ANALOG CIRCUIT OPTIMIZATION WITH HIERARCHICAL GENETIC ALGORITHMS - 3RD ORDER LOW-PASS BUTTERWORTH FILTER EXAMPLE

by

Olcay Durul Azeri

B.S., Electronic & Telecommunication Engineering,

Yıldız Technical University, 2005

Submitted to the Institute for Graduate Studies in

Science and Engineering in partial fulfillment of

the requireds for the degree of

Master of Science

Graduate Program in Electrical & Electronics Engineering

Boğaziçi University

2009

To my lovely

FAMILY

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Günhan Dündar. Special thanks go to him for all the help, guidance, encouragement, and motivation during my Ms. Thesis.

I would like to thank Prof. Dr. Francisco Fernandez for his innovative ideas and guidance.

I would like to thank Prof. Dr. H. Levent Akın and Yrd. Doç. Dr. Arda Deniz Yalçınkaya for their contribution.

I would like to thank Özsun S. Sönmez for his help and guidance.

I would like to thank special persons in this world; my Dad (Ekrem Azeri), my Mom (Güleser Azeri) and my sister (Oya Ilgın Azeri or my cücü ☺).

I would like to thank my friends: Sezgin Bayrak, Erdal Karaçal, Ahmet Köseoğlu, Kınay Bozdemir to let me using their PC's for simulation.

I would like to Seyrani Korkmaz for his help.

ABSTRACT

ANALOG CIRCUIT OPTIMIZATION WITH HIERARCHICAL GENETIC ALGORITHMS - 3RD ORDER LOW-PASS BUTTERWORTH FILTER EXAMPLE

In the several previous studies, various kinds of Hierarchical Genetic Algorithm structures have been used to solve complex problems. In this thesis, a master-slave mode, two-layered Hierarchical Genetic Algorithm was designed to optimize an implementable complex integrated circuit.

Our expectations from two-layered proposed HGA is to minimize the total process time, to reach the same solution quality with standard HGA in complex problems and to increase the compatibility to any other topology by working two modules collaboratively with each other.

In the example chosen in the thesis, the upper module (master module) will optimize a 3rd order active low-pass Butterworth filter and the lower module (slave module) will optimize the OPAMPs (MOS technology based integrated circuit) in the filter circuit, with SPICE based simulation. Thanks to this algorithm which will be realized by the proposed HGA, solutions can be implemented with current MOS technologies and the same result quality can be obtained with standard HGA and a low total process time is obtained such as in the Genetic Algorithm.

ÖZET

HİYERARŞİK GENETİK ALGORİTMA İLE ANALOG DEVRE OPTİMİZASYONU – 3. DERECEDEN ALÇAK GEÇİREN BUTTERWORTH FİLTRE ÖRNEĞİ

Daha önce yapılan çalışmalarda karmaşık problemleri çözmek için çeşitli hiyerarşik genetik algoritma yapıları kullanılmıştır. Bu tezde, uygulanabilir karmaşık entegre devre optimizasyonu yapmak için master-slave modunda iki katmanlı Hiyerarşik Genetik Algoritma tasarlandı.

İki katmanlı olarak kullandığımız hiyerarşik genetik algoritmadan beklentimiz, iki modülün müşterek çalışarak, toplam işlem süresini kısaltmak, klasik genetik algoritmadaki sonuç kalitesini karmaşık problemler de sağlamak ve değişik topolojilere uyumluluğu artırmaktır.

Tezde seçilen örnekte, üst modül 3. dereceden alçak geçiren aktif Butterworth filtreyi optimize edecek ve alt modülde filtrelerde kullanılan OPAMP'ı (MOS teknoloji bazlı entegre devre) SPICE tabanlı simülasyon ile optimize edecek. Önerilen HGA ile gerçekleştirilecek bu sistem sayesinde, çözümler güncel MOS teknolojisi ile gerçeklenebilecek, standart HGA ile aynı sonuç kalitesi yakalanacak ve Genetik Algoritmada ki gibi düşük toplam işlem süresi sağlanacaktır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

f_c	Frequency
w_c	Angular Frequency
BW	Bandwidth of the OPAMP
E	Gain of the OPAMP
GA	Genetic Algorithm
HGA	Hierarchical Genetic Algorithm
K_0	Konstant in in transfer function
K_1	Coefficent of s in transfer function
K_2	Coefficent of s^2 in transfer function
K_3	Coefficent of s^3 in transfer function
k_f	Frequency value in the cost
k_p	Power consumption value of OPAMP in the cost
k_a	Chip layout area value of OPAMP in the cost
k_R	Maximum resistance/minimum resistance value in the cost
k_C	Maximum capacitance/minimum capacitance value in cost
LM	Lower Module
OPAMP	Operational Amplifier
PGA	Parallel Genetic Algorithm
Ro	Output resistance of the OPAMP
UM	Upper Module
W_1	Weight of frequency value in the cost
W_2	Weight of power value in the cost
W_3	Weight of area value in the cost
W_4	Weight of max resistance/min res. ratio value in the cost
W_5	Weight of max cap/min cap ratio value in the cost

1. INTRODUCTION

1.1. Overview of Genetic Algorithm

A genetic algorithm (in short GA) is an optimization technique to search and find approximate solutions to combinatorial optimization problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, recombination (crossover) and natural selection.

The earlier instances of Genetic Algorithms appeared in the late 1950s and early 1960s, programmed on computers by evolutionary biologists who were clearly seeking to model aspects of natural evolution. It did not occur to them that this strategy could be more generally applied to artificial problems.

Genetic algorithms originated from the studies of cellular automata, conducted by John Holland and his colleagues at the University of Michigan in 1970s. Research in GAs remained largely theoretical until the mid-1980s, when The First International Conference on Genetic Algorithms was held at The University of Illinois. As academic interest grew, the increase in desktop computational power allowed for practical application of the new technique. In 1989, The New York Times writer John Markoff wrote about Evolver, the first commercially available desktop genetic algorithm. Custom computer applications began to emerge in a wide variety of fields, and these algorithms are now used by several companies to solve large scale problems, data fitting, trend spotting, budgeting and virtually any other type of combinatorial optimization.

1.2. Overview of Analogue Circuit Synthesis

Analogue circuit synthesis is a essential in various levels at analogue and digital design processes by adjusting transistor sizing, calculating the passive component values and adjusting bias voltages and currents. Most MOS based integrated circuits (ASIC designs) require analogue modules to communicate external integrated circuits or passive components in distributed circuits [16]. Because of the nonlinearity in analogue designs, searching a huge solution space makes analogue simulation more difficult. In the last few years some techniques were used which incorporate heuristics [17], knowledge-based optimization [18], and simulation-based optimization [19]. Evolutionary algorithms and especially genetic algorithms were included to analogue circuit synthesis approximately three decades ago [20].

Some knowledge-based analogue synthesis computer programs; OASYS [21], BLADES [22] and IDAC [23] enable rapid synthesis of analog MOS circuits. On the other hand, the results of these knowledge based approaches are inaccurate.

In addition to, equation-based analogue synthesis techniques have also been used. Some examples of these approaches are OPASYN [24], OPTIMAN [25] and AMGIE [26]. These techniques are quite fast due to using analytical equations for circuit evaluation, if the terms in transfer function are not complex. As the equations get more complicated, this model loses its efficiency.

Nowadays, simulation-based approaches are widely used for analogue circuit synthesis. One commercially simulation-based approach is GBOPCAD [27] which uses HSPICE. Using a commercial tool brings some advantages; to get rid of writing software and to adapt easily to different simulation environments. On the contrary, using commercial tool has undeniable drawbacks. Two of the main drawbacks are latencies with user operating and breaking the synthesis process when it has an error in inter-application communication space.

Consequently, writing an in-house simulator and circuit synthesizer ensures several advantages. Latencies from operator can diminish by full automatic processes. Making a new search algorithm can decrease total process time with solving complex large-scale problems.

2. GENETIC ALGORITHM AND OPERATORS

In the nature, each species needs to adapt to a changing environment in order to maximize of its survival property. The stronger species and individuals have more chance to survive and breed. GA is formed with adaptation, breeding, elimination, and mutation of individuals like in the natural environment. In other words, GA uses these rules in order to solve problems or in order to optimize required processes or nearly unlimited number of applications. GA is also a powerful method which can simplify, clarify and solve more complicated problems quickly and robustly.

Terms in GA:

- **Individual** - A member of population or any possible solution
- **Chromosome** – Coded chain of an individual
- **Allele** - Gene in the chromosome or cell of coded chain
- **Population** - Group of all individuals
- **Search Space** - All possible solutions to the problem
- **Locus** - The position of a gene on the chromosome
- **Genome** - Collection of all chromosomes for an individual
- **Fitness Function** – tool can eliminate the unwanted individuals
- **Cost** – vital member of Fitness Function to help the make elimination

Operations in GA:

- **Reproducing** (Crossover)– Breeding of parents with the altering genes method which ensures to produce new generation
- **Mutation** – Random altering of the gene in the chromosome of an individual
- **Selecting** (Elimination) – Choosing the satisfactory or strong individuals and killing the unsatisfactory or weak individuals in the population by the help of fitness function.

2.1. Recombination

In the recombination process, new individuals are produced by breeding of the parents in the population. First, individuals are selected randomly in order to be coupled to breed. Recombination can be done by real numbers or binary numbers which is called crossover. Chosen couples breed via crossover techniques:

- Single-point / double-point /multi-point crossover
- Uniform crossover
- Shuffle crossover
- Crossover with reduced surrogate

In single-point crossover technique, a crossover position is determined and genes are altered from this position

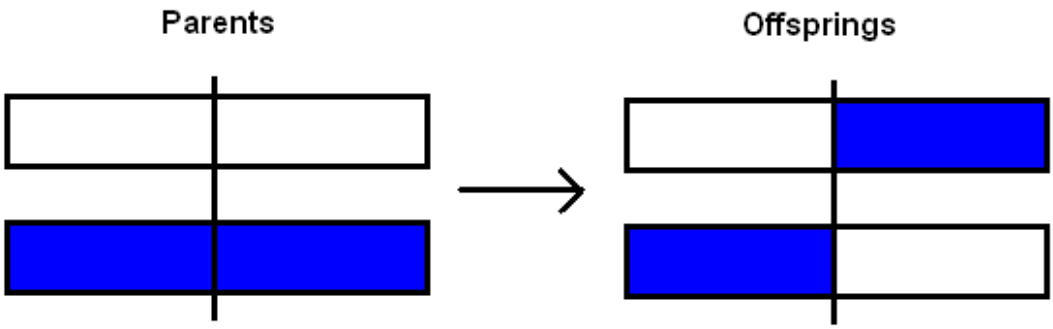

Figure 2. 1. Single point crossover

In double-point crossover technique, two crossover positions are determined and genes are altered from these positions.

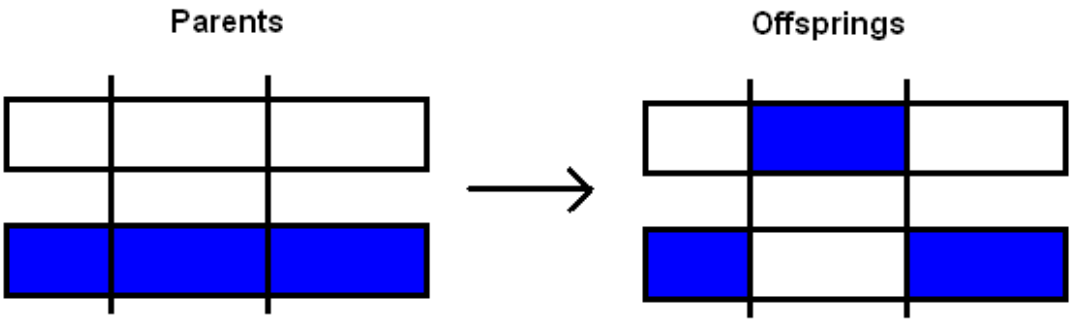

Figure 2. 2. Double point crossover

In multi-point crossover technique, more than two crossover points are selected, and selected positions determine where the genes are altered.

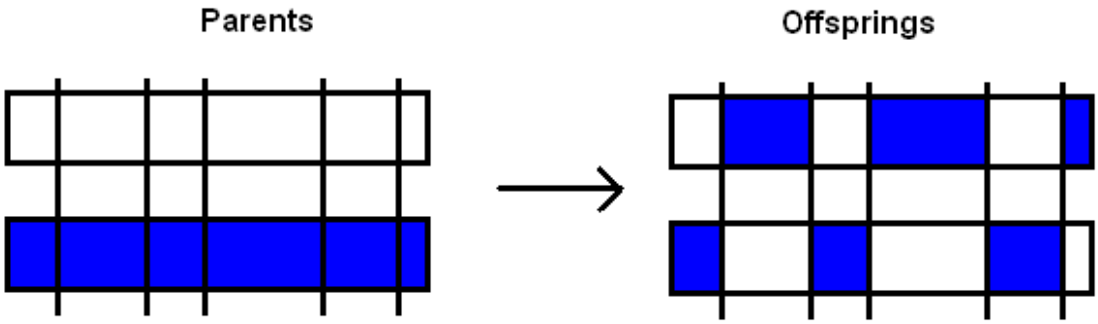

Figure 2. 3. Multi point crossover

In the uniform crossover, each bit or gene is selected randomly, either from the first parent or from the second one.

In selective crossover, one offspring of one parent gets the dominant allele genes. The second offspring gets the recessive genes.

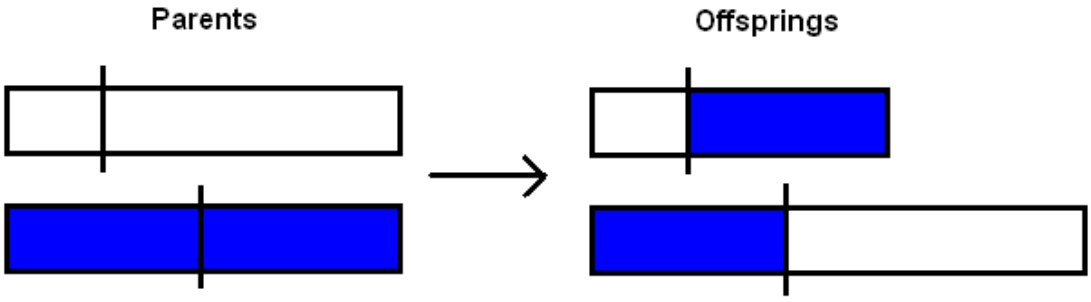

Figure 2. 4. Cut and splice crossover

In cut and splice crossover, crossover points are different for the couple which is dated. Children born with the different string lengths.

In shuffle crossover, there are three steps in this type of crossover. First, the positions of bits or genes in the string are randomly shuffled, then the two strings are crossed over, at last, the offspring is un-shuffled.

2.2. Selection

Selection determines which individuals are chosen after recombination and how many offspring each selected individual produces. The main idea is, to give preference to better individuals, allowing them to pass on their genes to the next generation. The goodness of each individual depends on its fitness. Fitness may be determined by an objective function or by a subjective judgement. Also cost or punishment is a part of fitness function.

The types of selections in GAs are:

- Roulette Selection
- Rank Selection
- Steady-State Selection
- Elitist Selection

- Tournament Selection
- Truncating Selection

In the roulette selection, parents are selected according to their fitness. The better the chromosomes are, the more chances they have to be selected. Like the in the game, there is a roulette wheel where all chromosomes in the population are placed, every individual has its place according to its fitness function.

Roulette selection has problems when the fitnesses are different very much among the individuals. If we assume that the best chromosome fitness is more than 90%, then it covers a very big part of the roulette surface, hence the other chromosomes will have very few chances to be selected.

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2, third 3, and the best will have the maximum fitness value. The maximum fitness value is the number of chromosomes in population.

In steady-state selection main idea of this selection is that a big part of chromosomes should survive to next generation.

GA then works in the following way. In every generation, a few (good - with high fitness) chromosomes are selected for creating a new offspring. Then, some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of the population survives to a new generation.

In the elitist selection, the fit members of each generation are guaranteed to be selected. When creating new population by crossover and mutation, we have a big chance, that we will loose the best chromosome.

Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

In tournament selection, subgroups of individuals are chosen from the larger population, and members of each subgroup compete against each other. Only one individual from each subgroup is chosen to reproduce.

In truncation selection, individuals are sorted according to their fitness. The parameter for truncation selection is the truncation threshold value. Truncation threshold value indicates the proportion of the population to be selected as parents. Individuals below the truncation threshold do not produce offspring.

2.3. Mutation

Mutation is a vital operator in GA. By mutation, individuals' genes are randomly altered. These variations or mutation steps are generally small. They will be applied to the variables of the individuals with a low probability (mutation probability or mutation rate). Normally, offspring are mutated after being created by recombination according to mutation rate.

Mutation occurs:

- Real value mutation
- Binary value mutation

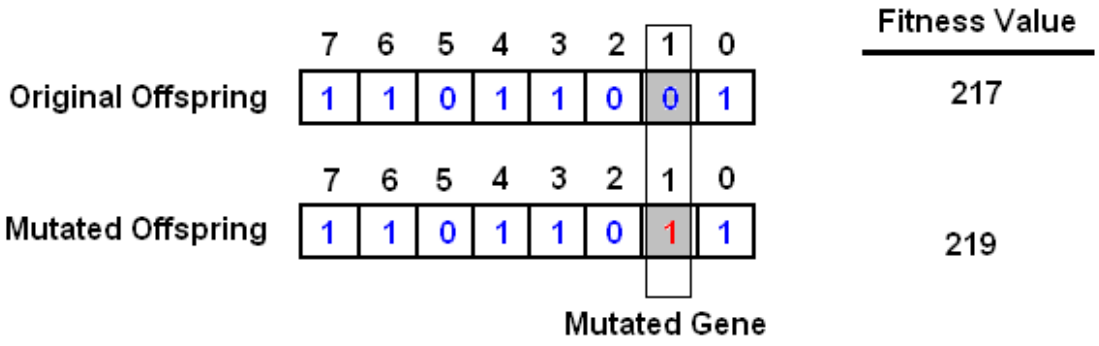

Figure 2. 5. Mutation schematic

2.4. Usage of Genetic Algorithm

GA and evolutionary algorithms are used in several approaches in order to solve problems and make optimization in large scale solution space.

The first and most important point is that genetic algorithms are intrinsically parallel. Most other algorithms are serial and can only explore the solution space to a problem in one direction at a time, and if the solution they discover turns out to be suboptimal, there is nothing to do but abandon all work previously completed and start over. However, since GAs have multiple offspring, they can explore the solution space in multiple directions at once. If one path turns out to be a dead end, they can easily eliminate it and continue work on more promising avenues, giving them a greater chance each run of finding the optimal solution.

Due to the parallelism that allows them to implicitly evaluate many schemas at once, genetic algorithms are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search exhaustively in any reasonable amount of time.

Another notable strength of genetic algorithms is that they perform well in problems for which the fitness landscape is complex - ones where the fitness function is discontinuous, noisy, changes over time, or has many local optima.

GA has been used to schedule jobs in a sequence dependent setup environment for a minimal total tardiness. All jobs are scheduled on a single machine; each job has a processing time and a due date. The setup time of each job is dependent upon the job which immediately precedes it. The GA is able to find good, but not necessarily optimal schedules, fairly quickly.

GA is also used to schedule jobs in non-sequence dependent setup environment. The jobs are scheduled on one machine with the objective of minimizing the total generally weighted penalty for earliness or tardiness from the jobs' due dates. However, this does not guarantee that it will generate optimal solutions for all schedules.

Added to these GA is used in distributed computer network topologies design and in financial modeling applications.

Also GA is used in those areas which are stated below:

- Aerospace engineering
- Astronomy and astrophysics
- Chemistry
- Electrical engineering
- Game playing
- Geophysics
- Materials engineering
- Mathematics and algorithmic
- Military and law enforcement
- Molecular biology
- Pattern recognition and data mining
- Robotics
- Routing and scheduling
- Systems engineering

3. HIERARCHICAL GENETIC ALGORITHM

With the growing required in GA applications, GA algorithms have started to develop day by day. In some applications, especially in large scale problems, adding specific improvements and tactics is required in genetic algorithm when the time consumption is considerable. A type of a genetic algorithm has started to be used which is called Hierarchical Genetic Algorithm in short (HGA).

The structure of Hierarchical Genetic Algorithm (HGA) is more flexible and modular than the conventional genetic algorithm. HGA has multi-layered hierarchical topology which brings it various efficiencies. The most significant advantage of being multi-layered topology ensures dividing large-scale problems into sub-problems by using parallel processed Genetic Algorithm increases the efficiency of the optimization search and diminishes the total process time.

HGA may have two layers: top layer (master) and the low layer (slave) or may be multi-layered: one top level and more than one bottom layers. Thanks to this architecture, it is possible to use a mix of simple models or GAs (rapid solvers) and the complex models or GAs (slow solvers) together in order to reach solutions. This mixed topology can provide us same quality in the complex modules (GAs) and same time consumption is simple modules (GAs). The top layer or higher sub-populations generally search a large space with lower resolution, opposite to this lower-layer or lower levels search smaller space with higher resolution. Communications among the populations are provided by migration of individuals with different strategies. In this hierarchical topology, solutions go up and down the layers and progressively the best solutions keep going up until they are completely refined, at last if the solution is satisfied top population make the decision on complete the whole processes. In the design of the HGA, the structure of the hierarchy and topology strategies like individual migrations, coordination among the top layer and bottom layers is important. Constructing an efficient coordination and load sharing in HGA allows us to accelerate the convergence speed of the algorithm to the optimum, and to diminish the total process time. Added to these, being a multi-layered structure in HGA

ensures multi-objective flexible architecture. Hence, altering one of the sub population or bottom GA algorithm ensure us solving any other problem with small changes or alterations.

As a consequence, using Hierarchical Genetic Algorithm with different strategies and models can achieve to solve complex problems with the same quality in GA but faster than GA. In other words, HGA works better than GA in complex problems.

3.1. Types of HGA

Hierarchical Genetic Algorithm is based on the Parallel Genetic Algorithms (in short PGAs). Hierarchical Genetic Algorithms (HGAs) work in the form of hierarchical topology, having different layers to perform different tasks (upper and lower level). Upper level and lower levels can be evaluated together or separately. There can be individual migration among the levels or individuals can be different for each level.

Some models of parallel model are stated below:

- Master-Slave (global) parallelization;
- Subpopulations with migration;
- Subpopulations with static overlapping;
- Subpopulations with dynamic overlapping;
- Massive parallelization.

In Master-Slave model, only evaluation of individuals and genetic operators are paralleled and such parallel processes are all dependents of the master process. In this kind of parallel HGA mode we can easily share the slave algorithm on other processors or computers.

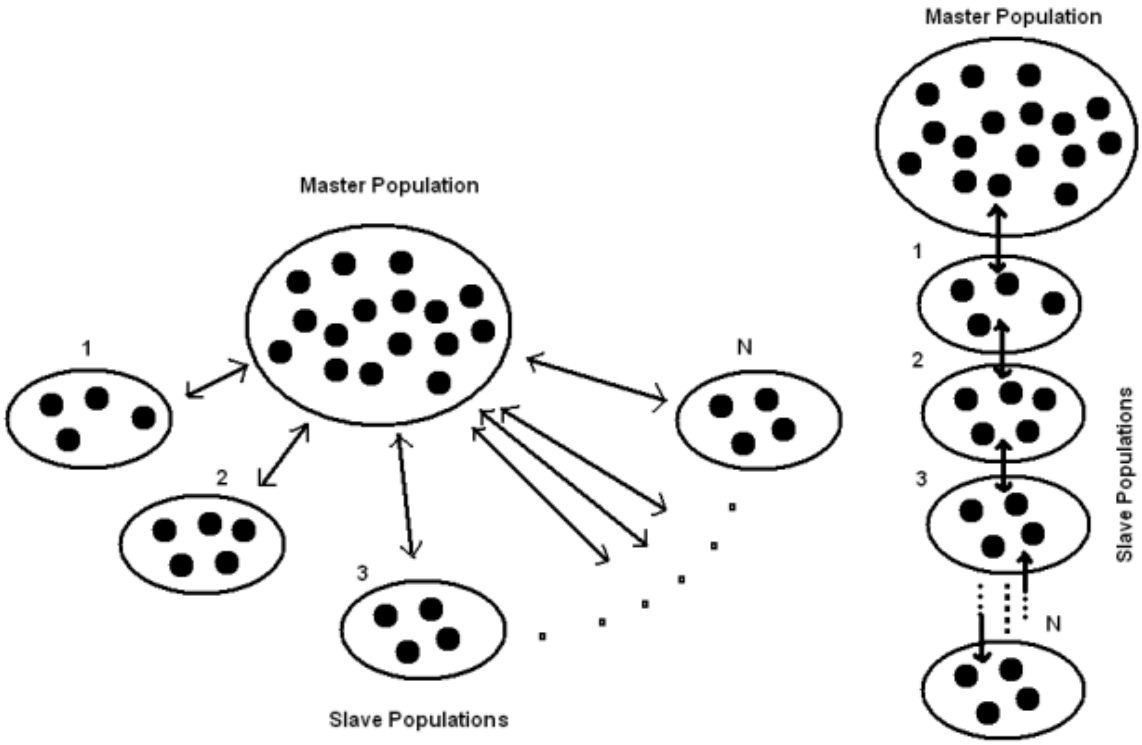

Figure 3. 1. HGA in master-slave mode

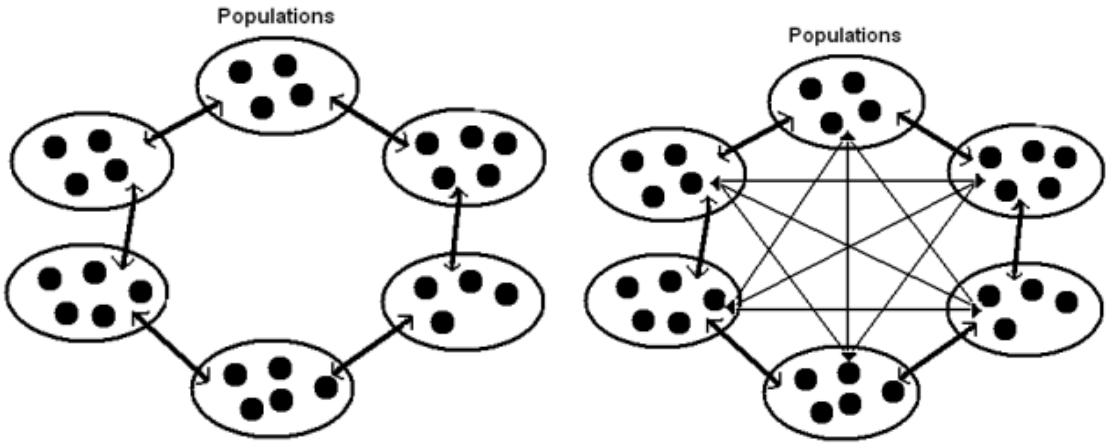

Figure 3. 2. HGA in island mode

4. PROPOSED HIERARCHICAL GENETIC ALGORITHM AND OPTIMIZATON EXAMPLE CIRCUIT

In this thesis, a two-layered hierarchical genetic algorithm is used to optimize a complex MOS integrated circuit. A third order Butterworth low pass filter is selected for a simulation example. The Proposed HGA is formed of two layers. A master population or first layer which is called upper population or upper module (in short UM) runs with GA to optimize the values of its own individuals. These individuals are formed by external capacitances, external resistances and cut-off frequency of filter and Butterworth characterization of the filter. The slave population or second layer which is called lower population or lower module (in short LM) also used different GA algorithm to optimize its own individuals. In this layer, transistor based OPAMP circuits are calculated and optimized with SPICE based formulas.

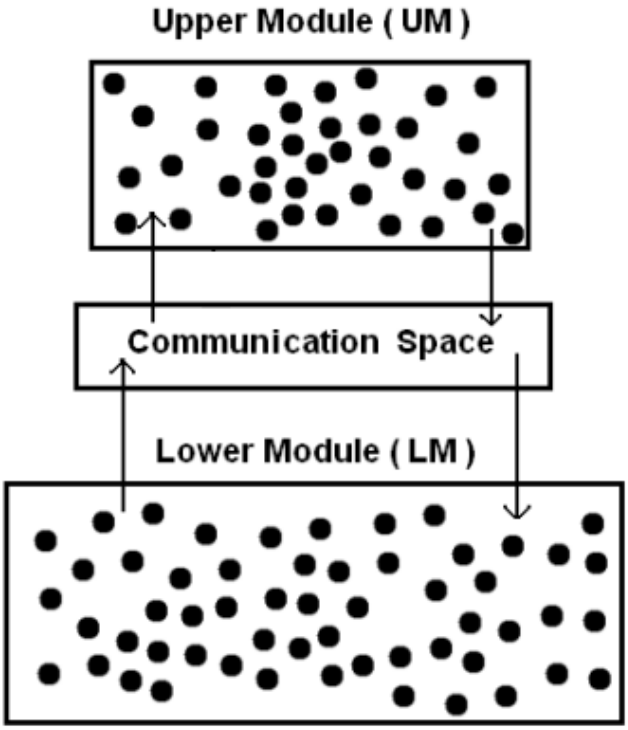

Figure 4. 1. The type of the Hierarchical Genetic Algorithm in this work

Both of the two modules have their own specific GA operators. Some genes or some part of gene strings migrate to the lower module, which processes these genes with its own operators and gives results to the communication space. The UM gets the new genes from the communication space and goes on to reproducing new generations. In this application both modules are executable files and the communication space is a folder which include two "*.txt" files. One of these "*.txt" files is used by the UM to write the genes which are sent to the LM, so this same file is the file which the LM reads the genes. The second file is used by the LM to write back the processed genes and it is also the file which the UM uses to read processed datas (immigrant genes).

First, the UM starts to process its own algorithm. After some generations, it sends the immigrant genes in the chromosomes to the LM, these genes or locus are processed and optimized by the LM. After some generations, the processed datas or immigrant genes are send back to the UM by the LM. This cycle continues until the satisfactory child or generation born.

The upper module (UM) optimizes a third order active Butterworth low pass filter with non ideal practical OPAMPs. The lower module (LM) optimizes the OPAMPs' bandwidth and output resistance and gain with Spice parameters by optimizing the transistor based circuit. The UM gives the required OPAMPs' gains, output resistances and bandwidths to the LM by writing them to external text file in the communication space. The LM gets these parameters and starts to process its own GA and finds the required OPAMPs approximate results. After optimizing, the LM sends the chip layout areas, chip power consumptions, bandwidth, gain and output resistance of OPAMPs and waits for new OPAMP requests from the UM. The UM gets the required or close to required OPAMP values and keeps on to process its own GA in order to get a satisfied individual, this transaction pursues until the UM gets the result and breaks up the all processes. The UM sends of 20 OPAMPs features (unity gain, bandwidth and output resistance) by writing it to text file in the directory, the LM process these 20 OPAMPs and turns back with the optimized values (chip area, chip power) by writing the external text file.

4.1. 3RD Order Butterworth Low-Pass Filter

To perform the proposed Hierarchical Genetic Algorithm a low-pass Butterworth filter is selected. 3rd order active low pass Butterworth with 2nd order Sallen-Key topology included is chosen.

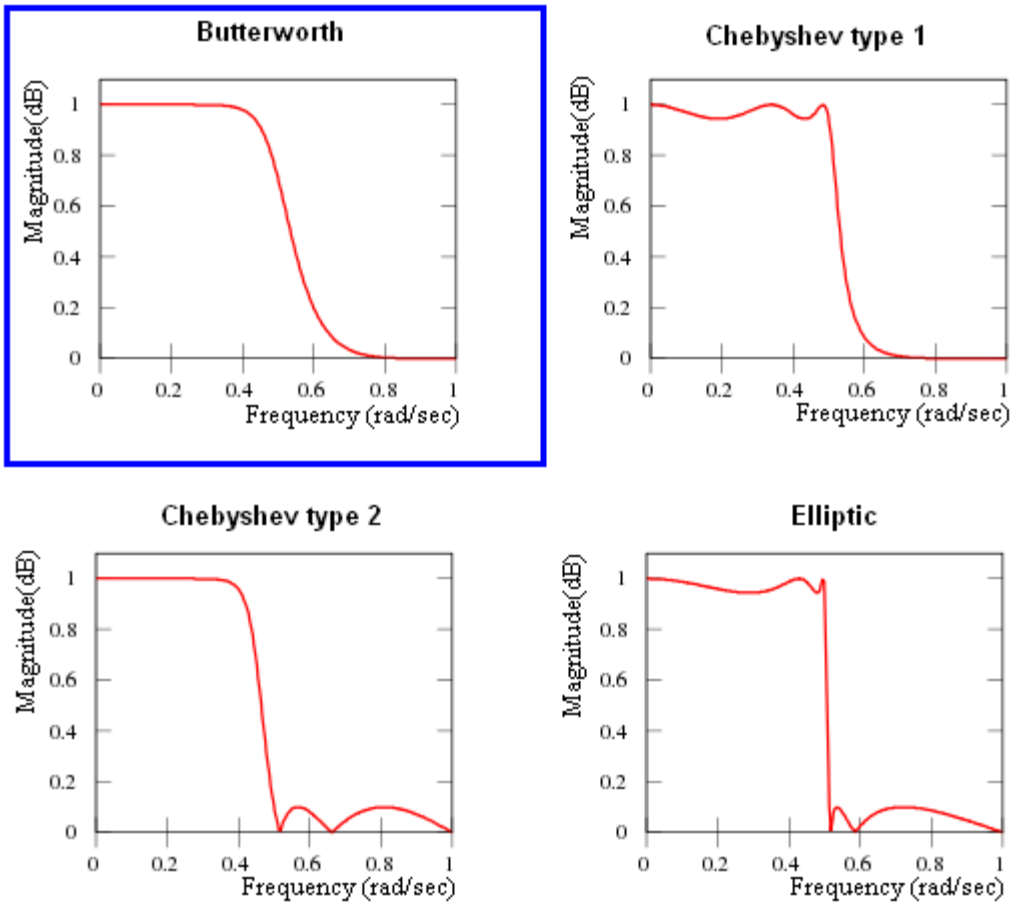

Figure 4. 2. Comparison of low pass Butterworth filter with other types

Butterworth filter has flattest pass-band magnitude response. Added to these, pulse response is better than the Chebyshev and rate of attenuation is better than the Bessel filter.

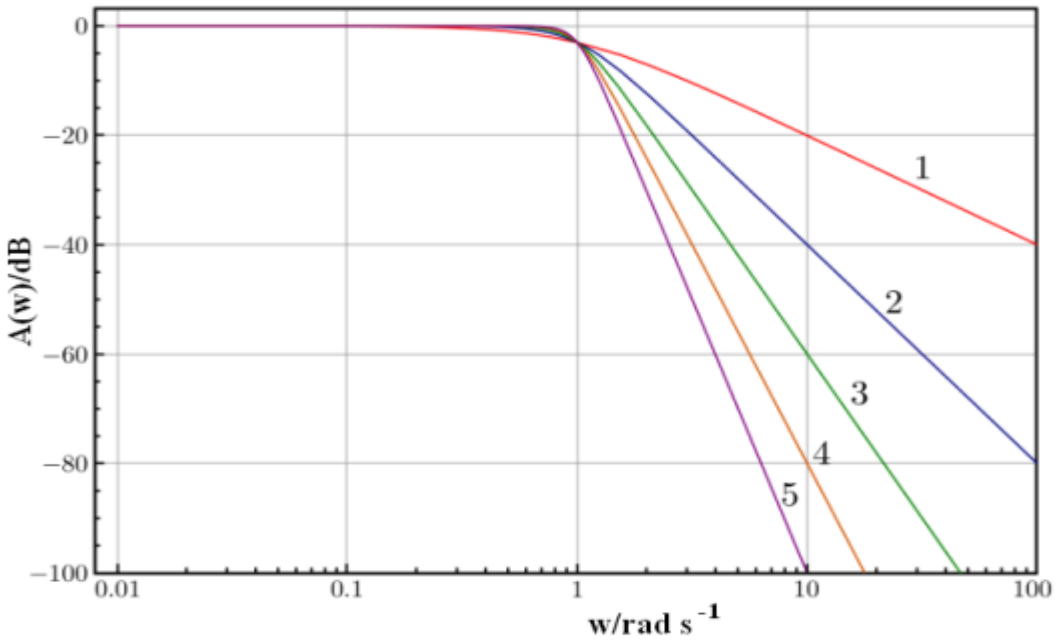

Figure 4. 3. Frequency response of low pass Butterworth filter according to order

The order of the filter is selected as three which is formed by one odd order topology and an even topology by cascading.

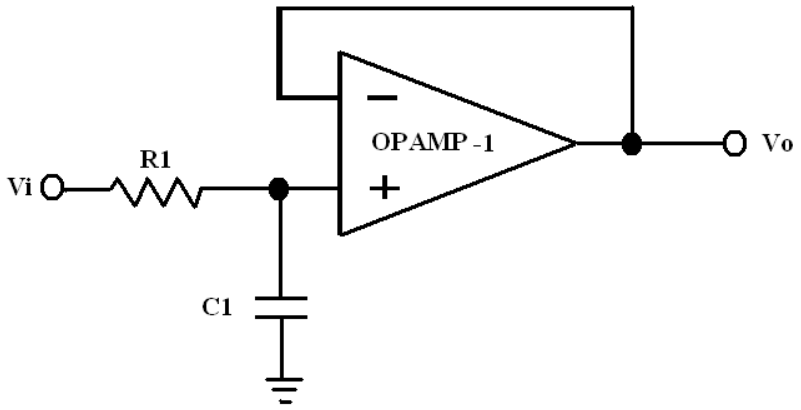

Figure 4. 4. First stage or first order of the total topology

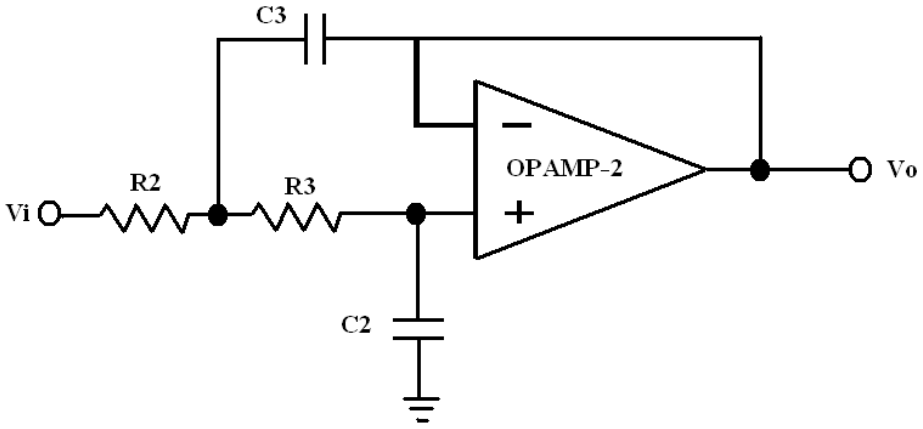

Figure 4. 5. Second stage or 2nd order of the total topology

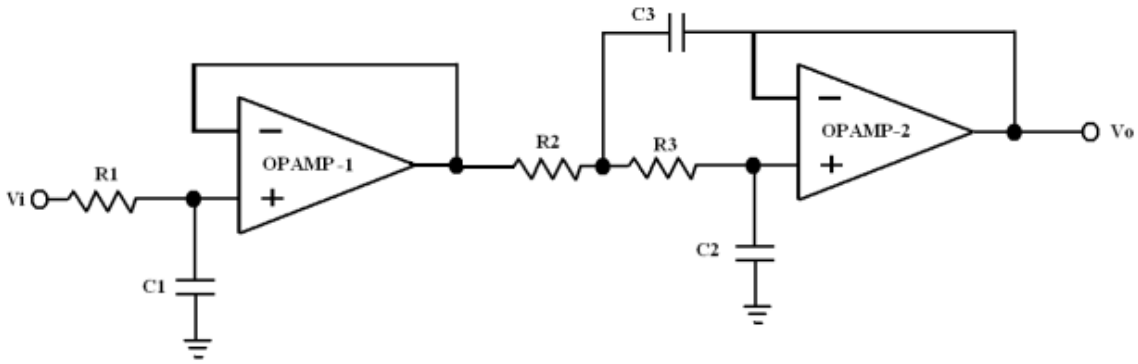

Figure 4. 6. Total topology of the 3rd order low pass Butterworth filter with Sallen-Key topology

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{1}{(s+1)(s^2+s+1)} \tag{4.1}$$

$$w_c = 2\pi f_c \tag{4.2}$$

3rd order Butterworth polynomial is:

$$H(s)=\frac{V_0(s)}{V_i(s)}=\frac{1}{(1+\frac{s}{w_c})((\frac{s}{w_c})^2+\frac{s}{w_c}+1)} \tag{4.3}$$

So, from the circuit transfer function;

$$\frac{2}{w_c}=C_1R_1+C_2R_2+C_3R_3 \tag{4.4}$$

$$w_c=\frac{2}{C_1R_1+C_2R_2+C_3R_3} \quad \text{and} \quad w_c=2\pi f_c$$

Cutoff frequency with ideal OPAMP is:

$$f_c=\frac{1}{\pi(C_1R_1+C_2R_2+C_3R_3)} \tag{4.5}$$

The topology of the filter circuit is formed by two cascading structures. First stage is 1st order and the second stage is 2nd order low pass active filter. In the lower module, values of the OPAMP (bandwidths, gains, output resistances, layout areas and power consumptions) are optimized with the MOS technology.

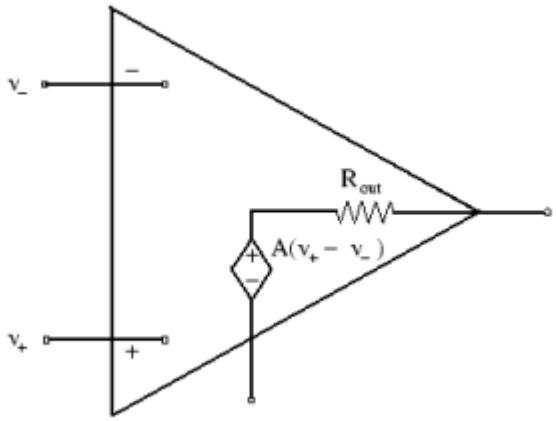

Figure 4. 7. Non ideal OPAMP model in this work

Input resistance of the OPAMP is not included and in the analysis since MOS technology is used. Output resistances and gains of two OPAMPs have maximum and minimum values in order to get implementable results.

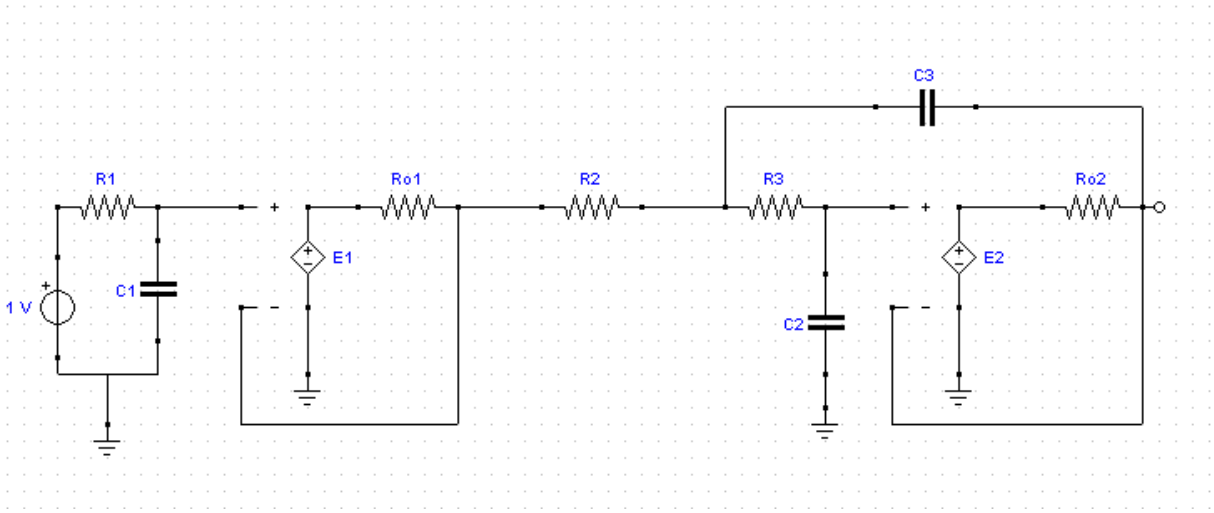

Figure 4. 8. Non ideal circuit schematic of 3rd order low pass Butterworth filter

$$H(s)=\frac{V_0(s)}{V_i(s)}=\frac{1}{(s+1)(s^2+s+1)} \tag{4.1}$$

It is equal to:

$$H(s)=\frac{V_0(s)}{V_i(s)}=\frac{1}{s^3+2s^2+2s+1} \tag{4.1}$$

If we scale this formulas by

$$s \to \frac{s}{w_c} \tag{4.7}$$

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{w_c^3}{s^3 + 2w_c s^2 + 2w_c^2 s + w_c^3} \tag{4.8}$$

With (4.6):

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{K_3 Z_2 s^2 + K_3 Z_1 s + K_3 Z_0}{s^3 + \frac{K_2}{K_3} s^2 + \frac{K_1}{K_3} s + \frac{K_0}{K_3}} \tag{4.9}$$

Zeros appear because of the non-ideality of the OPAMPs.

Matching the coefficients between the equation 4.8 and equation 4.9, angular frequencies are:

$$2w_c = \frac{K_2}{K_3} \tag{4.10}$$

$$2w_c{}^2 = \frac{K_1}{K_3} \tag{4.11}$$

$$w_c{}^3 = \frac{K_0}{K_3} \tag{4.12}$$

If we give different index parameter to the three different w_c s :

$$2w_{cA} = \frac{K_2}{K_3} \tag{4.13}$$

$$2w_{cB}{}^2 = \frac{K_1}{K_3} \tag{4.14}$$

$$w_{cC}{}^3 = \frac{K_0}{K_3} \tag{4.15}$$

In ideal form, it should be

$$w_{cA} = w_{cB} = w_{cC} \tag{4.16}$$

Also with the equation

$$w_c = 2\pi f_c \tag{4.2}$$

The cutoff frequency of the Low-pass Butterworth filter should be:

$$f_{cA} = f_{cB} = f_{cC} \tag{4.17}$$

$$K_0 = \mathrm{E2\ E1\ +\ E1\ +\ E2\ +1} \tag{4.18}$$

$$K_1 = \mathrm{E2\ E1\ C1\ R1\ +\ E1\ C1\ R1\ +E2\ C1\ R1\ +\ C1\ R1\ +\ E2\ E1\ C2\ R3\ +\ E2\ E1\ C2\ R2}$$
$$\mathrm{E1\ +\ E2\ E1\ C1\ R1\ +\ E1\ C1\ R1\ +E2\ C1\ R1\ +\ C1\ R1\ +\ E2\ E1\ C2\ R3\ +\ E2\ E1\ C2\ R2\ +}$$
$$\mathrm{E1\ C2+R3\ +\ E1\ C2\ R2\ +\ E2\ C2Ro1\ +\ E2\ C2\ R3\ +\ E2\ C2\ R2\ +\ C2\ Ro1\ +\ C2\ R3\ +}$$
$$\mathrm{C2\ R2\ +\ E1+C3\ Ro2\ +\ E1\ C3\ R2\ +\ C3Ro2\ +\ C3\ Ro1\ +\ C3\ R2} \tag{4.19}$$

$$K_2 = \mathrm{E2\ E1\ C2\ C1\ R1\ R3\ +\ E2E1\ C2\ C1\ R1\ R2\ +\ E1\ C2\ C1\ R1\ R3\ +\ E1\ C2\ C1\ R1\ R2}$$
$$\mathrm{+\ E2\ C2\ C1\ R1\ Ro1\ +\ E2C2\ C1\ R1\ R3\ +\ E2\ C2\ C1\ R1\ R2\ +\ C2\ C1\ R1\ Ro1\ +}$$
$$\mathrm{C2\ C1\ R1+R3\ +\ C2\ C1\ R1\ R2\ +\ E1\ C3C1\ R1\ Ro2\ +\ E1\ C3\ C1\ R1\ R2\ +\ C3\ C1\ R1\ Ro2}$$
$$\mathrm{+\ C3\ C1\ R1+Ro1\ +\ C3\ C1\ R1\ R2\ +\ E2E1\ C3\ C2\ R2\ R3\ +\ E1\ C3\ C2\ R3\ Ro2\ +}$$
$$\mathrm{E1\ C3\ C2\ R2\ Ro2\ +\ E1\ +C3\ C2\ R2\ R3\ +\ E2\ C3\ C2R3\ Ro1\ +\ E2\ C3\ C2\ R2\ R3\ +}$$
$$\mathrm{C3\ C2\ Ro1\ Ro2\ +\ C3\ C2\ R3\ Ro2\ +\ C3\ C2\ R3\ Ro1\ +\ C3\ C2\ R2\ Ro2\ +\ C3\ C2\ R2\ R3} \tag{4.20}$$

$$K_3 = \text{E2 E1 C3 C2 C1 R1 R2 R3} + \text{E1 C3 C2 C1 R1 R3 Ro2} + \text{E1 C3 C2 C1 R1 R2 Ro2}$$
$$+ \text{E1 C3 C2 C1 R1 R2 R3} + \text{E2 C3 C2 C1 R1 R3 Ro1} + \text{E2 C3 C2 C1 R1 R2 R3} +$$
$$\text{C3 C2+C1 R1 Ro1 Ro2} + \text{C3 C2 C1R1 R3 Ro2} + \text{C3 C2 C1 R1 R3 Ro1} +$$
$$\text{C3 C2 C1 R1 R2 Ro2} + \text{C3 C2 C1 R1 R2 R3} \quad (4.21)$$

From the equations: (4.13), (4.14), (4.15) and (4.2)

$$2w_{cA} = \frac{K_2}{K_3} \quad 2w_{cB}{}^2 = \frac{K_1}{K_3} \quad w_{cC}{}^3 = \frac{K_0}{K_3}$$

And $w_c = 2\pi f_c$

$$f_{cA} = \frac{K_2}{4\pi K_3} \quad (4.22)$$

$$f_{cB} = \frac{1}{2\pi}\sqrt{\frac{K_1}{2K_3}} \quad (4.23)$$

$$f_{cC} = \frac{1}{2\pi}\sqrt[3]{\frac{K_0}{K_3}} \quad (4.24)$$

It should be, $f_{cA} = f_{cB} = f_{cC}$ (4.17)

Equation 4.17 has to be obtained in order to make a flat pass-band magnitude response in Butterworth filter. This equation is also used in fitness function, bigger difference among the f_{cA}, f_{cB} and f_{cC} means more increment at the cost. Added to this, ratio of the max resistance to min resistance and the ratio of the max capacitance to min capacitance are considered. These proportions are vital to make a Butterworth characterization.

4.2. Performing Hierarchical Genetic Algorithm

To find the optimized low-pass 3rd order Butterworth filter, Hierarchical Genetic Algorithm (in short HGA) performs with two modules. The upper module (UM) optimized resistances, capacitances, frequency, Butterworth characterization, and superficially OPAMP's power consumption and chip layout area. The lower module (LM) optimizes the OPAMP's chip layout area, power consumption and the bandwidth with transistor base using spice model of current technology.

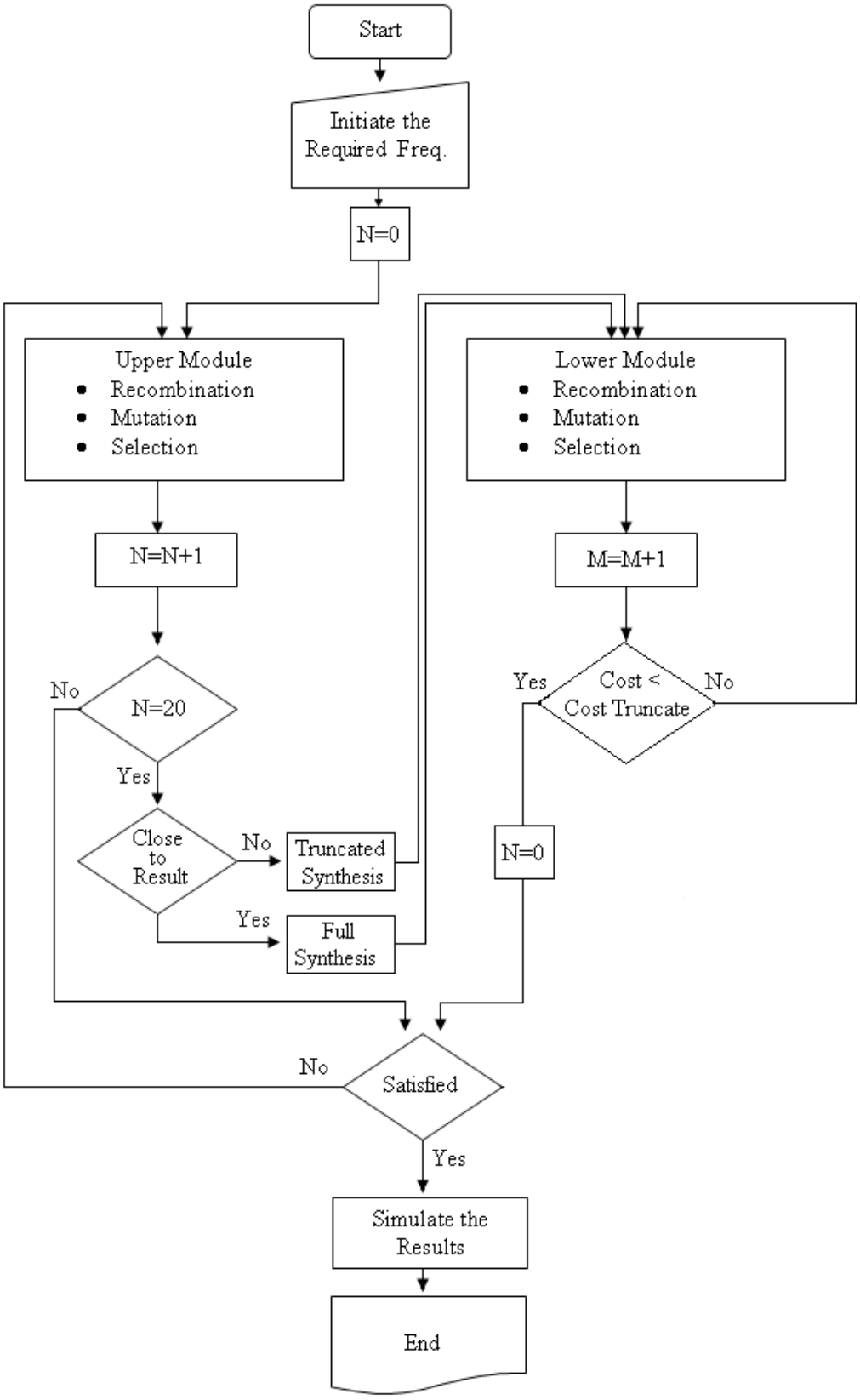

Figure 4. 9. The flow chart of the Proposed HGA

4.3. Construction of a Chromosome of an Individual with Encoded Genes

Before using the genetic algorithm, individuals in the population should be created. Also chromosomes of every individual have to be formed. This could be as a string of real numbers or, as is more typically the case, a binary bit string. This bit string will be referred to chromosomes from now on. A typical chromosome may look like the example as stated below:

Chromosome example: 10010101110101001010011101101110111111101

The individuals in this work formed an array which has 15 members or loci. Each member of string array has 32 bit binary cells. Every cell includes one characterization in the filter circuit.

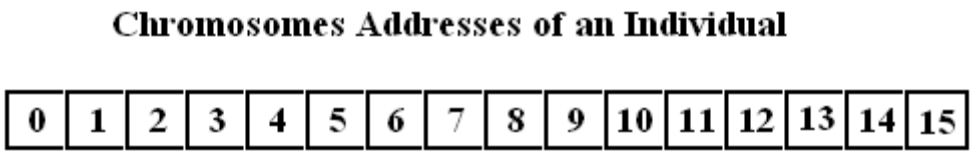

Figure 4. 10. General form of an individual in the HGA, an array with 12 elements

The chromosomes formed by fifteen genes which are named cells in the array.

1. The gene# 0 is symbolized of R1 resistance in the circuit schematic in figure 4.11.
2. The gene# 1 is symbolized of R2 resistance in the figure 4.11.
3. The gene# 2 is symbolized of R3 resistance in the figure 4.11.
4. The gene# 3 is symbolized of C1 capacitance in the figure 4.11.
5. The gene# 4 is symbolized of C2 capacitance in the figure 4.11.
6. The gene# 5 is symbolized of C3 capacitance in the figure 4.11.
7. The gene# 6 is symbolized of gain of the first OPAMP in the figure 4.11.
8. The gene# 7 is symbolized of output resistance Ro1 of the first OPAMP in the figure 4.11.
9. The gene# 8 is symbolized of gain of the second OPAMP in the figure 4.11.

10. The gene# 9 is symbolized of output resistance Ro2 of the second OPAMP in the figure 4.11.
11. The gene# 10 is symbolized of the power consumption of the OPAMP.
12. The gene# 11 is symbolized of the layout chip area of the OPAMP.
13. The gene# 12 is symbolized of ratio of the max R to min R in the circuit.
14. The gene# 13 is symbolized of ratio of the max C to min C in the circuit.
15. The gene# 14 is symbolized of Bandwidth of the OPAMP.
16. The gene# 15 is symbolized of Fitness value of the individual.

All these cells or genes are chromosomes which are formed by 32 bit binary string.

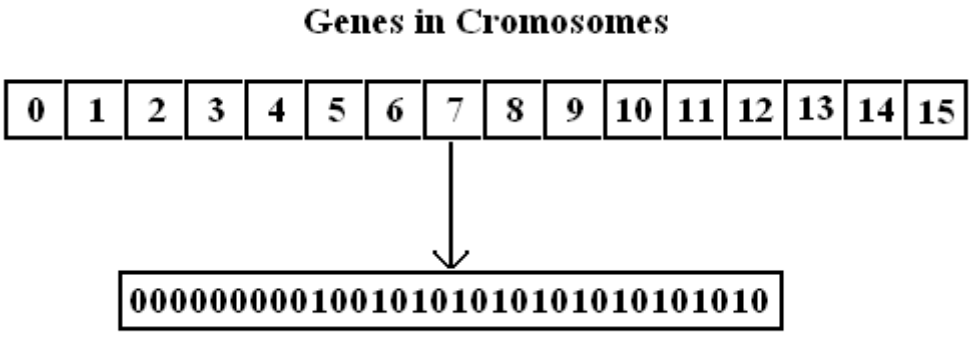

Figure 4. 11. One element in the array of an individual

4.4. Recombination

Parents in the population are mated randomly, after this mating; crossover is performed, between the couples. Two new children are born from a couple, hence from ten couples 20 children are breed. The population has thirty individuals before selecting the fittest ones.

4.4.1. Crossover

The crossover method in the proposed HGA is similar with double point crossover but it has some other developed tactics. The first gene where the crossover starts, the last gene where the crossover finished and crossover depth (genes which will be altered) are determined. For example, the number which determines the first gene that the crossover starts is 7. The number which determines the how many genes are altered in the crossover is 11. The genes of couple's chromosomes from the lowest significant 7th bit to 18th are altered. Thanks to the gene alteration two new children are born.

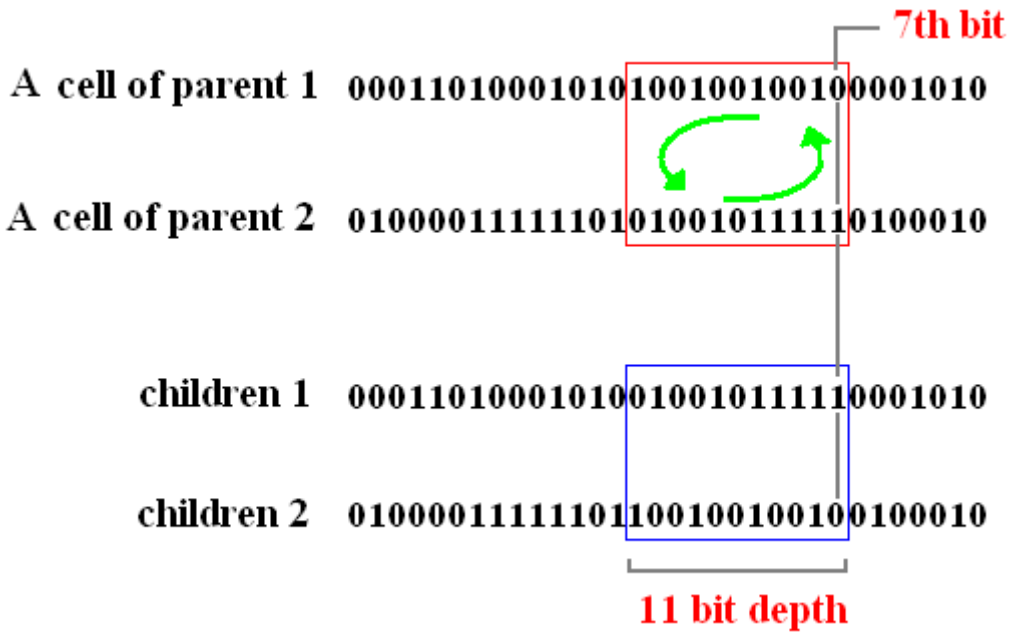

Figure 4. 12. Crossover operator in this work

4.4.2. Mutation

As we know, an individual in this work has 15 cells in the array; all cells have their own chromosomes. In the mutation operator, there are three random numbers which affect the process. First random number is bit "1" or "0". This bit determines what the mutated gene will be. The second number determines which cell will be mutated in the individual. The third number determines the bit or the gene which will be altered in the chromosome.

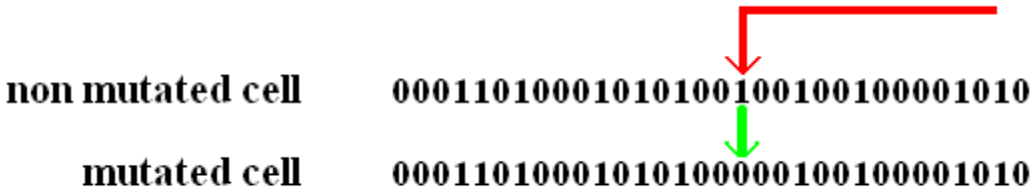

Figure 4. 13. Mutation operator in this work

4.5. Selection

In this work, selection is close to elitist selection. The fitness function is calculated for all individuals in the search space. The fitness function generates the cost using consistence among f_{cA}, f_{cB}, f_{cC} cutoff frequency values, max/min R value, max/min C value, power consumption value of the OPAMP and chip layout area value of the OPAMP.

All these values are normalized by their own normalization functions. After normalization, all these values get their own weights. Summation of these values after normalization and giving weight, cost value or fitness function value is formed. After calculating the fitness function value, an algorithm writes this value to the individual's 15^{th} gene, hence all individuals carry their own fitness value in their arrays. In selection operator, an algorithm sort all the population in the search space; after sorting, most fit 10 individuals are selected to survive and to breed in the next generations.

4.5.1. Fitness Function

The fitness function is formed by consistence among f_{cA}, f_{cB}, f_{cC} cutoff frequency values, max/min R value, max/min C value, power consumption value of the OPAMP, chip layout area value of the OPAMP. Power consumption value and chip layout area value are calculated from formulas of a model which is formed from 300 simulated results.

k_f =Frequency value in the cost

k_p = Power consumption value of OPAMP in the cost

k_a = Chip layout area value of OPAMP in the cost

k_R = Maximum resistance/minimum resistance value in the cost

k_C = Maximum capacitance/minimum capacitance value in the cost

W_1 =Weight of frequency value in the cost

W_2 = Weight of power value in the cost

W_3 = Weight of area value in the cost

W_4 = Weight of max resistance/min resistance ratio value in the cost

W_5 = Weight of max cap/min cap ratio value in the cost

$$Fitness\_Function = W_1((1+k_f)^2 - 1) + W_2 k_p + W_3 k_a + W_4 k_R + W_5 k_c \quad (4.25)$$

Normalization of cutoff frequency:

$$k_{f0} = \frac{|f_{cA} - wanted\_frequency|}{|wanted\_frequency\_tolerance|} \quad (4.26)$$

$$k_{f1} = \frac{|f_{cB} - wanted\_frequency|}{|wanted\_frequency\_tolerance|} \quad (4.27)$$

$$k_{f2} = \frac{|f_{cC} - wanted\_frequency|}{|wanted\_frequency\_tolerance|} \quad (4.28)$$

$$k_f = k_{f0} + k_{f1} + k_{f2} \quad (4.29)$$

Power Consumption:

E = Gain of the OPAMP

BW = Bandwidth of the OPAMP

Ro = Output resistance of the OPAMP

$$P_1 = \frac{1.5101}{0.25928 * E + 1.2106} \tag{4.30}$$

$$P_2 = 1.2182 * 10^{-7} * BW + 0.00034877 \tag{4.31}$$

$$P_3 = \frac{0.0073585}{5.8601 * 10^{-5} * Ro + 2.2895} \tag{4.32}$$

Power Consumption values from model:

$$P = \frac{P_1 + P_2 + P_3}{3} \tag{4.33}$$

Normalization of power values:

$$k_p = \frac{\left| P - powerconsump_{good} \right|}{\left| powerconsump_{good} - powerconsump_{bad} \right|} \tag{4.34}$$

OPAMP chip layout area values from model:

$$A_1 = 1.3354 * 10^{-16} * A * BW + 2.054 * 10^{-8} \tag{4.35}$$

$$A_2 = -2 * 10^{-13} * Ro + 4.3 * 10^{-8} \tag{4.36}$$

$$A = \frac{A_1 + A_2}{2} \tag{4.37}$$

Normalization of area values:

$$k_a = \frac{\left| A - Area_{good} \right|}{\left| Area_{good} - Area_{bad} \right|} \tag{4.38}$$

Max Resistance/Min Resistance ratio:

$$Rmm = \frac{R\max}{R\min} \tag{4.39}$$

Normalization of resistance ratio:

$$k_R = \frac{\left| R_{mm} - Rmm_{good} \right|}{\left| Rmm_{good} - Rmm_{bad} \right|} \tag{4.40}$$

Max Capacitance/Min Capacitance ratio:

$$Cmm = \frac{C\max}{C\min} \tag{4.41}$$

Normalization of capacitance ratio:

$$k_C = \frac{\left| C_{mm} - Cmm_{good} \right|}{\left| Cmm_{good} - Cmm_{bad} \right|} \tag{4.42}$$

When, k_f, k_p, k_a, k_R and k_C are combined with their weights fitness function is formed.

$$Fitness\_Function = W_1((1+k_f)^2 - 1) + W_2 k_p + W_3 k_a + W_4 k_R + W_5 k_c \quad (4.43)$$

4.6. Communication Space Between Master and Slave Module

While the HGA performs, master module (UM) and slave module (LM) communicate by using communication space in order to realize individuals and locus migration. In this space, there are two text files which include required values of the OPAMP (gain, output resistance and bandwidth) and optimized results (OPAMP's chip layout area and chip power consumption) written by the slave module. The UM writes out the required OPAMP's features to the "non_sim.txt" file and the LM writes out this OPAMP's features area and power consumption to the "sim.txt" file.

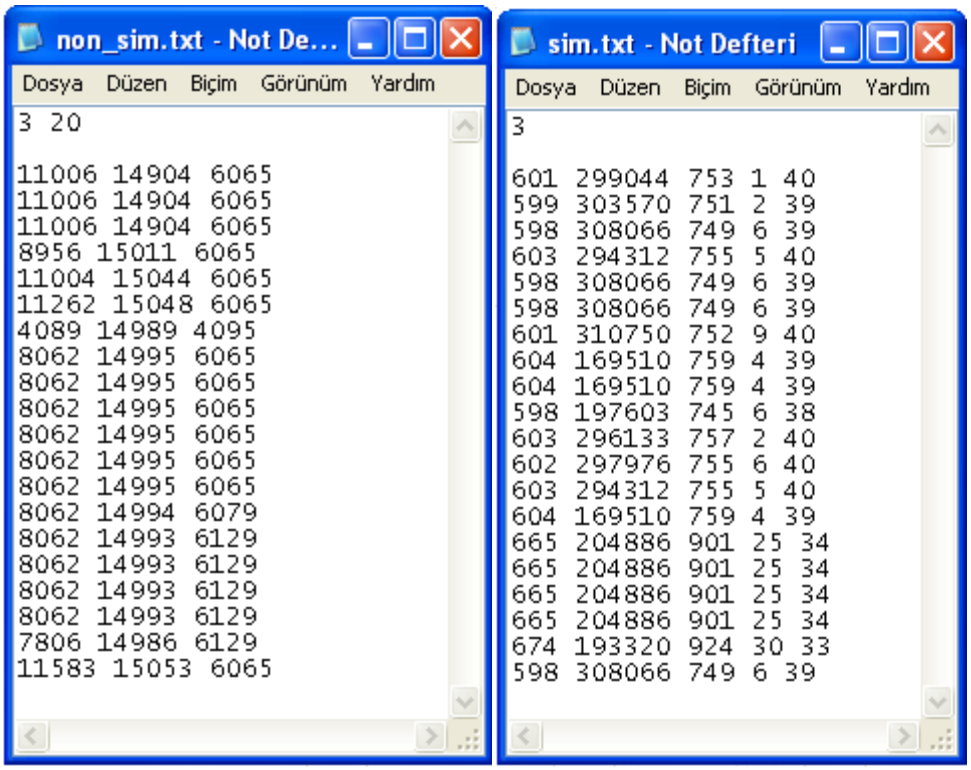

Figure 4. 14. "non_sim.txt" and "sim.txt" files in communication space

As we see in Figure 4.15, there are two numbers at first line and three columns in the “non_sim.txt” file. First number in the first line indicates how many times master and slave modules communicate. Second number in the first line indicates the number of the individuals that the UM send to the LM and in the file this number is 20. The three columns consist of the OPAMP’s gain, OPAMP’s output resistance and OPAMP’s bandwidth, respectively.

In the “sim.txt” file the number in the first line indicates how many times the UM and the LM communicate like the first number in the first line in the “non_sim.txt” file. The five columns consist of the OPAMP’s gain, OPAMP’s output resistance, OPAMP’s bandwidth, OPAMP’s chip power consumption values and OPAMP’s chip layout areas, respectively. The order between the UM and the LM is determined by the first number both in two text files

5. EXAMPLES AND RESULTS

In the first example, upper module worked alone. The simulation and optimization job which is done from lower module is also done by upper module with using internal model. The results of this module are within 60% of SPICE simulation results. The results of an example are presented in Table 5.1. R_1, R_2 and R_3 values were forced to 1000Ω and $C_1 = 1000nF$ $C_2 = 500nF$ and $C_3 = 2000nF$. The desired frequency was 10000 Hz and the tolerance is 10%.

The initial values are:

The initial frequency = 159.2297 Hz

9000 Hz < Desired frequency < 11000 Hz

The results are given in the Table 5.1.

Table 5. 1. The result table of first example, model based GA

<table>
<tr><td colspan="6">The calculated frequency = 9517.90332 Hz</td></tr>
<tr><td colspan="3">Total Process Time</td><td colspan="3">Generation Number</td></tr>
<tr><td colspan="3">923 second</td><td colspan="3">47</td></tr>
<tr><td>R_1</td><td>R_2</td><td>R_3</td><td>C_1</td><td>C_2</td><td>C_3</td></tr>
<tr><td>$62\,\Omega$</td><td>$72\,\Omega$</td><td>$107\,\Omega$</td><td>$251\,nF$</td><td>$105\,nF$</td><td>$340\,nF$</td></tr>
<tr><td colspan="2">Gain of OPAMP</td><td colspan="2">Bandwidth of OPAMP</td><td colspan="2">R_{OUT} of OPAMP</td></tr>
<tr><td colspan="2">6401</td><td colspan="2">14276</td><td colspan="2">33997</td></tr>
<tr><td colspan="3">Power Consumption of OPAMP</td><td colspan="3">Chip Layout Area of OPAMP</td></tr>
<tr><td colspan="3">8mW</td><td colspan="3">35 μm^2</td></tr>
</table>

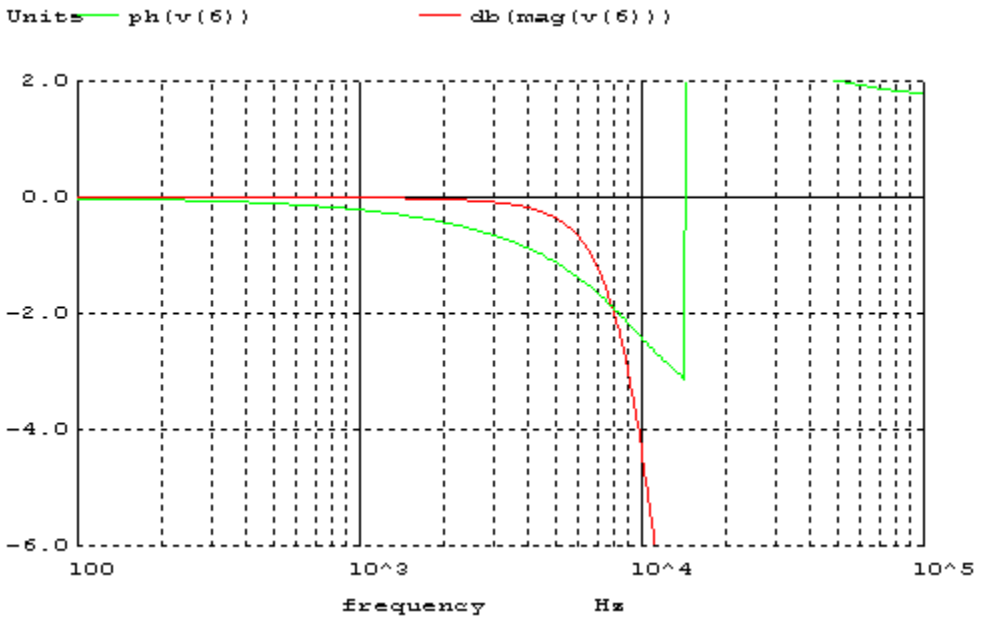

Figure 5. 1. Spice output of the *.cir file which is created by only the Upper Module

As we can see from the Figure 5.1, the values and the formulas can calculate the correct frequency response, using GA in the UM.

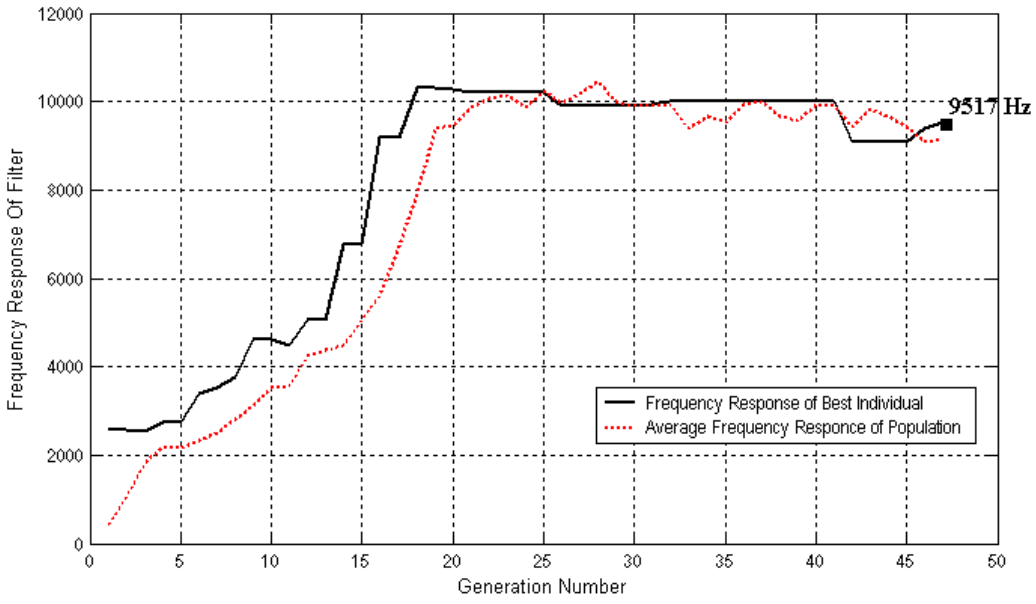

Figure 5. 2. Frequency response change of first example (model based GA)

In the Figure 5.2, we can observe changing of the frequency response by the generation number. Up to the 20th generation, it increases rapidly, from the 25th generation to the 40th generation, other values of the cost: chip layout area of the OPAMP, power consumption of the OPAMP, max Res./min Res. Ratio and (max cap)/(min cap) ratio converge. After 40th generation GA tries to fix and optimization details.

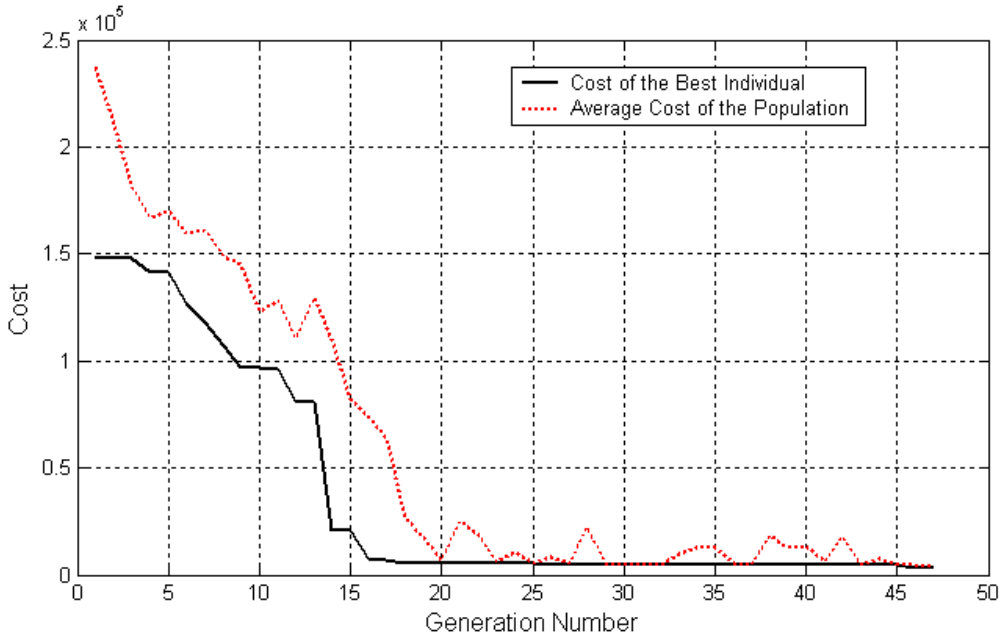

Figure 5. 3. Cost Change with generation number for first example (model based GA)

Like the frequency graphic in figure 5.3, cost also rapidly decreases up to the 20th generation. After the 20th generation, it starts to converge slowly, and at 46th generation GA finds the satisfied cost.

As a result, if the UM works alone with its inner OPAMP optimization formulas, optimization takes very little time but the quality of the result and realization of OPAMP is not satisfied.

In second example, HGA runs with conventional method. In other words, upper module makes communication and wants OPAMP optimization from lower module in every step or in every new generation. The lower module makes complete simulation and optimization of the required OPAMPs upon every demand of the UM.

The results of an example are provided in Table 5.2:

R_1, R_2 and R_3 values are forced to 1000Ω and the $C_1 = 1000nF$, $C_2 = 500nF$ and $C_3 = 2000nF$. The desired frequency is 10000 Hz and the tolerance is 10%.

The initial values are:

The initial frequency = 159.2297 Hz

9000 Hz < Desired_frequency < 11000 Hz

The results are given in the Table 5.2.

Table 5. 2. The result table of the second example, standard HGA

<table>
<tr><td colspan="6">The calculated frequency = 9975.095703 Hz</td></tr>
<tr><td colspan="3">Total Process Time</td><td colspan="3">Generation Number</td></tr>
<tr><td colspan="3">128456.23 second (35.68 hr)</td><td colspan="3">71</td></tr>
<tr><td>R_1</td><td>R_2</td><td>R_3</td><td>C_1</td><td>C_2</td><td>C_3</td></tr>
<tr><td>254Ω</td><td>42Ω</td><td>112Ω</td><td>$62\,nF$</td><td>$100\,nF$</td><td>$511\,nF$</td></tr>
<tr><td colspan="2">Gain of OPAMP</td><td colspan="2">Bandwidth of OPAMP</td><td colspan="2">R_{OUT} of OPAMP</td></tr>
<tr><td colspan="2">3356</td><td colspan="2">14962</td><td colspan="2">4502</td></tr>
<tr><td colspan="3">Power Consumption of OPAMP</td><td colspan="3">Chip Layout Area of OPAMP</td></tr>
<tr><td colspan="3">2mW</td><td colspan="3">$34\ \mu m^2$</td></tr>
</table>

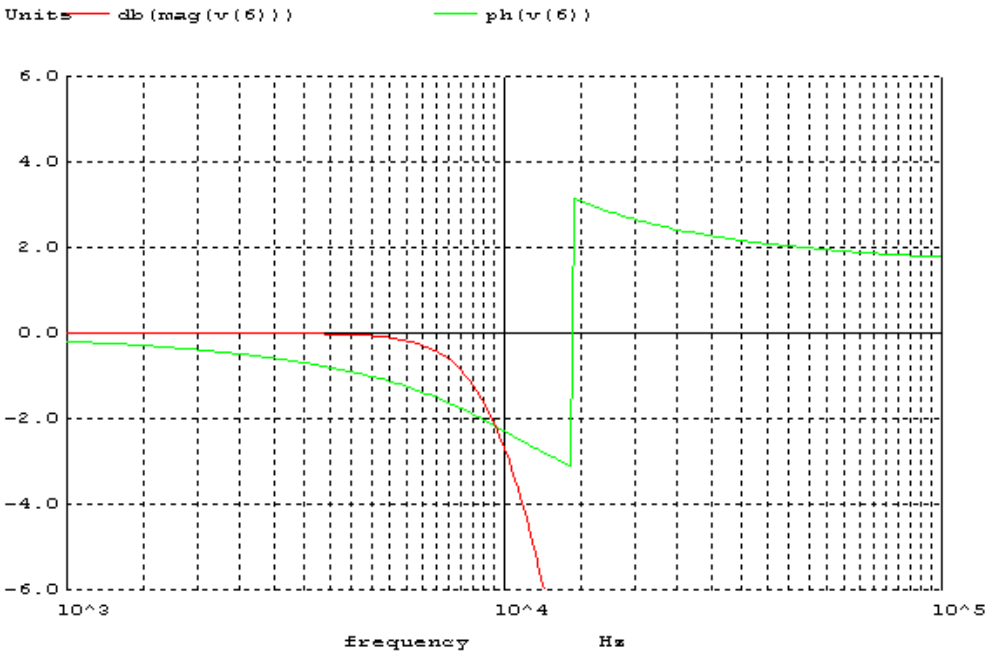

Figure 5. 4. Spice output of the cir file which is created by the HGA

The flat band of the low-pass Butterworth is best in this example as we see clearly in Figure 5.4.

As we see in the Figure 5.5, there is a small gap between the best individual and the rest of the population.

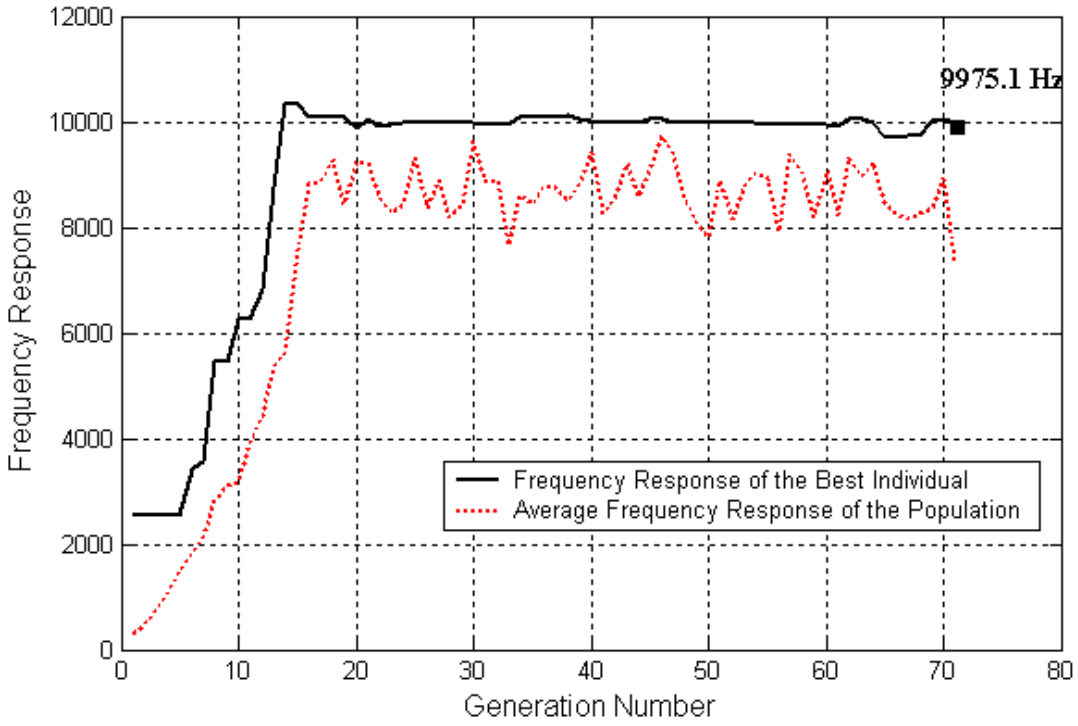

Figure 5. 5. The graphic of the cutoff frequency of the low pass Butterworth filter with generation number, formed by standard HGA

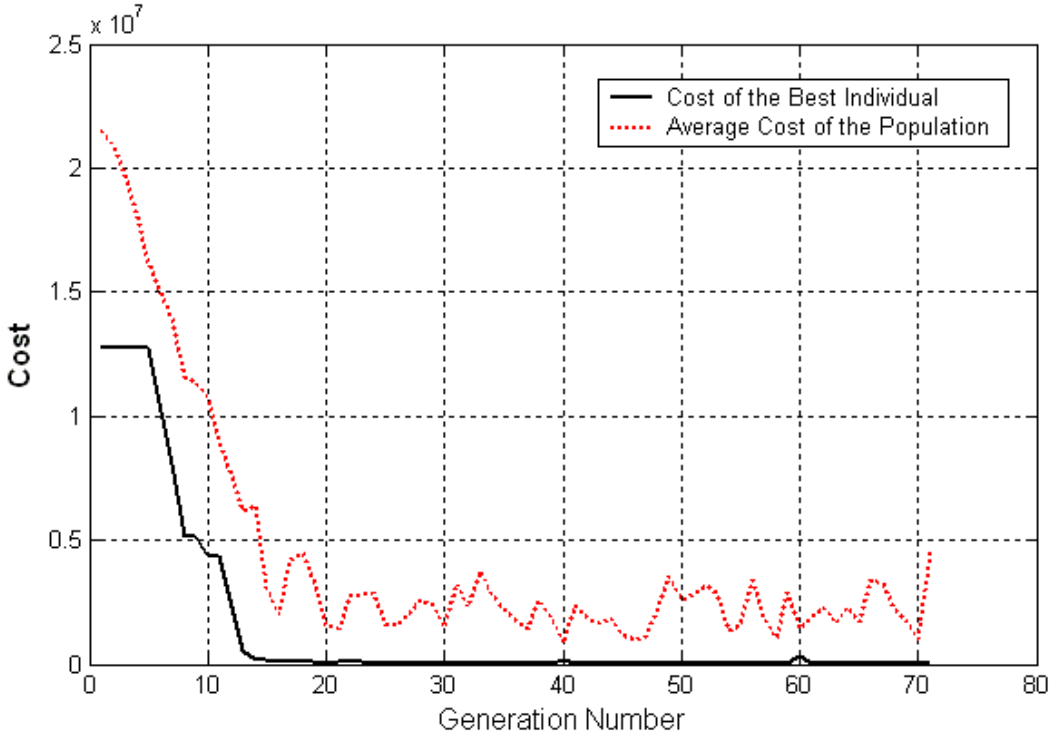

Figure 5. 6. The graphic of the cost with generation number formed by standard HGA

Like in the Figure 5.5, in figure 5.6, there is a gap again between average of the population and the best individual. The UM communicate with the LM in every generation by sending 20 elite individual among the 30 individuals; hence 10 individuals do not go to the LM.

In third example, the proposed HGA is run. The upper Module processes for 20 iterations and sends features of the required OPAMPs to the lower module by writing them in "non_sim.text" in the communication space and waits for the results that will be sent back by the lower module. When the upper module send the required OPAMP's features, the lower module gets the features (gain, bandwidth and output resistance) of OPAMP and starts to optimize and design these OPAMPs. The lower module does not run whole iterations; it runs for a limited number of iterations and gives back the results to the UM. This loop sustains until the elite individual in the population gets close to satisfied result. The UM controls the distance between the best individual and the desired individual. If the population, especially the elite individual starts to get close to the solution, the UM wants from the LM to make full optimization. If the population starts to go far away from the satisfied result, UM wants from LM to make short-coming optimization again.

The results of an example stated below:

R_1, R_2 and R_3 values are forced to 1000Ω and the $C_1 = 1000nF$, $C_2 = 500nF$ and $C_3 = 2000nF$. The wanted frequency is 10000 Hz and the tolerance is 10%.

The initial values are:

The initial frequency = 159.2297 Hz

9000 Hz <Desired frequency < 11000 Hz

The results are given in the Table 5.3.

Table 5. 3. The result table of the third example, proposed HGA

The calculated frequency = 9930.5302 Hz					
Total Process Time			**Generation Number**		
2143 second			61		
R_1	R_2	R_3	C_1	C_2	C_3
$122\,\Omega$	$98\,\Omega$	$240\,\Omega$	$126\,nF$	$45\,nF$	$240\,nF$
Gain of OPAMP		**Bandwidth of OPAMP**		**R_{OUT} of OPAMP**	
2517		14895		1191	
Power Consumption of OPAMP			**Chip Layout Area of OPAMP**		
2mW			34 μm^2		

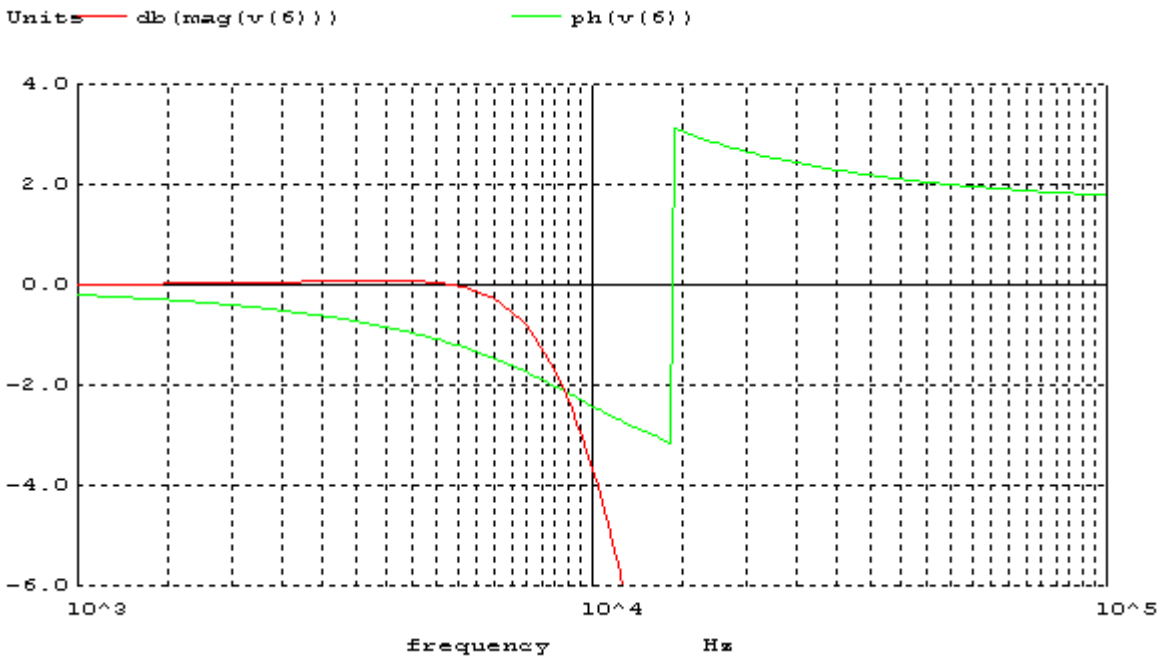

Figure 5. 7. Spice output of the cir file which is created by the proposed HGA

As we see in the figure 5.7 the frequency response is near 9905.415 Hz. In addition when we consider the features of the OPAMP, solution is implementable and realizable.

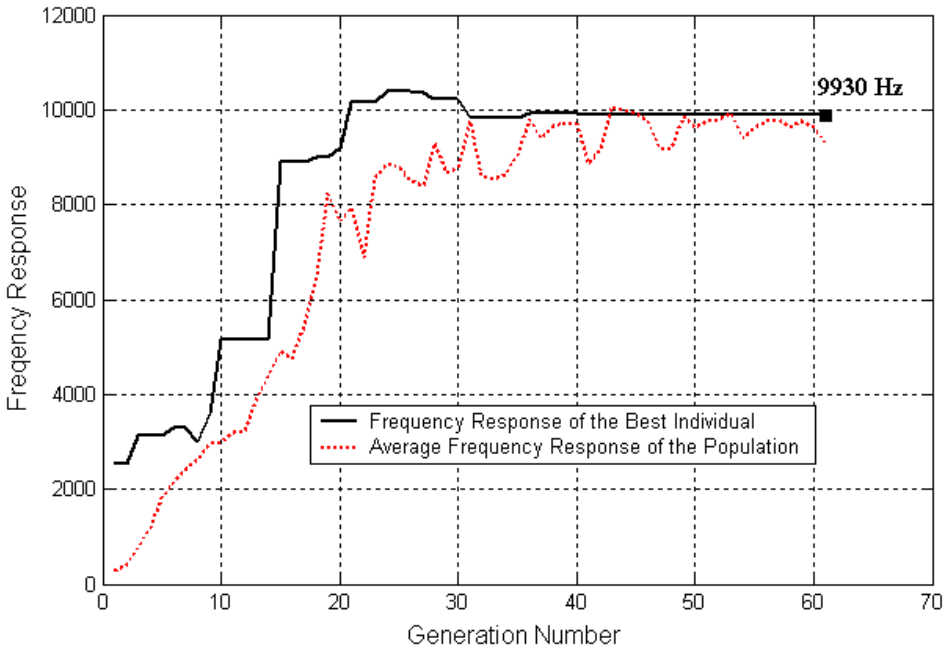

Figure 5. 8. Frequency response change that is created by the proposed HGA

In Figure 5.8, the frequency response of elite individual and average of population is shown. The dotted line is the average frequency response of the population. We can easily see the sudden peaks in dotted line which are caused by the mutations.

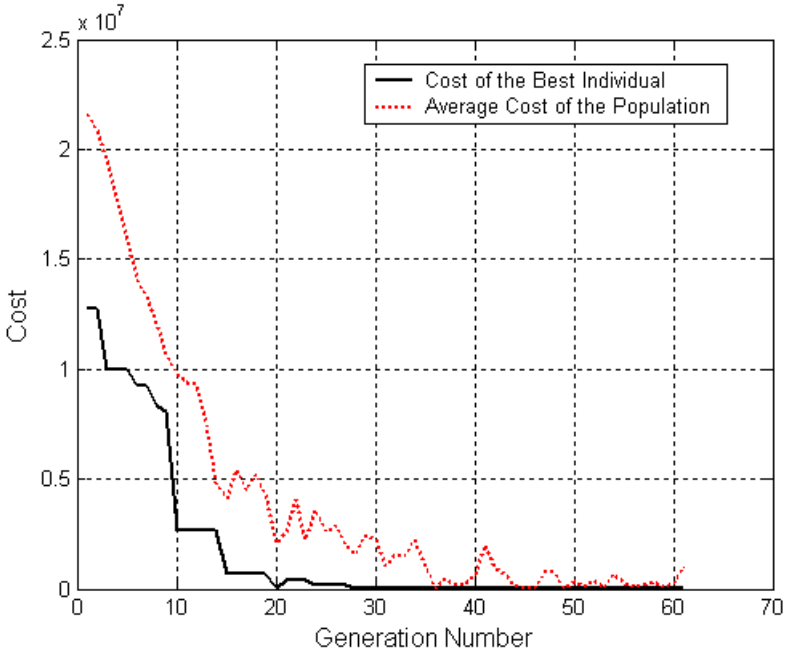

Figure 5. 9. Cost change that is created by the proposed HGA

Parallel with the frequency response graphic, the peaks in this graphic overlap peaks in the frequency response graphic because the effects of mutations.

5.1. Comparison

According to the examples adequate data are obtained in order to make comparison among the model based GA, standard HGA and the two-layered proposed HGA.

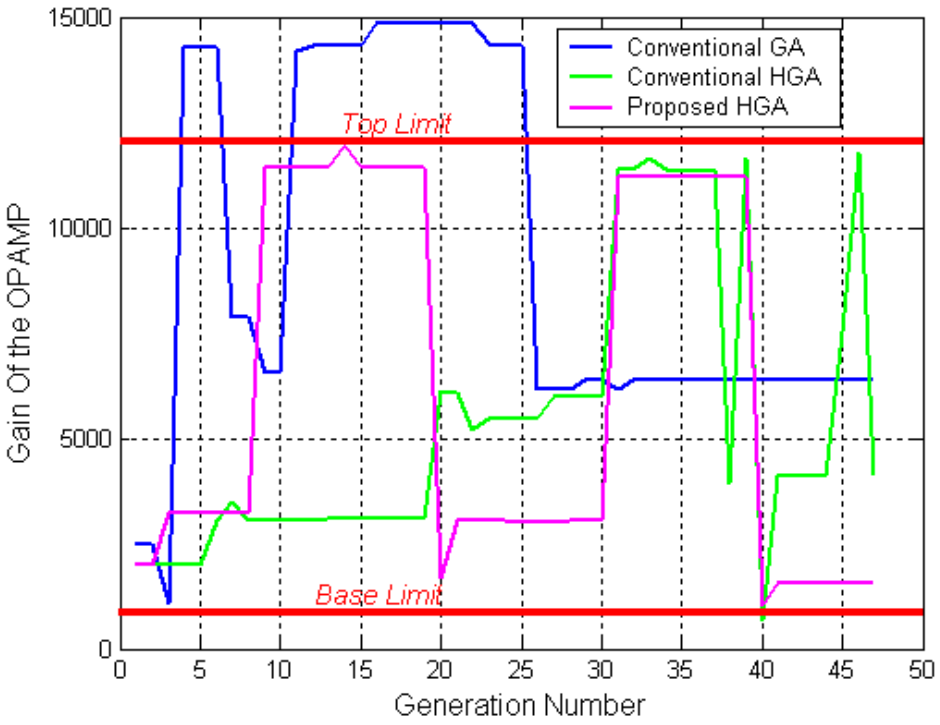

Figure 5. 10. Gain of the OPAMP from result of the test

As we see, in Figure 5.10. Standard HGA and the proposed HGA have the result (gain of the OPAMP) in the maximum and minimum implementation limit. On the other hand model based GA has poor result to implement the design.

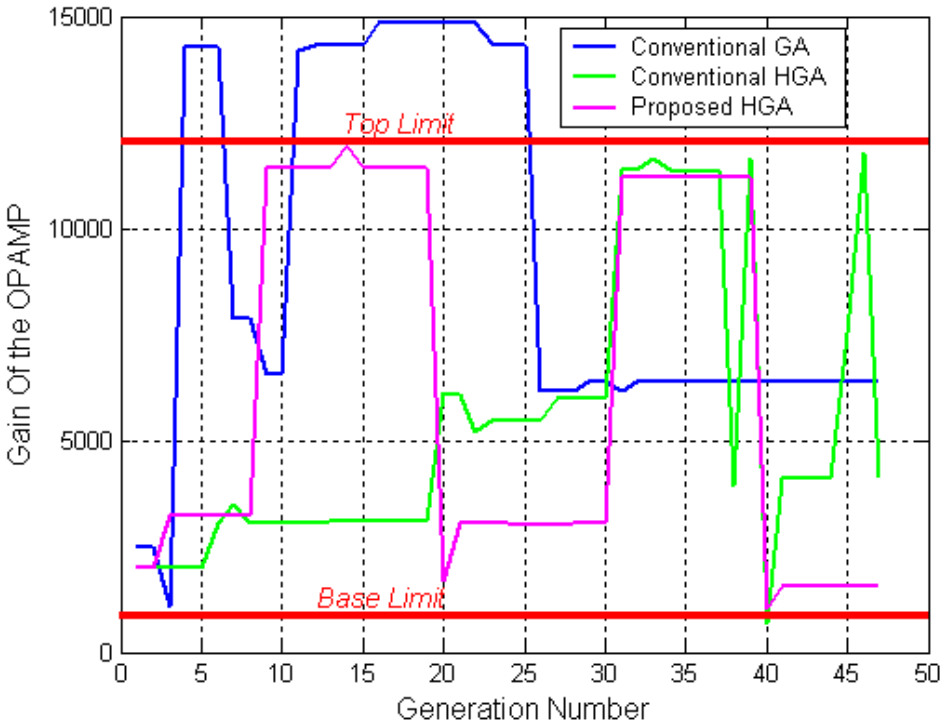

Figure 5. 11. Gain of the OPAMP form result of the test

Like in the previous graphic, standard HGA and proposed HGA have same quality of the solution.

Table 5. 4. Total process time comparison

Total Process Time (second)	
Model based GA	0.431
Standard HGA	128456.23 (35.68 hr)
Two-layered HGA	2143

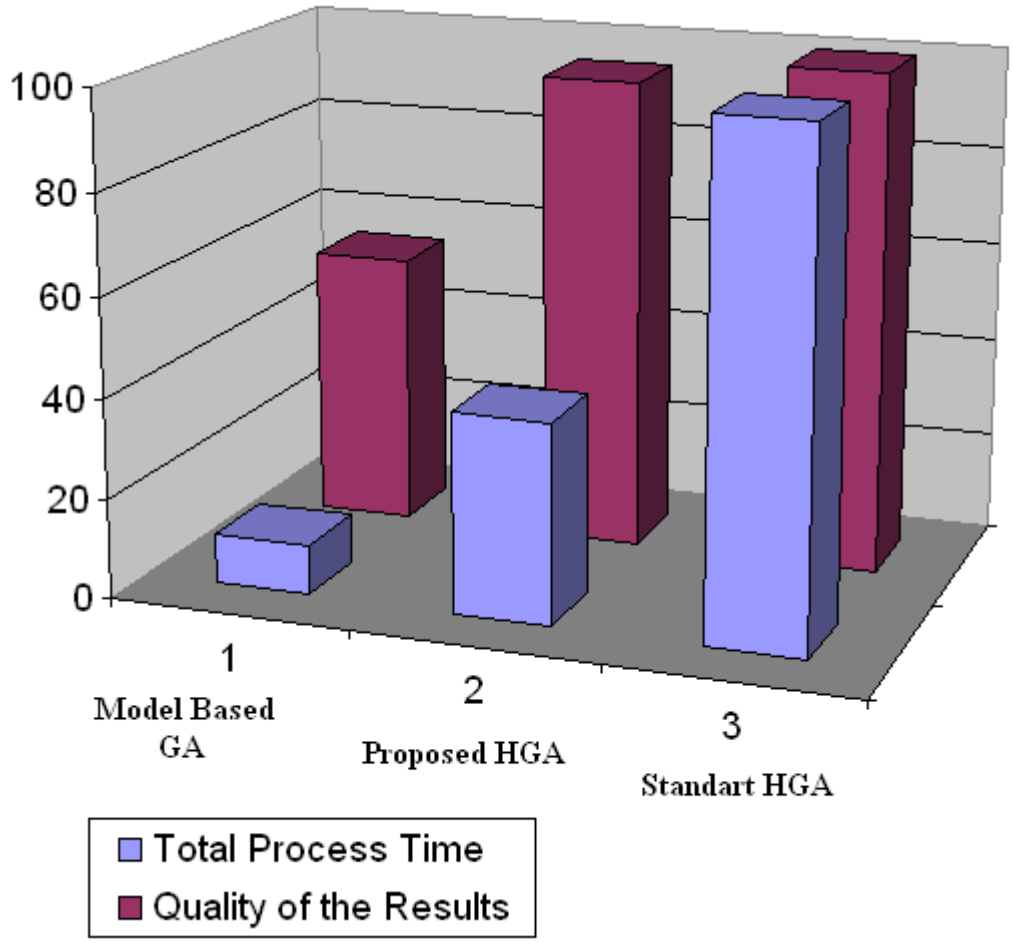

Figure 5. 12. Graphical comparison of three applications.

It clearly seems that two-layered proposed HGA can solve problems in the examples with close computation time to GA and same result quality as in standard HGA.

6. CONCLUSION

Two-layered proposed Hierarchic Genetic Algorithms are used to optimize MOS based complex integrated circuits in 3rd order low-pass Butterworth filter in order to measure its performance. To measure the performance of the proposed HGA, the same example problem performs to model based GA and to standard HGA. After performing various tests to these three applications (model based GA, standard HGA and proposed HGA) results prove that the proposed HGA has advantages. In the performance tests, three applications are operated to optimize and design low-pass filter with the same frequency responses. At the end of every test, total process time, calculated filter frequency response, Butterworth characterization and features of OPAMPs are observed. When we consider these control parameters, which are laid out from optimizations, proposed two-layered HGA can gives satisfied outcomes. The proposed HGA performs same quality with the standard HGA and close to process time of model based GA. The quality of the results is determined by the LM according to realization and implementation in current MOS technology. Added to these, thanks to the modular architecture, modules and the technology can easily change in order to solve different complex problems.

All in all, two-layered proposed HGA can make optimization and solve complex large-scale problems with approximately process time in GA and same result quality in standard HGA.

7. REFERENCES

1. Worapradya K and S. Pratishthananda, "*Fuzzy Supervisory PI controller Using the Hierarchical Genetic Algorithms*", Control Conference, 2004. 5th Asian**,** Volume: 3, p.p: 1523- 1528 Vol.3 , 20-23 July 2004

2. Wiles J. and B. Tonkes, "*Visualization of Hierarchical Cost surface for Evolutionary Computing*", Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on**,**Volume :1, p.p 157-162, 12-17 May 2002

3. Bo H. and W. Lixin, "*The Application of Genetic Algorithm in Multi-Hierarchical Complex Mechanical Structure Scheme Innovation Design*", Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD '06. 7th International Conference on**,** No: 9487103, p.p: 1-6, 17-19 Nov. 2006

4. Fan Z., D. Erik Goodman, J. Wang, R. Rosenberg, K. Seo and J. Wu, "*Hierarchical Evolutionary Synthesis of MEMS*", Evolutionary Computation, 2004. CEC2004. Congress on**,** Volume: 2, p.p 2320- 2327, 19-23 June 2004

5. Zhou Z., Y. S. Ong and P. B. Nair, "*Hierarchical Surrogate-Assisted evolutionary Optimization Framework*", Evolutionary Computation, 2004. CEC2004. Congress on**,** Volume: 2, p.p: 1586- 1593, 19-23 June 2004

6. Wang C., Y. C. Soh, H. Wang and H. Wang, *"A Hierarchical Genetic Algorithm for Path Planning in a Static Environment with Obstacles*" Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on**,** Volume: 3, p.p: 1652- 1657, 2002

7. Lu Y., H. Zhang, W. Zhang, "The *Application of Hierarchical Evolutionary Approach for Sleep Apnea Classification*", Machine Learning and Cybernetics, 2005.

Proceedings of 2005 International Conference on**,** Volume: 6, p.p:3708-3712, 18-21 Aug. 2005

8. Isaacs A., T. Ray, W. Smith, "A *Hybrid Evolutionary Algorithm With Simplex Local Search*" Evolutionary Computation, 2007. CEC 2007. IEEE Congress on**,** p.p: 1701-1708 , 25-28 Sept. 2007

9. Lim D., Y. Ong, Y. Jin, B. Sendholf, B. Lee, *"Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing*", Future Generation Computer Systems , Volume 23 , P: 658-670, 4 May 2007

10. Herrera F., M. Lozano, C. Moraga, *"Hierarchical Distributed Genetic Algorithms"* Evolutionary Computation, 2000. Proceedings of the 2000 Congress on**,** Volume: 1, p.p: 272-276, 2000

11. Fan. Z., J. Hu, K. Seo, D. E. Goodman, R. C. Rosenberg and B. Zhang, "*A Bond Graph Representation Approach for Automated Analog Fitler Design*", Gecco 2001, p:1253, 2001

12. Ma M. and L. Zhang, *"Optimization a fuzzy network with a hierarchical genetic algorithm", Intelligent Sytesms, Vol*. 11 No. 3, pp. 76-84, June 1996

13. Olivera. C. M A., L. A. N. Lorena, S. Stephani and A. J. Preto
"A Hierarchical Fair Competition Genetic Algorithm for Numerical Optimization", Gecco 2002,

14. Gulsen. M. and A. E. Smith, "A Hierarchical Genetic Algorithm for System Identification and Curve Fitting with a Supercomputer Implementation", 1031 Benedum Hall University of Pittsburgh,USA, 1998

15. Kunicka A. and H. Kwasnicka, “Hypermarket – an evolutionary paths planner”, Evolutionary Computation and Global Optimization 2006, ISSN 0137-2343; pp. 247-256, June 2, 2006

16. Gielen G. G. and W. Sansen, “Symbolic Analysis for Automated Design of Analog Integrated Circuits”, Boston, MA: Kluwer, 1991.

17. Sussman G. J. and R.M. Stallman, “Heuristic Techniques in Computer-Aided Circuit Analysis,” IEEE Trans. Circuits and Systems, Vol. 22, 1975.

18. Harjani R., R.A. Rutenbar and L.R. Carey, “A Prototype Framework for Knowledge-Based Analog Circuit Synthesis,” Proc. 24th Design Automation Conf., 1987.

19. Ochotta E.S., R.A. Rutenbar and L.R. Carley, \Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," IEEE Trans. Computer-Aided Design, Vol. 15, pp. 273{294, 1996.

20. John H. H., “Adaptation in Natural and Artificial Systems”, Univ. of Michigan Press, Ann Arbor, 1975.

21. Harjani R., R.Rutenbar, and L. R. Carley, “OASYS: A framework for analog circuit synthesis,” *IEEE Trans. Computer-Aided Design*, Vol.8, pp. 1247-1265, Dec. 1989.

22. El-Turky F. and E. Perry, “BLADES: “An artificial intelligence approach to analog circuit design,” *IEEE Trans. Computer-Aided Design*, Vol. 8, pp. 680–692, June 1989.

23. Degrauwe M., “IDAC:An interactive design tool for analog CMOS circuits,” *IEEE J. Solid-State Circuits*, Vol. SC-22, pp. 1106–1116, Dec. 1987.

24. Stehr G., H. E. Graeb, and K. J. Antreich, “Analog performance space exploration by normal-boundary intersection and by Fourier-Motzkin elimination,” *IEEE Trans. Computer-Aided Design*, Vol. 26, 1733-1748, Oct. 2007.

25. Ingber L., “Very fast simulated re-annealing,” *Mathl. Comput. Modelling*, Vol. 12, pp. 967-973, 1989.

26. Michalewicz Z., “*Genetic Algorithms + Data Structures = Evolution Programs*,” Springer, 1998.

27. Yuan J., N. Farhat, and J. V. der Spiegel, “GBOPCAD: A synthesis tool for high performance gain-boosted opamp design,” *IEEE Trans. Circuits Syst. I, Fundamental Theory and Applications*, Vol.52, pp. 1535-1544, Aug. 2005.