A Design Method in CMOS Operational Amplifier Optimization Based on Adaptive Genetic Algorithm

JIANHAI YU, ZHIGANG MAO
Microelectronics Center
Harbin Institute of Technology
No. 92, West Dazhi Street, Harbin
CHINA
yujianhai1977@sina.com

Abstract:- A new method based on adaptive GA(genetic algorithm) for optimizing the parameters of CMOS operational amplifier is presented in this paper. The synthesis of the Op-amp (operational amplifier) can be translated into multiple-objective optimization task, in which a large number of specifications have to be taken into account. Such as DC-gain, bandwidth of unity gain, phase-margin, power, noise and others. The feature of the method is that combining the skills of manual experience of analog circuits design and genetic algorithm, through adjusting the GA with the evolution process, the problems of convergence and multiple objective optimization tasks can be solved; and operational amplifiers for different use can be designed depending on various performance specifications. The results of groups of simulation show that this method can optimize the parameters of analog circuits accurately and efficiently.

Key-Words: - genetic algorithm (GA); CMOS analog circuit; operational amplifier; evolution, fitness, sizing

1 Introduction

The analog part of a complex mixed signal system is often small compared to the digital part. However, the design of the analog part is often the most time consuming part in the entire design. For the reason that analog library cells are usually not suitable for all required analog functions and the automatic design tools for synthesis of analog circuits is much less and simpler compared with the digital circuits (in digital field there are various complex FPGA，CPLD for the use of automatic design). The analog circuits have to be designed by hand or preferable with the help of analog simulation tools. It needed many experienced experts design the circuit manually, and optimize the parameters repeatedly to increase the performance.

There are two steps in the design of analog circuit. The first one is to construct the topology of the circuit, while in the op-amp design, many classical structures have been designed and these structures are fixed wonderfully. The second one is to adjust the circuit parameters. This is a time-consumed iteration, and also in different technology conditions the scaling transistors bring out different results [1] [2].

Previously there are some synthesis tools that have been designed. The first one is IDAC [3].IDAC sizes several topologies and the user selected the sized circuit with the best performance. The main drawback of this approach is the fact that a lot of topologies have to be sized completely, of which only one will be used. To avoid this computational overhead, other tools, like OASYS [4] and OPASYN [5], are selected on forehand one topology, based on heuristics. If the tool can not size the selected topology correctly, other heuristics are used to redo the topology selection. Unfortunately, these heuristics are very difficult to create and there is a risk that a non-optimal topology is selected. In SEAS [6],the knowledge intensive topology selection heuristics are avoided by using an evolution algorithm to modify the topology. However, the several intermediate topologies still have to be sized completely in SEAS, using a time consuming simulated annealing algorithm. The first methodology that handles topology selection and circuit sizing simultaneously was presented by P.Maulik [7].In this approach, topology selection is embedded in the circuit sizing problem. Therefore the risk of selecting a non-optimal topology is reduced, without the need of sizing many topologies. However, a lot of expert design knowledge is required to generate the essential design equations. On the basis of P.Maulik methodology named DARWIN was proposed [8], in this method topologies are built up

from basic building blocks. The topology selection and circuit sizing are performer by means of simple genetic algorithm. For the reason only a little amount of expert design knowledge is used in the program and it is not based on simulation tools and model library, so it is useful in determining the structure of the circuit while in the sizing process it is not nicety.

From then on Genetic Algorithm were employed in many CAD applications and analog circuits design. For GA is a global search algorithm, which has been proved to be the very useful method in the EHW (Evolvable Hardware) and the design of analog or digital circuits [9][10][11]. This search technique can be successfully applied to a class of optimization problems in which the searching space is too large to convergence by conventional techniques, and also it can save much time in finding the target [12].

Much research work has been devoted to the field of sizing transistors and synthesis of the analog circuit by GA. Ricardo S. Zebulum who digs into the search of EHW based on GA for many years, puts forward a way that introduces the GA into the synthesis of the CMOS op-amp [13].While it is not precise nowadays because it is separated from the advanced simulation tool, model library and the manufacture technology. And also the simple GA has some intrinsic shortcomings so it should be adjusted properly in order to get the accurate results. Wim Kruiskamp gave us a tool that is able to synthesize CMOS op-amp. This tool brings forward a good technique in finding the best topology from many available model combination candidates by GA but it is so rough in determining the sizes of the transistor. Some equation based method was put forward [14]. The author used the GA into the evolving of the analytic equations to find the optimized parameters in the analog CMOS circuits design. Although this method saves much time in finding the result, the shortcoming is obvious: for the result is based on the deducing and evolving of the equation, the achieved result is not as accurate as simulation based method and a little far away from the real simulation goal.

With the time goes on some other algorithm or method are used to optimize the analog circuits. For example the Optimization algorithm (Levenburg Marquardt algorithm) is embedded in Hspice to optimize parameters, and the searching direction is a combination of the Steepest Descent method and the Gauss-Newton method [15]. While, these algorithms have their inherent drawback for it is easy to converge to the local best solution. Every time you input a set of different initial values you will not get the same result. In the literature [16], author used simplicity algorithm to optimize the parameters of the op-amp, for this algorithm is based on derivative so it is also easy to convergence to the local best optimized solution and the result is greatly relative to the initial value. In the end of the paper the comparing of the result of the method based on adaptive GA and the method in above paper is proposed.

In this paper, one transistor sizing method is proposed with the fixed classical 2-stage op-amp structure, which used the adaptive GA as the searching tool and Hspice as the fitness producer and evaluator. This method is more accurate and useful in actual analog circuit design comparing with the previous approaches not only because the GA is self-adaptive with the evolution process and its direction is controlled by the manual analog circuit design experience, but also its evaluation is based on the simulation results and model technology library. However, in a few cases it may consume a little more time than others.

The organization of the rest of the paper is as follows:

Section 2 describes the genetic algorithm.

Section 3 introduces the circuit of this experiment.

Section 4describes the modification of the genetic algorithm in order to overcome the inherent drawback of the GA simultaneously satisfy the all kinds of specification.

Section 5 introduce the simulation results and gives the conclusion.

Section 6 concludes the work.

2 Introduction of GA

Genetic Algorithm is a kind of newly developed global optimization algorithm in recent years, first proposed by professor Holland [17].It introduces the idea of biology genetics, enhancing the adaptability of each individual by the effective operation mechanism such as selection, crossover and mutation and so on. It simulates the evolution process of natural selection. Genetic Algorithm has attracted a large group researchers who extends it to many field such as optimization, circuits design, automatic control, machine learning and so on.

The evolution process of Simple GA is shown in Fig.1.

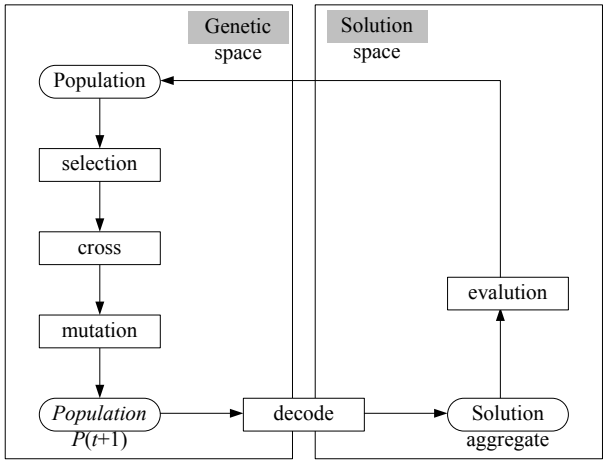

Fig.1 Evolution process of GA

As for the GA shown in the picture, the population is produced randomly at the beginning, and the by the genetic operation such as choosing, crossing and mutating the new population with the good characters is generated. From estimating the fitness of every individual in the new population is evaluated. Also the better one is chosen for the next evolution. Repeating above work the until the goal is achieved or the generation is limited.

Basic genetic algorithms include three main operators: selection, crossover, and mutation as described in detail below.

2.1 Selection

Selection is one of the most fundamental genetic operators. Selection operation may be modeled as follows:

$$P_{select}(n) = f(n) / \sum_{k=1}^{pop} f(k) \quad (1)$$

Where n is the n^{th} individual, pop is the population size and $f(n)$ is the fitness function. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly. Some of the most commonly used selection operators are: roulette wheel selection, tournament selection, ranking selection etc.

2.2 Crossover

This is the most powerful genetic operator, and may be considered as the main engine for exploration in a GA. This operator is responsible for the cutting and recombination of building blocks.

The simplest form of crossover is that, a single point is chosen on two equal length chromosomes and that are crossed at that particular point. It is possible to select two or more points for crossover, to get more genetic mixing but sometimes while using multipoint crossover it degrades the performance. Crossover can be shown in Fig.2.

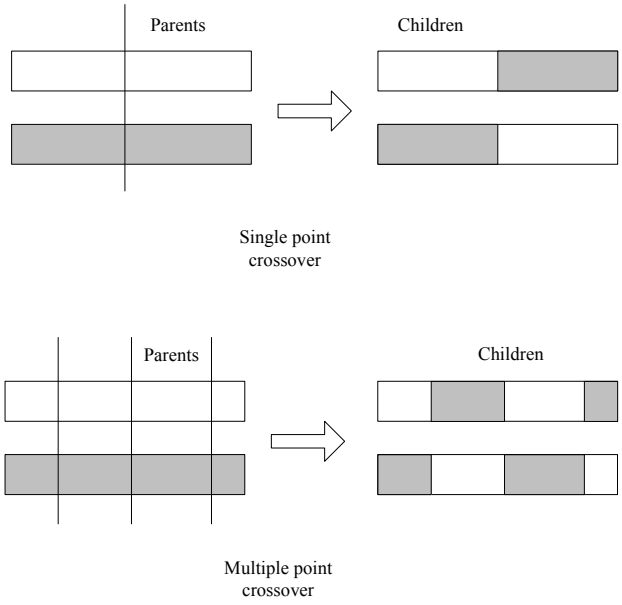

Fig. 2 Crossover

Crossover generally consists of forming a new solution by taking some parameters from one solution and exchanging it with another at the very same point. Thus we get new offspring. Some crossover operators use complex geometric methods to generate the off springs of two parents.

2.3 Mutation

This is a common genetic manipulation operator, and it involves the random alteration of genes during the process of copying a chromosome from one generation to the next. Mutation simply involves the incorrect copying of some parameters, which make up a solution. It may be illustrated in Fig.3.

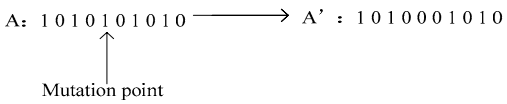

Fig.3 Mutation

Mutation is usually used to avoid premature convergence, which is a common problem in GAs, where fixed length binary coding is used. When proportional selection is used, all the individual chromosomes in the population become very similar before a nearly optimal solution is reached, thus preventing any further progress. In such cases mutation is essential. Mutation acts against the loss of efficiency due to the damage of good genetic material. Thus there is a payoff between exploitation and exploration illustrated here.

3 Design of AGA for the experiment

The basic goal of the design is to size the length and width of the CMOS transistors (op-amp) and also the values of the passive devices (resistance and capacitance) are adjusted simultaneously in the condition that the topology of the circuit is fixed.

The optimizing program is written by c language. At the beginning we generate the individual randomly *n* times (*n* represents the population size). The individual is made up of binary code string encoding a particular sized op-amp. Every time the individual is got the program will change the binary code string into the netlist file (*.sp). After running the Hspice the output file will be produced (*.lis). Through analyzing and calculating the results the fitness of every individual can be got. And then the GA can be used to choose the better individuals as the parents of next generation. After crossing and mutating, the new generation is produced. Performing the above works iteratively the goal will be achieved in the end. The operating flow diagram of the program is shown in Fig.4.

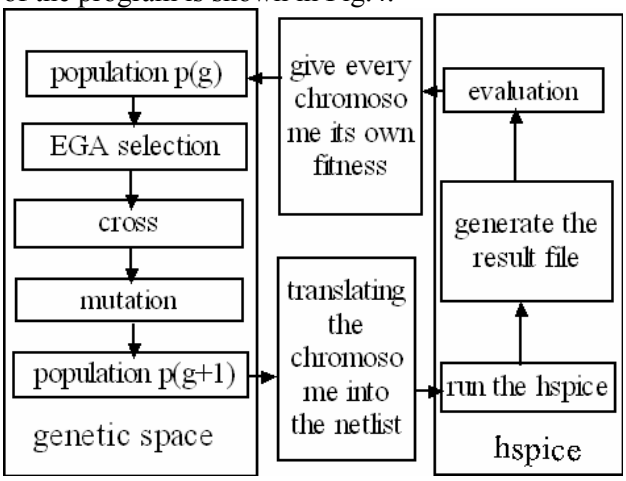

Fig.4 The operating flow diagram of the program

In order to prove that this method is suitable for multi-objective optimization the typical 2 stage op-amp (shown in Fig. 5) is chosen. For the reason that although it has only a few transistors it owns enough characteristics to be tested and compared, so many scholars select this structure as the classical structure to perform the simulation previously, and its structure is mature for many years.

3.1 Representation

In the program, every individual is presented by a binary code string. From fig. 5 we can see that there are 8 transistors and a miler compensating resistor and a miler capacitor to be adjusted. As a total there are 18 parameters to be adjusted and the each gene of the chromosome stands for one parameters.

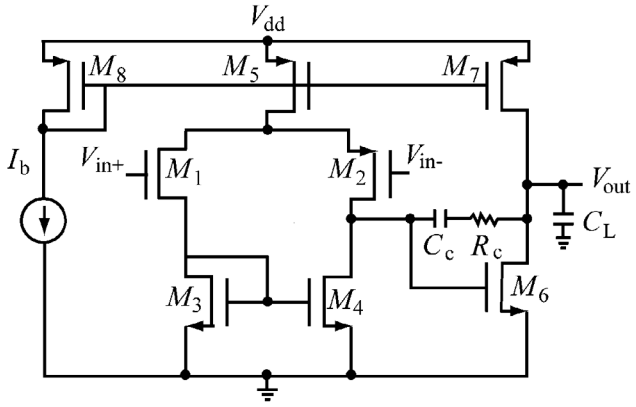

Fig.5 The schematic of two-stage op-amp

In order to make the optimization results reasonable and the process easy to control, the actual analog circuit design experience should be combined with the GA. So some constrains are added in the initial boundary conditions:

(1) For the reason that the dimension of transistor M2 is equal to that of transistor M1 and the dimension of transistor M3 is equal to that of transistor M4. So the number of the optimized parameters is reduced to 14.

(2) And that W_7 (width of M7) is determined by the systematic offset cancellation relation in equation (2):

$$W_7 = 0.5 \times \frac{L_7}{L_5} \times \frac{W_6}{W_3} \times \frac{L_3}{L_6} \times W_5 \qquad (2)$$

Thus the parameter vector is compressed to:

[W1,L1,W3,L3,W5,L5,W6,L6,L7,W8,L8, R,C]

(3) In order to simplify the calculation we define that every 8 bits represent one value. In this experiment there are 13 values to be adjusted so the code string is 104-bit binary code.

(4) For in the traditional analog design the length of the transistor is about 3-4 times of the minimum technical length, and we use TSMC 0.25um process so we define every length of the transistor is from0.25 to 3um. The step length is 0.01um.

(5) Avoid some transistors working in the linear region some widths of the transistor have to be restricted. M5 is the main route of the current so the width of M5 is the largest in the circuit: 30 times of the length at least, and we define the 1200um is the largest value of W5. M8 control the current of the current source so its width is not serious to affect the circuit so its shortest length is

about 10 times of the length, and 200um is its largest value. As the first stage of the op-amp, and M1, M2 are the input transistors so the widths of them are larger than that of M3, M4 and smaller than that of M5, so their widths are about 30 times of the length at least, and their width is limited to 1000um. The widths of M3, M4 is only larger than the length and their searching scope is from 10um to 100um. M6 is the common transistor in the circuits and it contributes less sway to the results so we make its width about 5um at least and 100um is the largest value.

(6) The miler compensating capacitor is at least 0.22times of the load capacitor, and we define the load capacitor 5pf. So the boundary of the Cc can be limited.

The searching space of the adjusted transistor values is shown in Table 1.

Table 1 Searching space of the parameters

	W1(μ)	L1(μ)	W3(μ)	L3(μ)	W5(μ)	L5(μ)
Max	1000	3	100	3	1200	3
Min	30L1	0.25	10	0.25	30L5	0.25
	W6(μ)	L6(μ)		L7(μ)	W8(μ)	L8(μ)
Max	100	2		3	200	3
Min	5	0.25		0.25	10L8	0.25
	R(Ω)	C(pF)				
Max	1000	6				
Min	100	1.1				

3.2 The fitness function

The design of the op-amp should consider many specifications, so it is a typical multi-objective optimization. GA is superior to conventional optimization algorithms in multi-objective problems because of the following four features:

1. Gas search with a population of points (candidate solutions), not a single point. Thus, they are less likely to be trapped in a local optimum.

2. GAs use only the values of the payoff (objective function) information, and not the derivatives or other auxiliary knowledge.

3. GAs work with a coding (representation) of a parameter set not the parameters themselves. Thus the search method is naturally applicable for solving secrete and integer programming problems.

4. GAs use randomized parents selection and crossover from the old generation. Thus they efficiently explore the new combinations with the available knowledge to find a new generation with better fitness values.

In order to reduce the calculation, we can mix every sub-objective into one general function, so the problem can be changed into one-objective optimization. The overall fitness can be achieved from equation (3):

$$Fitness(x) = \sum_{i=1}^{n} w_i.Fit_i(x) \quad (3)$$

In the above equation (2), w_i is the weight coefficient of every sub-objective. $Fit_i(x)$ is the fitness of every performance considered. $Fitness(x)$ is the overall fitness. And n is the number of the performance considered. In this simulation seven performances are considered. They are the DC gain, bandwidth of unity-gain, phase margin, 3-db bandwidth, thermal noise, power and slew rate.

The main problem of this method is the setting of the weights associated to each objective. In order to overcome this problem, GA can be combined with the specifications, and the adaptive weights along the optimisation process can be used in the sense that their values will be up-dated according to the average fitness value of every objective changing.

$$F_i(x) = \frac{f_i(x)}{\overline{f_i(x)}} \quad (4)$$

$$Fit_i(x) = \left[1 - \exp\left(-F_i(x)\right)\right] \quad (5)$$

The normalized equations are given in equation (4) and equation (5), where $\overline{f_i(x)}$ is the average fitness respect to $f_i(x)$ in one generation. And the $f_i(x)$ represents for the performance got from simulation. The purpose of equation (5) is to limit the value of $Fit_i(x)$ in the scope (0, 1). The normalisation is to account for the fact that the objectives are measured in different units and all of them must have the same influence in the overall fitness.

$$w_i = 100\frac{spec_i}{\overline{f_i(x)}} \qquad if\ \overline{f_i(x)} < accep_i \quad (6)$$

$$w_i = 10\frac{spec_i}{\overline{f_i(x)}} \qquad if\ accdp_i < \overline{f_i(x)} < spec_i \quad (7)$$

$$w_i = \frac{spec_i}{\overline{f_i(x)}} \qquad if\ spec_i < \overline{f_i(x)} \quad (8)$$

If the objective is to be maximised, the weight vector expression is defined from equations (6) ~ (8), where $spec_i$ stands for the user specification, $accep_i$ represents the performance user can tolerance. If a particular objective is to be maximised, its weight is defined as the ratio between the desired specification $spec_i$, and the current generation average fitness value $\overline{f_i(x)}$, for a particular objective *i*. This ratio is multiplied by 100 if $\overline{f_i(x)}$ is lower than the minimum tolerance value $accep_i$. The ration is multiplied by 10 if $\overline{f_i(x)}$ is between the $accep_i$ value and $spec_i$ value. The ratio is set to 1, if the $\overline{f_i(x)}$ has already over the $spec_i$ value.

If the objective is to be minimised, its weight takes a negative value equation (6) ~ (8) can be changed into equation (9) ~ (11):

$$w_i = -100\frac{spec_i}{\overline{f_i(x)}} \qquad if\ \overline{f_i(x)} > accep_i \quad (9)$$

$$w_i = -10\frac{spec_i}{\overline{f_i(x)}} \qquad if\ accdp_i > \overline{f_i(x)} > spec_i \quad (10)$$

$$w_i = -\frac{spec_i}{\overline{f_i(x)}} \qquad if\ spec_i > \overline{f_i(x)} \quad (11)$$

The amplification factor enhances the influence of the weight w_i as long as the objective is not satisfied. When the objective is satisfied, the weight is set to the unit.

Summarising this method, the idea is to assign large weights to objectives for which the average fitness is far from the target value, and low weights to objectives whose average values are around the desired ones. The search will then be driven by unsatisfied design requirements.

3.3 Genetic operator

In this algorithm, we choose the GA as the basic way to evolve the circuit. At the beginning, the roulette wheel selection is the selection way in the program. Since there are 104 bits in one gene, we favor the multiple-point crossover and the cross point is random in order to ensure that the crossover can make all kinds of individual as shown in Fig. 6.

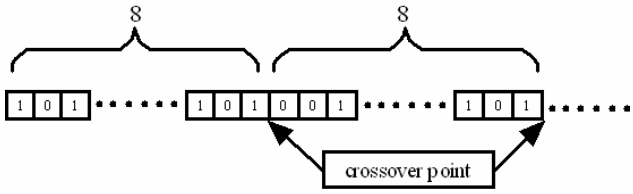

Fig.6 Picture showing 2-point crossover

In the whole process of implementing Genetic Algorithm, the choosing of cross probability P_c and mutation probability P_m is the bottleneck of the operation and performance of Genetic Algorithm, and it directly affects the convergence of the algorithm. The larger P_c is, the faster the new individuals come into being. If P_c is too large, the genetic mode will probably be destroyed soon. However, if P_c is too small, the search will slower or even static. As far as mutation probability P_m, if it is too small, it is hard to produce the new individual structure, if it is too large, the Genetic Algorithm will become pure random search algorithm. However, at present, there is no universal method to definite P_c and P_m at one time. It is too fussy to definite different P_c and P_m for different optimization problems by iterative experiments. Therefore, an adaptive Genetic Algorithm proposed in this paper, with the P_c and P_m modified along with the evolution process.

$$P_c(t) = \min\{P_{c0}.e^{-a.t/t_{\max}}/f_d(t), P_{c0}\} \qquad 0 < t < t_{\max} \quad (12)$$

$$f_d(t) = \overline{fit(t)}/\left[fit_{\max}(t) - fit_{\min}(t)\right] \quad (13)$$

$$P_m(t) = P_{m0}.e^{(k-t)}\{t<100, k=t; t>100, k=100\} \quad (14)$$

In the equation (12) ~ (14), *a*, *k* are constant respectively. $f_d(t)$ is used to test the variety of the population. $fit_{\max}(t)$ and $fit_{\min}(t)$ are the maximum and minimum the fitness in one generation. It can be seen that the smaller is $f_d(t)$ the more various is the population, or the reverse. And the beginning of the evolution the population has great variety, and the P_c are larger comparatively. With evolution goes on the population turns into convergence and the P_c become smaller and smaller in order to prevent the better individual from destroying.

In order to avoid convergence problem the probability of mutation decreases remarkably with the number of generation increasing as described in equation(14) where P_{m0} represents the initial mutation probability, and the t indicates the number of generation, the coefficient k is variable to control the mutation. From many trials we find that P_{c0}=0.8 is suitable for the optimization and P_{m0} can be chosen 0.1. And as for the k, in the experiment when t<100, k=t, when t>100, k=100.

4 Simulation and Discussion

The example of this simulation is the typical 2-stage op-amp. The library of the device model used for simulation is TSMC 0.25μm technology library operating at 2.5V power supply. The computer used is the ultra-10 workstation with the cpu400, 256M EMS memory. For adaptive Genetic Algorithm the population size is 100 (randomly produced). The maximal generation is 200. The initial probability of the crossover P_{c0}=0.8, the initial mutation probability is P_{m0}=0.1. Constant a=2. In this experiment seven performances are considered, they are DC-gain, bandwidth of unity-gain, phase-margin, 3-db bandwidth, power, noise, slew-rate. Program language is the c language, simulation tool is Hspice 2003. To complete one simulation about 10 hours are needed which is mainly consumed in the Hspice simulation.

The simulation results are given in Table 2, Table 3, Table 4, which give a comparison of two different outcomes in the different performance specification, in contrast to the simulation results and specification of literature [13], where example1 uses the same specification of the literature. The goal of example2 is to optimize one high gain and low power operation amplifier so it has to sacrifices other performances.

Fig.7 and Fig.8 give the process of average fitness of DC-gain and the excellent individual of example2. It is can be seen from the picture that after a period of vibration the average fitness continue going up. It account for the reason that waiting for other performance to satisfy the *accep* the optimization strength can make effect on DC-gain. It is the same reason for excellent individual.

Fig.9 to Fig.15 describes the evolution process of DC-gain, bandwidth of unity-gain, phase-margin, 3db-bandwidth, power, noise and slew-rate respectively of example1. It can be seen from the pictures that the evolving effect is obvious for DC-gain, bandwidth of unity-gain, noise and slew-rate. While the noise is turned up after the evolution for the reason its optimization is opposite to other performances and its effective ability is weak in the whole fitness function, and after other performance getting to the goal its fitness going down. As for slew-rate, it is derived from the equation (15):

$$I_5 = SR \bullet C_c \tag{15}$$

So in this experiment the I_5 is to be optimized.

Fig. 16 is the p-f (phase-frequency), a-f (amplitude-frequency) characteristics and noise characteristics of example2 respectively.

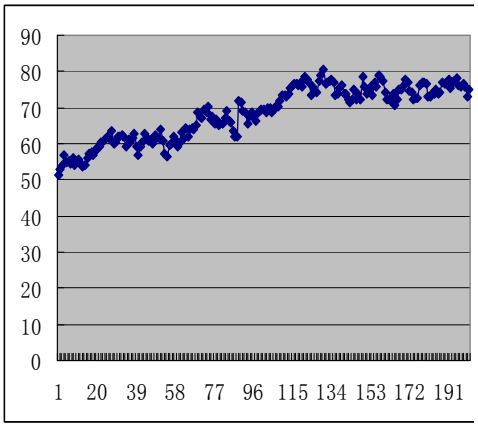

Fig.7 The evolution process of average fitness of DC-gain in example1

Table 2 the simulation specifications of literature, example1 and example2

Performance	Design goal	Literature	Example1	Example2
DC-gain	*accep*		60db	60db
	spec	>80db	80db	100db
Bandwidth of unity-gain	*accep*		30MHz	0.1MHz
	spec	>40MHz	40MHz	5MHz
Phase-margin	*accep*		50	45
	spec	>60	60	50
3-db bandwidth	*accep*		1k	0.01kHz
	spec		10k	0.1kHz
Power	*accep*		5mW	2mW
	spec	<2mW	2mW	1mW
Noise	*accep*		100nv/$\sqrt{Hz}$	100nv/$\sqrt{Hz}$
	spec	<300nV/$\sqrt{Hz}$	10nv/$\sqrt{Hz}$	10nv/$\sqrt{Hz}$
Slew-rate	*accep*		10V/μs	1V/μs
	spec	>30V/μs	30V/μs	5V/μs

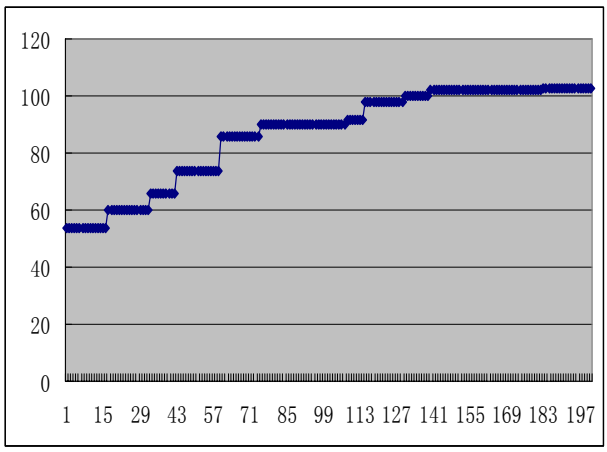

Fig.8 The evolution process of best fitness of DC-gain in example1

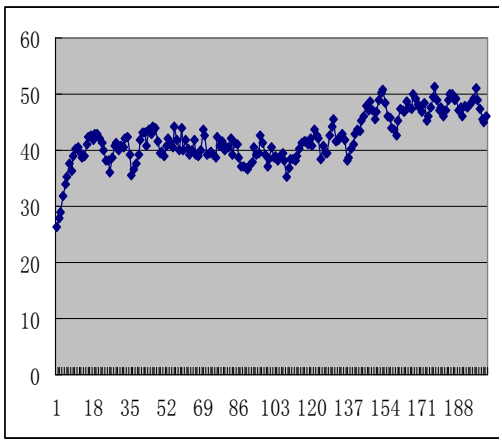

Fig.9 The evolution process of DC-gain in example 2

Table 3 the optimized performance of literature, example1 and example2

Performance	Literature	Example1	Example2
DC-gain	83.1db	88.8db	103db
Bandwidth of unity-gain	43.4MHz	89MHz	10.6MHz
Phase-margin	60.6	65.5	57.1
3-db bandwidth		4.6kHz	0.11kHz
Power	1.076mW	1.06mW	0.993mW
Noise	6.5nV/$\sqrt{Hz}$	9.18nV/$\sqrt{Hz}$	15.2nV/$\sqrt{Hz}$
Slew-rate	37.5V/μs	33.2V/μs	5.9V/μs

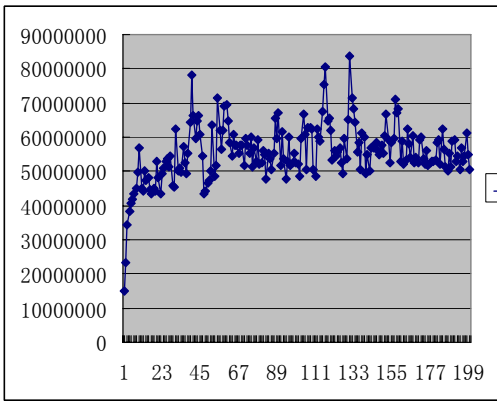

Fig.10 The evolution process of bandwidth of unity gain in example 2

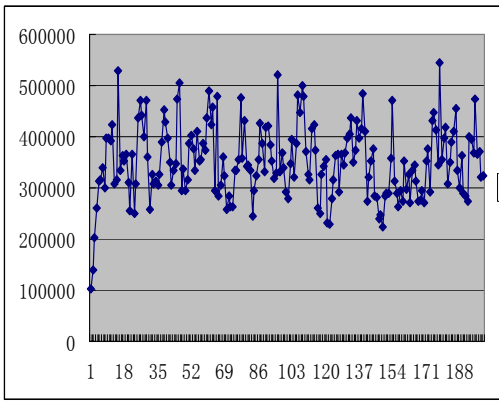

Fig.12 The evolution process of 3-db bandwidth in example 2

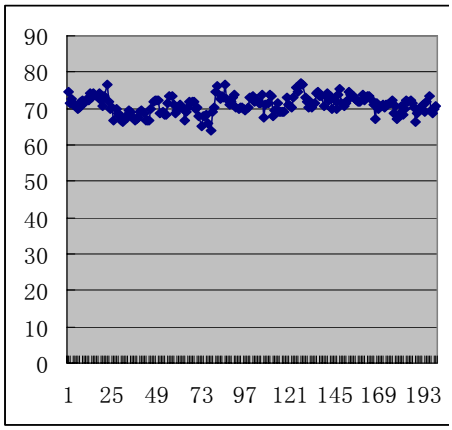

Fig.11 The evolution process of phase-margin in example 2

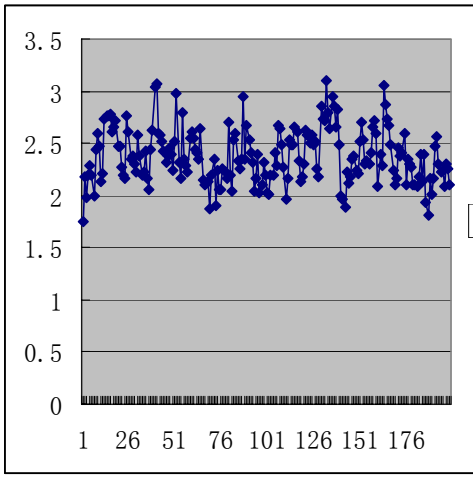

Fig.13 The evolution process of power in example 2

Table 4 the optimized parameters of literature, example1 and example2

Literature (μm)		Example1 (μm)		Example2 (μm)	
L1=1.43	W1=25.9,m=8	L1=1.83	W1=20.1,m=8	L1=4.2	W1=21.3,m=8
L3=0.95	W3=13.59,m=3	L3=1.36	W3=70.2,m=8	L3=3.98	W3=12.1,m=1
L5=1.8	W5=25.8,m=18	L5=2.20	W5=21.1,m=6	L5=0.95	W5=4.9,m=1
L6=0.95	W6=15.59,m=8	L6=0.53	W6=34.2,m=6	L6=3.98	W6=23.3,m=12
L7=1.8	W7=25.8,m=24	L7=1.85	W7=11.9,m=20	L7=0.95	W7=55.2,m=1
L8=1.8	W8=25.8,m=3	L8=1.33	W8=9.0,m=1	L8=1.18	W8=5.1,m=1
R	C=3.68pf	R=635 Ω	2.48nf	R=273 Ω	C=4.96pf

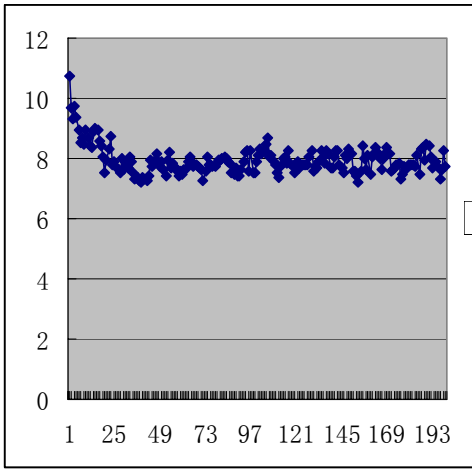

Fig.14 The evolution process of noise in example 2

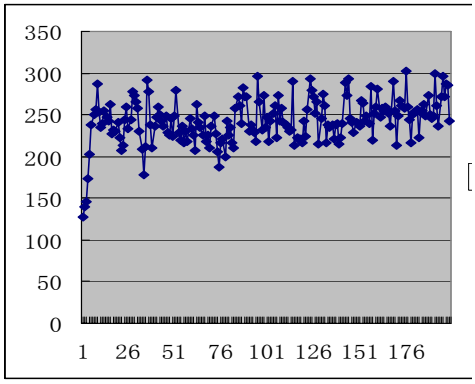

Fig.15 The evolution process of slew-rate in example 2

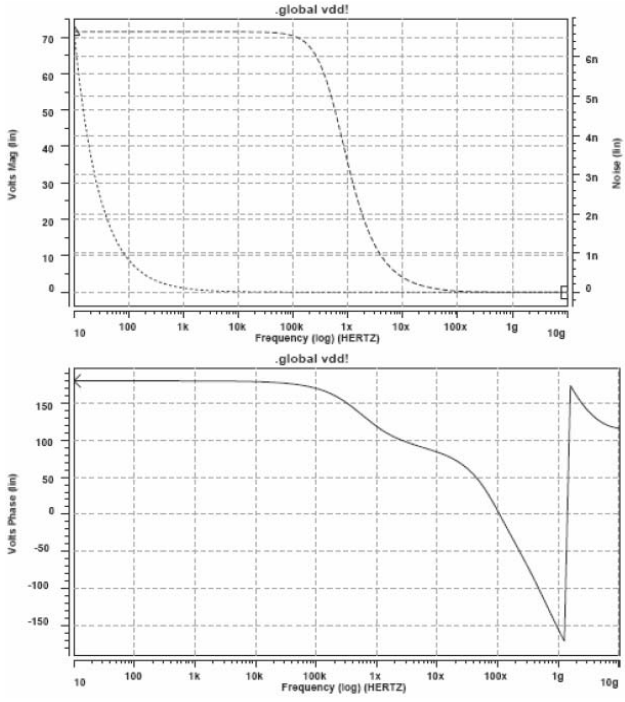

Fig.16 Diagram of the n-f and a-f characteristic of the goal circuit, and the p-f characteristic is below.

4 Conclusion

In this paper we present an accurate method based on self-adaptive GA and manual experience that is useful in adjusting parameters of the transistors in an analog circuit if the topology of the circuit is fixed.

The work of this method is similar to the work by human manual adjusting, and the result of the

experiment can be turned into layout directly if the characteristics satisfy our need. This method can be developed to an EDA software tool in the future.

References:

[1] Paulino N, Goes J. *Desing methodology for optimization of analog building blocks using genetic algorithm,* IEEE ISCAS[C].May 2001,5:6-9.

[2] Deval Y, Begueret J B, Tomas J. *Toward analog circuit synthesis: A global methodology based upon design of experiments*, IEEE Integrated Circuits and Systems Design, Proceedings of 13th symposium[c]. Sept 2000, pp.295-300

[3] M.Degrauwe, et al, *IDAC: An interactive design tool for analog integrated circuits,* IEEE J. Solid-State Circuits, 1987, SC-22,1106-1116.

[4] R.Harjani, R. A. Rutenbar, and L.R.Carley, *OASYS: A framework for analog circuit synthesis*, IEEE Trans on Computer Aided Design, 1989, CAD-8, 1247-1266.

[5] H. Y. Koh, C. H. Sequin, and P. R. Gray, *OPASYN: A compiler for MOS operational amplifiers,* IEEE Trans on Computer Aided Design, 1989, CAD-9, 113-125.

[6] Z. Ning, M. Kole, T. Mouthaan, and H. Wallinga, *Analog circuit design automation for performance,* Proc. 14th IEEE CICC, Boston, MA, 1992, IEEE, New York, pp.821-824.

[7] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, *A mixed-integer nonlinear programming approach to analog circuit synthesis,* Proc.29th ACM/IEEE DAC, Anaheim, CA, 1992,IEEE Los Alamitos, pp698-703.

[8] Wim Kruiskamp and Domine Leenaerts, *DARVIN: CMOS op-amp synthesis by means of a genetic algorithm,* 32nd Annual Design Automation conference, 1995, pp. 433-438.

[9] Drissa Traore, Zhi-Gang Mao, *Self-reconfigurable FSM with exterior transition concept and analysis,* WSEAS Transactions on circuits and systems, Vol.5, 2006, pp.640-646.

[10] Abdullah A. Hussain, Zhi-Gang Mao, *A compression technique for instruction density optimization*, WSEAS Transactions on circuits and systems, Vol.5, 2006, pp.763-768.

[11] Ching-Mai Ko, Yu-Jung Huang, Shen-Li Fu, Mei-Hui Guo, *MCM placement based on multi-objective optimization approach,* WSEAS Transactions on circuits and systems, Vol.5, 2006, pp.753-759.

[12] D. E. Golderberg, *Genetic Algorithms in search optimization and machine learning*, Addison Wesley, Reading MA , 1989.

[13] Ricardo S. Zebulum, Marco A. Pacheco and Marley Vellasco, *Synthesis of CMOS Operational Amplifiers Through Genetic Algorithms*, XI Brazilian Symposium on Integrated Circuit, 9, 1998, pp.125-128.

[14] Wei-Shan Zheng, *A new design method in CMOS analog cell circuit optimization*, Journal of Applied Sciences, 3, 2006 pp. 150-153.

[15] HSPICE Users Manual, Meta software, 2003.

[16] De-Zhong Za, Ting Yi, Jie Fang, Zhi-Liang Hong, *Design and synthesis of a fully differential high speed CMOS operational transconductance amplifier* Journal of computer-aided design & computer graphics, 12,2004, pp. 1640-1644.

[17] Holland J H. Adaptation in Natural and Artificial Systems, MIT Press, 1975.