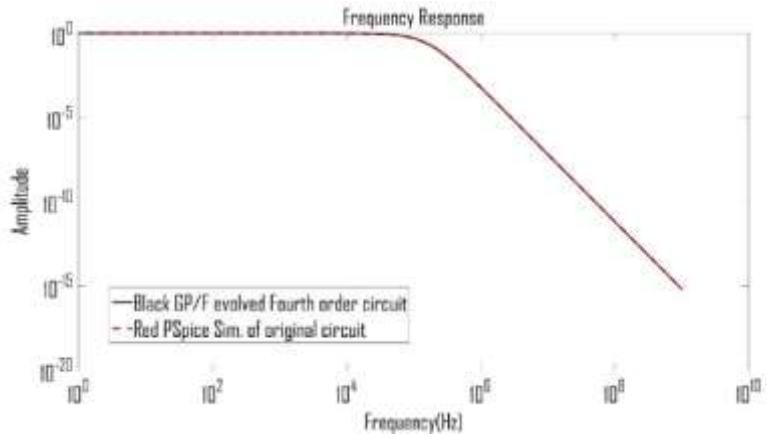

Figure 6: Frequency response curve of GP/F evolved (black), 7th iteration MSCAM simulation (blue) and PSpice simulation (red) for the low pass filter

3.2 Fifth-order active high pass filter circuit

The GP/F evolved desired circuit is shown in Figure 7. It takes fourteen minutes to evolve the circuit and seven iterations. The MSCAM frequency response of the evolved circuit is shown in red colour, the GP/F evolved PSpice circuit simulation is indicated in black colour as indicated in Figure 8. The original circuit specifications are with cut-off frequencies of 1134 Hz. The algorithm evolved the circuit with an error of 9.6085E-8 and a gain of 1.

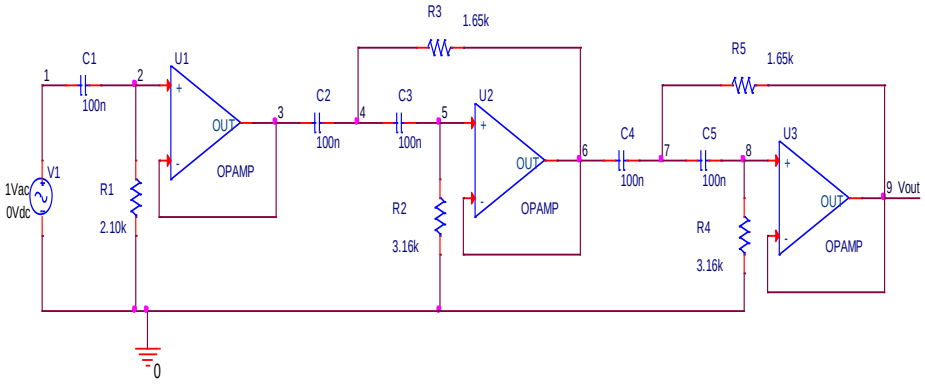

Figure 7: GP/F evolved circuit for the active high pass filter with feedback

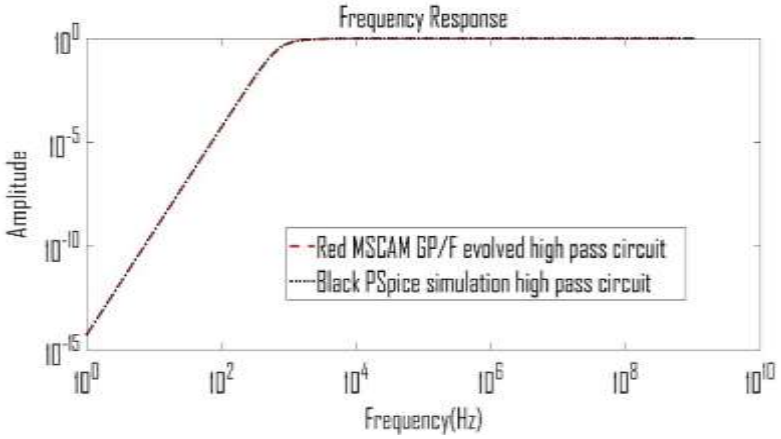

Figure 8: Frequency response curve of GP/F evolved (black) and PSpice simulation (red) for the for the high pass filter

3.3 *Active band pass filter circuit*

The GP/F evolved desired circuit is shown in Figure 9. It takes twenty-six minutes to evolve the circuit after thirteen iterations. The MSCAM frequency response of the evolved circuit is shown in red colour, the GP/F evolved PSpice circuit simulation is indicated in black colour as indicated in Figure 10. The original circuit specifications are with the lower and upper cut-off frequencies of 31.42 Hz and 47.86 kHz respectively. The algorithm evolved the circuit with an error of 3.5614E-7 and a gain of 1.

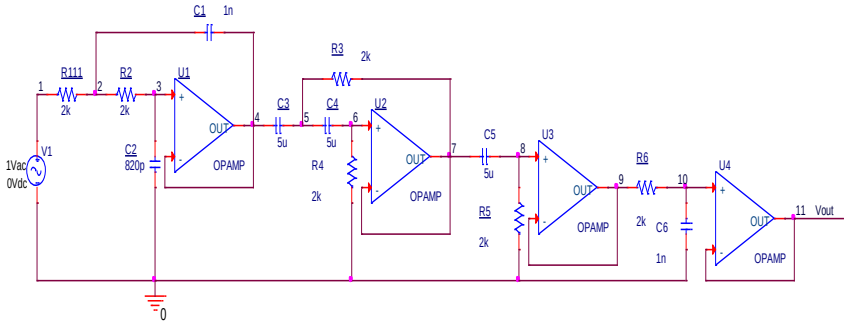

Figure 9: GP/F evolved circuit for the active band pass filter

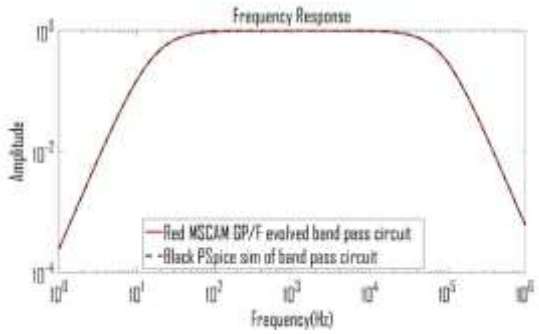

Figure 10: Frequency response curve of GP/F evolved (red) and PSpice simulation (black) for the band pass filter

3.4 Active band stop filter circuit

The GP/F evolved desired circuit is shown in Figure 11. It takes twenty-two minutes to evolve the circuit after eleven iterations. The MSCAM frequency response of the evolved circuit is shown in red colour; the GP/GF evolved PSpice circuit simulation is indicated in black colour all in Figure 12. The original circuit specifications are with lower and upper cut-off frequencies of 45.7 kHz and 562.3 kHz respectively. The algorithm evolved it with an error of 8.1899E-8 and a gain of 1.

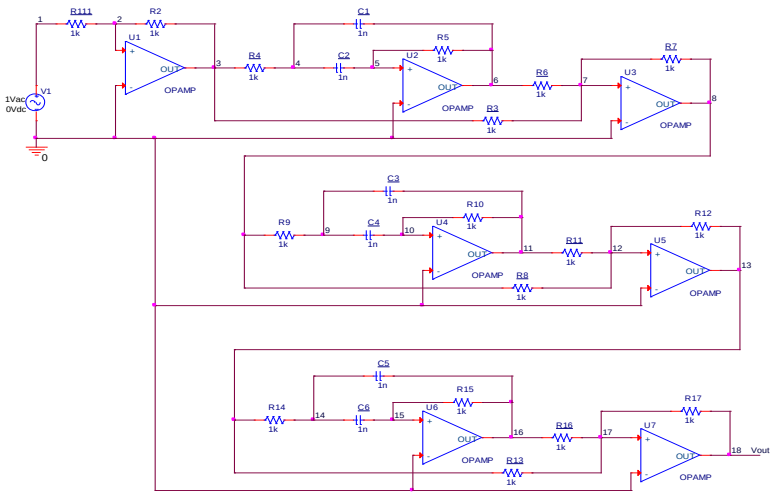

Figure 11: GP/F evolved circuit for the active band stop filter

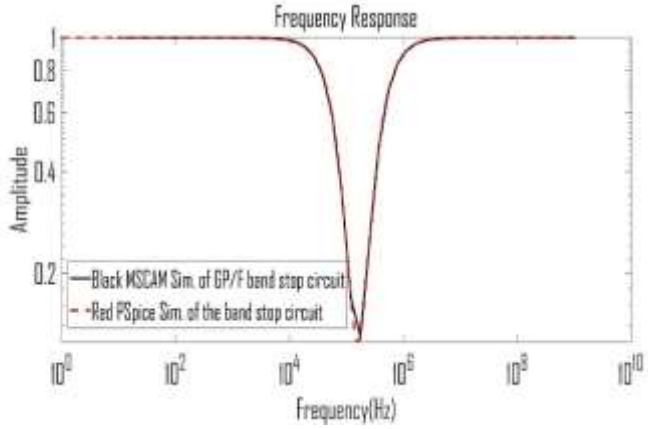

Figure 12: Frequency response curve of GP/F evolved (black) and PSpice simulation (red) for the band stop filter

4.0 CONCLUSION

This paper introduces three methods namely: GF, MSCAM, and automatically simulated Netlist into existing GP, which is a new contribution of this work, it enhances the development of independent Matlab toolbox for the evolution of active filter. The simulator uses only Matlab compared to existing GP which combines Matlab and PSpice. The elapsed time used while transferring simulation between software packages is reduced and able to evolve operational amplifier circuits of its first kind. This standard algorithm with little modification can be used to design different circuits unlike the human design method that each circuit has to undergo tedious mathematical computation.

References

[1] A. Waters, "Active Filter Design," McGraw-Hill, 1991.

[2] W. G. Jung, "Op Amp Applications Handbook," Newnes, 2005.

[3] J. Tow, "A step-by-step active-filter design," Spectrum, IEEE, vol. 6, pp. 64-68, 1969.

[4] K. K. Anumandla, R. Peesapati, S. L. Sabat, S. K. Udgata, and A. Abraham, "Field programmable gate arrays-based differential evolution coprocessor: a case study of spectrum allocation in cognitive radio network," IET Computers & Digital Techniques, vol. 7, pp. 221-234, 2013.

[5] S. Maheshwari, "Analogue signal processing applications using a new circuit topology," IET Circuits, Devices & Systems, vol. 3, pp. 106-115, 2009.

[6] S. Vakili, S. M. Fakhraie and S. Mohammadi, "Evolvable multi-processor: a novel MPSoC architecture with evolvable task decomposition and scheduling," IET Computers & Digital Techniques, vol. 4, pp. 143-156, 2010.

[7] S. Maheshwari and B. Chaturvedi, "High-input low-output impedance all-pass filters using one active element," IET Circuits, Devices & Systems, vol. 6, pp. 103-110, 2012.

[8] G. Alpaydin, S. Balkir, and G. Dündar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," Evolutionary Computation, IEEE Transactions On, vol. 7, pp. 240-252, 2003.

[9] E. Martens, and G. Gielen, "Classification of analog synthesis tools based on their architecture selection mechanisms," Integration, the VLSI Journal, vol. 41, pp. 238-252, 2008.

[10] O. Mitea, M. Meissner, L. Hedrich and P. Jores, "Automated constraint-driven topology synthesis for analog circuits," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, 2011, pp. 1-4.

[11] T. Kaya and M. C. Ince, "The Design of Analog Active Filter with Different Component Value using Genetic Algorithm," International Journal of Computer Applications, vol. 45, pp. 43-47, 2012.

[12] J. D. Lohn, S. P. Colombano, G. L. Haith and D. Stassinopoulos, "A parallel genetic algorithm for automated electronic circuit design," in Proc. of the Computational Aerosciences Workshop, NASA Ames Research Center, 2000.

[13] R. A. Vural, B. Erkmen, U. Bozkurt and T. Yildirim, "CMOS differential amplifier area optimization with evolutionary algorithms," in Proceedings of the World Congress on Engineering and Computer Science, 2013, .

[14] R. A. Vural and T. Yildirim, "Component value selection for analog active filter using particle swarm optimization," in Computer and Automation Engineering (ICCAE), 2010 the 2nd International Conference On, 2010, pp. 25-28.

[15] R. A. Vural, T. Yildirim, T. Kadioglu and A. Basargan, "Performance evaluation of evolutionary algorithms for optimal filter design," Evolutionary Computation, IEEE Transactions On, vol. 16, pp. 135-147, 2012.

[16] M. Walker, "Introduction to genetic programming," Tech.Np: University of Montana, 2001.

[17] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions On, vol. 21, pp. 1237-1252, 2002.

[18] S. Silva and J. Almeida, "GPLAB-a genetic programming toolbox for MATLAB," in Proceedings of the Nordic MATLAB Conference, 2003, pp. 273-278.

[19] K. Rodríguez and R. Mendoza, "A Matlab Genetic Programming Approach to Topographic Mesh Surface Generation," 2011.

[20] H. Hou, S. Chang and Y. Su, "Economical passive filter synthesis using genetic programming based on tree representation." in IEEE International Symposium On Circuits And Systems, 2005, pp. 3003.

[21] S. Chang and Y. Su, "Automated passive filter synthesis using a novel tree representation and genetic programming," Evolutionary Computation, IEEE Transactions On, vol. 10, pp. 93-100, 2006.

[22] A. Senn, A. Peter, and J. G. Korvink, "Analog circuit synthesis using two-port theory and genetic programming," in AFRICON, pp. 1-8, 2011.

[23] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," Evolutionary Computation, IEEE Transactions On, vol. 1, pp. 109-128, 1997.

[24] X. Peng, E. D. Goodman and R. C. Rosenberg, "Robust engineering design of electronic circuits with active components using genetic programming and bond graphs," in Genetic Programming Theory and Practice VAnonymous Springer, pp. 185-200, 2008.

[25] G. Gielen, and W. Sansen, "Symbolic Analysis for Automated Design of Analog Integrated Circuits," Springer Science & Business Media, 2012.

[26] O. J. Ushie, M. Abbod and E. Ashigwuike, "Matlab Symbolic Circuit Analysis and Simulation Tool using PSpice Netlist for Circuits Optimization," International Journal of Engineering and Technology Innovation, vol. 5, no. 2, pp. 75-86, 2015.

[27] E. Cheever, "Symbolic Circuit Analysis in MATLAB (SCAM)," Swarthmore College, http://www.swarthmore.edu/, November, 2005.