Ročník 2013 electroscope Číslo V

A Novel Tool for Automated Design of Analog Integrated Circuits

M. Kubař [1], J. Jakovenko [1]
[1] Department of Microelectronics, Czech Technical University,
Technická 2, 166 27 Prague
E-mail : kubarmil@fel.cvut.cz, jakovenk@fel.cvut.cz

Abstract:
This paper presents a novel optimization tool, which was made for the design of the analog integrated circuits. The proposed tool is based on the robust version of the differential evolution optimization algorithm. Corners of technology, temperature, voltage and current supplies are taken into account during the optimization. This ensures robust resulting circuits. These circuits usually do not need any schematic change and are ready for the layout. The developed tool is implemented directly to the Cadence design environment to achieve very short setup time of the optimizations. The design automation procedure was enhanced by novel optimization watchdog feature. It was created to control optimization progress and to reduce the search space to produce better circuits in shorter time. Another novel feature for accurate design of current mirrors was created and implemented to the tool. The novel tool and features were successfully tested by optimization of two design examples.

INTRODUCTION

While the analog part of the integrated circuit covers usually 10 % of its area, its creation takes about 90% of the total design time. An optimization of the analog circuits can save enormous part of this time. At present much effort is spent on development of an optimization tool to make the design time of the analog circuit shorter [1], [2] and [3].
Our work is aimed mainly to create the optimization tool for industry everyday design work. It requires a very short setup time of the design task, accurate ready-to-use results and tool robustness and to be able to converge to the solution for a generic circuit and its specification.
The analog design automation approaches published so far in accordance with [4] and [5] are:

- Knowledge based - contains complete design plan describing how the circuit components must be sized to reach the solution of the design problem. There is no guarantee of finding the optimum solution.

Following three design automation approaches are optimization based. They use an optimization engine instead of a design plan to perform the design task:

- Equation based [6] and [7] - use analytic design equations to evaluate the circuit performance. The main drawback is, that not all design characteristics can be easily captured by analytical equations. Moreover, the approximations introduced in the analytical equations yields low accuracy design.
- Simulation based [8], [9] and [10] – use simulations to evaluate the circuit performance. This is a very flexible solution, since it accommodates to any type of circuit topology and yields superior accuracy. The drawback of these methods is a long computation time needed to evaluate performance of the circuit by simulations.
- Learning strategy based [1] and [11] - the behavior of the circuit is modeled by a learning mechanism based on the distribution of variation. They require set of training samples, which must be evaluated at the beginning of the optimization. The amount of the training data will influence the accuracy of the performance predictions made by the learning machine.

The equation based methods are not accurate enough to design robust circuits automatically. The learning based strategies can produce powerful circuit. But their setup time can be longer than a design without any optimization tool, because of training samples creation. Simulation based tools produce the most accurate circuits and their setup time is the shortest. Thus we chose this approach for the proposed tool despite of its long computation time.
Many works about the automated analog circuit design published recently are quite sophisticated and present powerful analog circuit synthesis ideas and improvements. On the other hand these works or the principles they present are not optimal solutions for the industry design optimization tool. Such a tool must satisfy the requirements of our approach mentioned above: very short setup time of the design task, accurate ready-to-use results and tool robustness to enable the convergence for an arbitrary circuit.
Algorithm presented in [1] uses learning based method for fast convergence. It uses simulations to generate training samples. Neural networks core quickly optimizes the circuits. That core is created in Matlab and do not use simulations. Thus it is not accurate enough.
Method presented in [2] is a powerful optimization method based on a hybrid approach. It combines Matlab equation based and Hspice simulation based approaches. This method achieved a very good optimization times even for complex circuits. But it

costs quite long setup time especially because of finding the penalty coefficients they use.
Powerful learning mechanism is used in [11]. It is combined with corner simulation based method. It leads to long setup time to create enough training samples that can pay off in complex design tasks. But the setup time is usually too long in typical design cases.
Work [3] uses Alienor-based method to significantly reduce the number of design variables leading to enhanced convergence. But this approach is not usable in simulation based optimization tool that needs all design variables to define parameters of the circuit devices.
Work [8] uses simulation based approach with accurate device models. It performs corner analysis only for the final solution. Thus the specification can not be met in worst case corner.
Simulation based approached in [9] is based on Particle Swarm Optimization. It is able to reach powerful solution but only for typical conditions and corners.
State-of-the-art of the analog circuit design automation is described in [4] and [5]. Moreover open research points in this field are discussed in [4]. We propose a solution of one of them. We reduce search space of the optimization task by a novel optimization watchdog feature. The space reduction causes faster optimization convergence and reduced computation time. Moreover the novel approach for automated design of automated current mirrors was created and implemented to the tool.
We present a novel optimization tool implemented to Cadence design environment GUI (graphical user interface). It makes the tool more user friendly and makes the setup time of the automated design task short. The proposed optimization tool uses the simulation based approach with real device models and full corner analysis to produce robust ready-to-use circuits.
The novel optimization approach was successfully tested on the optimization of two-stage Miller OTA and voltage regulator. The presented tool is universal and can be used to optimize an arbitrary circuit.

OPTIMIZATION ALGORITHM

Suitable optimization method is required to make a robust optimization tool. Gradient based methods that are frequently used for optimizations can be easily trapped in a local optimum. Much better performance could be achieved by evolutionary techniques. They are designed to converge to the global extreme because of their stochastic behavior [7]. These techniques are also well suited for multi-criterion optimization [10] which is the case of analog circuit optimization.
Differential evolution began to be one of the best evolution algorithms for solving the real-valued test function [12] and [13]. Recently, differential evolution algorithm became a very good choice for analog circuit sizing [7], [14] and [15] in terms of: optimization convergence time; optimization stability of non-convex, multi-modal and non-linear functions; rapid convergence speed; multi-variable real-valued functions solving and since differential evolution operations are simple and easy to program.
Differential evolution is similar to the overall structure of the genetic algorithm. The main difference is the mutation operation. This operation uses a perturbation of two members as the vector to be added to the third member, which produces a new vector. The new vector is then mixed with the predefined parameters in accordance with certain rules to produce trial vectors. This operation is called crossover. If the trial vector fitness is less than the target vector fitness, the trial vector is placed instead of the target vector to the next generation. These operations must be done for all members of the population in order to produce the same number of competitors in the next generation.
More circuit parameters are optimized usually in the analog circuit optimization tasks. Therefore the fitness function is necessary for the circuit fitness determination. We use the fitness function presented in [8] that showed good optimization convergence speed and results:

$$F = \sqrt{\sum_{i=1}^{cp} X^2}$$
$$X = \frac{CP_{SPi} - CP_{SIMi}}{CP_{SPi}} \quad for\ CP_{SPi} > 0 \qquad (1)$$
$$X = CP_{SIMi} \quad for\ CP_{SPi} = 0$$

where CP_{SP} represents the circuit parameter specification and CP_{SIM} denotes the simulation value of this circuit parameter. The sum is done for *cp* optimized circuit parameters. The circuit parameter which satisfies its specification does not contribute to this sum. Fitness function is equal to 0 if all simulated circuit parameters meet the specification.
In past few years several improvements of the differential evolution were presented [12] and [15] to improve its abilities. They enhance the algorithm usually despite of its other advantage, for example improving of convergence speed with danger of the convergence to the local extreme. This is not needed for the proposed robust optimization tool, that must produce circuit as powerful as possible even with the cost of longer computation time (there is usually enough of machine time but not of analog designer time in the industry design). That is why we have chosen the simple but robust version of the differential evolution called DE/rand/1/bin [15] for our optimization approach.

OPTIMIZATION TOOL

The optimization core together with implementation of the optimization method described above was

designed using Ocean scripting language. Designed scripts enable:

- Implementation of the optimization algorithm.
- Spectre circuit simulations required by the optimization method.
- Post-processing of the simulations output to extract circuit optimized parameters.

The optimization core interface is implemented to the Cadence GUI (Graphical User Interface) by the Skill language as a new toolbar. The optimized circuit is selected with the definition of its parameters in this toolbar. This kind of implementation makes the setup time of our optimizations very short (few tens of seconds). The flow diagram of the optimization tool is shown on Fig. 1.

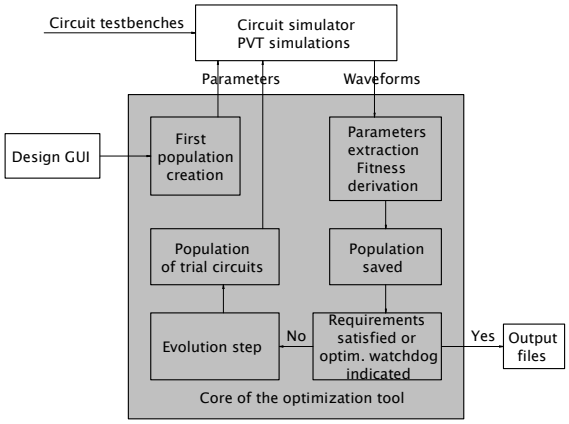

Fig. 1: Algorithm of the optimization tool

The GUI is used only to select the circuit type to be optimized, enter the circuit specification and run the optimization procedure. The Skill script was created to insert new "optimize" toolbar to the Cadence main window. The circuit which has to be optimized can be selected together with the definition of circuit parameters in that toolbar. The specification of the circuit is send to the optimization core as a text file in the format of Ocean scripting language.

The optimization core uses pre-created net-list of the optimized circuit test-benches to run Spectre simulation. The design variables (as transistors widths and lengths) are sent to circuit simulator by text file in the Spectre simulator format. Output of the optimization tool is text files containing the details of all the circuits in all created populations.

The main optimization tool core script is shown on Fig. 3. It calls several second-order scripts that are not included in this article because covers together more than 1000 lines.

First of all the initialization of the optimization tool is made (scripts *declaration.ocn* and *skill_out.ocn*). Design variables bounds, parameters of the differential evolution are set. Moreover, design specification is loaded from *skill_out.ocn* script. This script is created by the Skill script of the optimization tool interface. The last step of the initialization phase is the opening of the output files.

Then first population is created (*first_params.ocn* script). PVT Spectre simulations of the circuits in the first population are run. The worst case optimized circuit parameters are extracted from the simulations results (*first_sim.ocn* script). The details about first population are stored in the output files (*output_data.ocn* script). The fitness of the circuits in the first population is computed and checked if the specified circuit was found. The optimization process continues until the solution is found.

```
load("/2s_ota/skill_out.ocn")
load("/2s_ota/declaration.ocn")
of=outfile("/2s_ota/results/outputfile.scs"))

for(i 1 n
   load("/2s_ota/first_params.ocn"))
   load("/2s_ota/first_sim.ocn")
   load("/2s_ota/output_data.ocn")
   done = if(((F[i]=0.0)||(done==1)) 1 0))

z = z + 1

while((done<1)
   for(i 1 n
      load("/2s_ota/random.ocn")
      load("/2s_ota/param_evo.ocn"))
      load("/2s_ota/evo_sim.ocn"))

      delta = F[i] - Ftmp
      stuck = if(((delta>=wdd)||(stuck==0)) 0 1)

      load("/2s_ota/evo_new_population.ocn"))
      done = if(((F[i]=0.0)||(done==1)) 1 0)
      load("/2s_ota/output_data.ocn")))

   z = z + 1
   wdc = if((stuck==1) (wdc+1) 0)
   stuck = 1
   done = if((wdc>=wdp) 1 done))

load("/2s_ota/final_output.ocn"))
close(of)
```

Fig. 2: Ocean script – the optimization tool core

Only the worst case corner simulations are run for each circuit parameter to speed up the optimization. It was needed to run the optimization for all corners to determine the worst case for each circuit parameter. It was done during the design examples creating (only one population of one individual was enough to specify the worst case).

The procedure of the differential evolution (mutation, crossover and selection) is done in the second "for" loop. Three random numbers are computed (*random.ocn* script), design variables of the trial circuits are generated (*param_evo.ocn* script) and trial circuits are simulated. Their fitness is computed and new population is created (script *evo_new_population.ocn*). The details of the new population are stored to the output files. The "while" loop is run again until the final condition is satisfied.

First final condition is the occurrence of circuit with fitness function equal to zero – goal of the optimization. Another final requirement is the occurrence of predefined number of populations (parameter WD_P) in a row without significant progress of the optimization. It is defined by specific difference between fitness of the trial and target circuit – parameter WD_D. This indicates that the

optimization is not able to get much better circuit in reasonable time. This is the baseline of the novel feature – optimization watchdog – that helps to optimize better circuit in shorter time. The watchdog is implemented as follows:

$$WD_C = 0 \; if \; \exists i = 1,...,n \;\; F_i(t+1) \le F_i(t) - WD_D$$
$$WD_C = WD_C + 1 \;\; otherwise \qquad (2)$$

where WD_C, WD_D and WD_P are special watchdog variables, n is the number of individuals in one population. The optimization ends without satisfying the specification if WD_C is equal to WD_P.

WD_D and WD_P parameters are set to their default values by the optimization tool. On the other hand the setting can be changed by the tool user. WD_C variable is evaluated during the optimization process by the tool as shown on Fig. 2.

The first reason for optimization watchdog implementation is the natural feature of the differential evolution. The optimization process can diverge if the specification of the circuit is set beyond its limits. It can also diverge if the bounds of the design variables are set too high or too low. If several populations without significant individual improvement occur, the optimization is stopped consecutively.

The main reason for the optimization watchdog implementation is the convergence time improving by a reducing of the search space. The output file generated by *final_output.ocn* script contains information about design variables bound settings and indication how the bounds should be improved. The ranges of some design variables are narrowed down to reduce the search space to improve convergence speed of the optimization.

The optimization watchdog first generates WD_P low (from 1 to 2 in dependence on the circuit complexity – higher number for more complex circuits) and WD_D high (from 0.05 to 0.1 – lower value for more complex circuits) for short optimization which quickly scans the circuit, its specification and setting of the design variables bounds. Then the design variables bounds are improved by the tool in accordance to the output of the first short optimization. WD_P is set higher (from 3 to 5) and WD_D lower (from 0.01 to 0.05) for second optimization.

Second task performed by the watchdog is to identify the design variables bounds set inappropriately. It recommends what bound should be extended to achieve enhanced circuit performance.

The specification can be found to be beyond the limit of the circuit after first short optimization. At that point the specification can be changed for example by some tradeoff between consumption and slew-rate of the circuit.

Another novel feature for accurate current mirror design was created and implemented to the proposed tool. It is based on using of the same width and length for all current mirror transistors. Multiple transistors in parallel are used to increase or decrease bias current in the specific branch of the circuit. It is much better transistor matching approach than to size widths and lengths of the current mirror transistor independently [9].

The width and length of the current mirror transistors are not optimized (they are not design variables). They are set to be in correct mode in accordance with the bias current that is optimized. It is done by "rule of thumb" used in analog circuit design – to have gate-source voltage higher than threshold voltage increase by 100 mV. This rule must be fulfilled for all PVT corners.

We used a look-up table for setting correct dimensions of the current mirror transistors. We simulated width and length of the transistors to have correct operation point. The dimensions of the transistors are finally sized in accordance with this look-up table and to have gate area at least 2 um^2. The dimensions of the PMOS and NMOS current mirror transistors are shown on Fig. 3 for various bias currents. Values for current higher than 20 µA and lower than 0.1 µA are extrapolated.

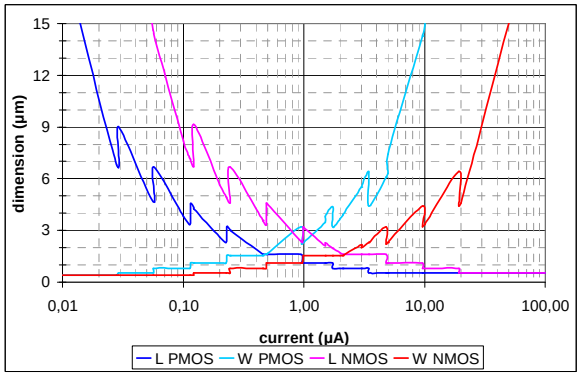

Fig. 3: Dimensions of the current mirror transistors

CIRCUIT OPTIMIZATION

The novel optimization tool was tested on optimization of two-stage Miller OTA and voltage regulator. Circuit schematic of the first design example is shown on Fig. 4

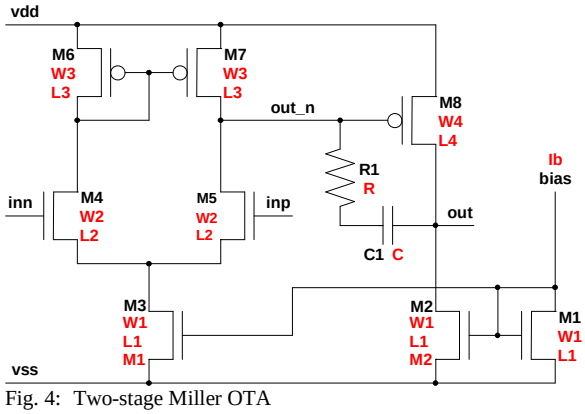

Fig. 4: Two-stage Miller OTA

Design variables are highlighted in red. Widths and lengths of transistors *M1*, *M2* and *M3* are not optimized but derived from bias current (to have appropriate area for good matching and appropriate operation point for good mirroring ratio).

AMIS 0.35 µm CMOS technology device models were used. The optimization were run for temperature range -10 °C to 50 °C, supply voltage range 1.8 V to 2.0 V and bias current variations of 30%. Population size *n* was chosen to 15. Scaling factor SF was set to 0.8 and crossover constant CR to 0.7. Load of the OTA was 10 pF.

Optimization watchdog is verified by this design example. Watchdog parameters were set 1 for WD_P and 0.1 for WD_D for the first optimization. The bounds of the design variables were updated after this optimization. The parameters WD_P and WD_D were set to 3 and 0.05 respectively for the second optimization with the reduced search space. Parameter WD_P was set to 5 and parameter WD_D to 0.01 for the simulation without using of the watchdog.

Table 1. contains details about the circuit specification, optimization results with and without watchdog using. The optimization time of the first/second watchdog optimization was 120/276 minutes respectively. Total time was 396 minutes using watchdog and 960 minutes without the watchdog. Thus the time was reduced more than two times.

Table 1: Circuit parameters - two-stage Miller OTA

Param.	*Spec.*	*Results WD / ~~WD~~*
Gain	90 dB	90 / 94 dB
PM	60 °	64 / 60 °
GB	2.0 MHz	2.8 / 2.4 MHz
SR	2.0 V/µs	2.3 / 2.2 V/µs
Cons.	20 µA	16 / 13 µA

Design variables with their bounds before/after the search space reduction and resulting values are in Table 2.

Table 2: Design variables – two-stage Miller OTA

Vaiable	*Bounds*	*Results WD / ~~WD~~*
W1 (µm)	0.4 – 50	2.6 / 1.6
L1 (µm)	0.55 – 50	1.6 / 2.7
W2 (µm)	0.5/10 – 50	37.2 / 50.0
L2 (µm)	0.5 – 20	1.7 / 505
W3 (µm)	0.5/8 – 50	41.0 / 18.1
L3 (µm)	0.5/13 – 50	49.4 / 50.0
W4 (µm)	0.5/9 – 50	38.4 / 46.5
L4 (µm)	0.5 – 10/7	0.7 / 0.5
R (kΩ)	0.1 – 10	0.1 / 0.1
C (pF)	0.1 – 10/5	0.58 / 0.38
M1 (-)	1 – 10	1 / 2
M2 (-)	1 - 10	2 / 6
Ib (µA)	0.1 – 30/9	4.0 / 1.2

Circuit schematic of the second design example is shown on Fig. 5 with design variables highlighted in red. Width and length of current mirror transistors *M1* and *M2* (parameters *W1* and *L1*) are derived from the bias current by the novel algorithm. Resistance of devices *R1* and *R2* (parameters *R1* and *R2*) are set by reference voltage, typical output voltage and current consumption of the resistor divider (set to 20 µA). The number of design variables is 8.

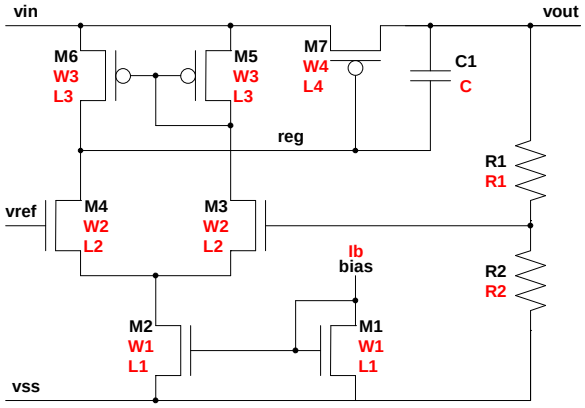

Fig. 5: Voltage regulator

Load current range of the regulator is between 20 µA and 2 mA. Reference voltage range is between 1.23 V and 1.24 V. Capacitive load of the regulator is 500 pF with 5Ω series resistor. WD_P was set to 3 and WD_D to 0.05 in this case.

Circuit specification and optimization results of the circuit parameters of the optimized voltage regulator are shown in Table 3.

Table 3: Circuit parameters – voltage regulator

Param.	*Spec.*	*Result*
Vreg-typ	1.5 V	1.5 V
Vreg-var	50 mV	7 mV
Line reg.	30 mV	0.4 mV
Load reg.	30 mV	1.2 mV
PM	60 °	92 °
BW	150 kHz	171 kHz
Temp. drift	0.5 mV/°C	0.003 mV/°C
PSRR (50Hz.	40 dB	51 dB

Table 4: Design variables – voltage regulator

Vaiable	*Bounds*	*Result*
W1 (µm)	0.4 – 50	0.4
L1 (µm)	0.55 – 50	8.2
W2 (µm)	1 – 20	12.2
L2 (µm)	1 – 20	10.2
W3 (µm)	1 – 20	18.7
L3 (µm)	1 – 20	10.2
W4 (µm)	250 – 500	434.8
L4 (µm)	0.5 – 1	0.5
C (pF)	0.1 – 10	0.85
R1 (Ω)	N/A	13.25
R2 (kΩ)	N/A	61.75
Ib (µA)	0.1 – 10	0.1

Design variables with their bounds and optimization results values are in Table 4. Lower and upper bounds

of variables *R1* and *R2* are not needed since values of those variables depend just on the circuit specification (input voltage, output voltage and current consumption of the resistor divider).
Optimization time was 48 minutes (solution found in 5th population). Optimization had very good progress since the number of optimization parameters is low even despite of high number of optimized circuit parameters.

CONCLUSIONS

A novel tool for analog circuit optimization was presented in this paper. The tool is implemented to the Cadence design environment to have very short setup time and to be easily used. Simple differential evolution method is used as the optimization algorithm to be robust and thus to be able to converge to the solution for every design example.
Novel optimization watchdog feature was implemented. It automatically changes bounds of the search space. The purpose is to reduce the search space and shorten optimization time.
Another novel feature to automatically size current mirrors was created and implemented to the tool. The resulting mirrors are accurate and robust.
Corner simulations are used during the circuit optimization. It causes higher accuracy of the results thus optimized circuit are usually ready for use.
It was proven that this tool is able to generate optimized circuits without any need of schematics and test benches creating by optimization of the two-stage Miller OTA and the voltage regulator.

REFERENCES

[1] A. Jarafi, S. Sadri and M. Zektri, "Design optimization of analog integrated circuits by using artificial neural networks," International Conference of Soft Computing and Pattern Recognition, (Paris), pp. 385-388, 2010.

[2] L. Bo, W. Yan, Y. Zhiping, L. Leibo, L. Miao, W. Zheng, L. Jing, V. F. Francisco, "Analog circuit optimization system based on hybrid evolutionary algorithms," Integration, vol. 42, no. 2, pp. 137-148, 2009.

[3] M. Fakhfakh, "A novel Alienor-based heuristic for the optimal design of analog circuits," Microelectronics Journal, vol. 40, no. 1, pp. 141-148, 2009.

[4] E. Tlelo-Cuautle, I. Guerra-Gomez, M. A. Dudarte-Villasenor, L. G. De la Fraga, G. Flores-Becerra, G. Reyes-Salgado, C. A. Reyes-Garcia, G. Rodriguez-Gomez, "Applications of evolutionary algorithms in the design automation of analog integrated circuits," Journal of Applied Sciences, vol. 10, pp. 1859-1872, 2010.

[5] M. F. M. Barros, J. M. C. Guilherme, N. C. G. Horta, "State-of-the-art on analog design automation, " Studies in Computational Intelligence, vol. 294, pp. 19-47, 2010.

[6] A. Bennour, A. Sallem, M. Kotti, E. Gaddour, M. Fakhfakh, M. Loulou, "Application of the PSO technique to the optimization of CMOS operational transconductance amplifiers," 5th International Conference on Design and Technology of Integrated Systems in Nanoscale Era, (Hammamet), pp. 1-5, 2009.

[7] S. L. Sabat, K. S. Kumar, S. K. Udgata, "Differential evolution and swarm intelligence techniques for analog circuit synthesis," World Congress on Nature & Biologically Inspired Computing, (Coimbatore), pp. 469-474, 2009.

[8] R. A. Thakker, M. S. Baghini., M. B. Patil, "Low-power low-voltage analog circuit design using hierarchical particle swarm optimization," 22nd International Conference on VLSI Design, (New Delhi), pp. 427-432, 2009.

[9] P. Prem Kumar, K. Duraiswamy, "An optimized device sizing of analog circuits using particle swarm optimization," Journal of Computer Science, vol. 8, pp. 930-935, 2012.

[10] J. Maršík, O. Šubrt, P. Martinek, "Developing automated design procedure for operational amplifier blocks," International Conference on Signals and Electronic Systems, (Krakow), pp. 269-272, 2008.

[11] M. Barros, J. Guilherme, N. Horta, "Analog circuits optimization based on evolutionary computation techniques, " Integration, vol. 43, no. 1, pp. 136-155, 2009.

[12] L. Gao-Yang, L. Ming-Guang, "The summary of differential evolution algorithm and its improvements," 3rd International Conference on Advanced Computer Theory and Engineering, (Chengdu), vol. 3, pp. V3-153 – V3-156, 2010.

[13] F. J. Ahlers, W. Di Carlo, C. Fleiner at al. Differential Evolution. [cit. 2013-08-13] Available from WWW: <http://www.icsi. berkeley.edu/~storn/code.html>.

[14] P. Martinek, D. Ticha, "The improved DE algorithm for filter design," Radioelektronika, pp. 1-4, 2008.

[15] A. Youyun, C. Hongqin, "Experimental study on differential evolution strategies," Global Congress on Intelligent Systems, (Xiamen), vol. 2, pp. 19-24, 2009.