SIMULATED ANNEALING FOR SIZING OF INTEGRATED CIRCUITS IN SPICE

Jernej Olenšek[1], **Árpád Bűrmen**[1], **Janez Puhan**[1], **Tadej Tuma**[1]

[1]University of Ljubljana, Faculty of Electrical Engineering
1000 Ljubljana, Tržaška 25, Slovenia

jernej.olensek@fe.uni-lj.si(Jernej Olenšek)

Abstract

This paper presents a new optimization algorithm for automatic sizing of integrated circuits (IC) in SPICE. We refer to the new method as DESA. It is a hybrid between two very popular oprimization methods. The first one is differential evolution (DE) which is a robust population based optimization method and has received a lot of attention in the recent years. It was also sucessfully applied to many practical applications. The second method is the simulated annealing algorithm (SA) which is a fairly simple but very powerfull stochastic global optimization method. Combination of DE and SA (DESA) is expected to exploit good global search capabilities of SA and efficient search mechanism and fast convergence of DE. DESA is fairly simple to implement and has only a few parameters. In order to verify its performance in IC design it was implemented in SPICE OPUS simulation and optimization tool. It was compared with the multistart version of the constrained simplex method (multistart COMPLEX), which was already part of SPICE OPUS and produced good results in IC optimization. The performance of DESA and the multistart COMPLEX method was verified on seven real-world cases of IC design. The comparison in terms of the final solution quality and the number of required cost function evaluations showed that DESA outperformed the multistart COMPLEX method on all considered test cases.

Keywords: global optimization, simulated annealing, differential evolution, IC design

Presenting Author's Biography

Árpád Bűrmen was born in Murska Sobota, Slovenia in 1976. He received his Uni.Dipl.-Ing. degree and his Ph.D. degree from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia in 1999 and 2003, respectively. Since 2002 he has been a teaching assistant at the Faculty of Electrical Engineering. His research interests include continuous and event driven simulation of circuits and systems, optimization methods, their convergence theory and applications, and algorithms for parallel and distributed computation.

1 Introduction

Design of integrated circuits (IC) is a very challenging and extremely time consuming task. The whole process can be roughly divided in three major steps. First the desired circuit performance has to be specified (design goals). According to these specifications an appropriate circuit topology must be selected. Finally the device parameters need to be determined (device sizing) so that the circuit meets the design specifications.

In the past the last two steps were completely dependent on designer's knowledge and experience. They also required an enormous amount of time since every circuit had to be tested manually. With fast development of computer technology the automation of the design process became a very attractive aspect and many computer-aided design (CAD) tools have been developed. Although the circuit topology selection still depends greatly on the experience of the designer, the amount of time and human effort needed to determine the appropriate device parameters can be greatly reduced with the use of modern CAD tools.

Once the topology has been selected, the quality measure of the circuit with the given device parameters must be defined. In the past the quality of the circuit was determined through measurements of a real circuit, which was extremely expensive and time consuming. Today it is obtained through numerical simulations of a circuit with the help of circuit simulators such as SPICE. The quality of the circuit is expressed as a real-valued function, referred to as the cost function (CF). It is a function of device parameters and includes all considered design goals (gain, bandwidth, noise, offset, delay,...). Large CF values indicate low quality circuits, while the values less than or equal to zero indicate that all design goals are satisfied or even surpassed. The problem of finding the optimal circuit can then translated into a global optimization problem (GOP) as described by Eq. (1)

$$
\begin{aligned}
& x^* = \arg\min_{x \in S} f(x) \qquad (1)\\
& f : S \to \mathbb{R}\\
& S = \left\{ x, x \in \mathbb{R}^N, l(i) \le x(i) \le u(i), i = 1, ...N \right\}
\end{aligned}
$$

where $f(x)$ is the cost function, x is a N-dimensional vector of optimization variables (parameters), and $l(i)$ and $u(i)$ are the lower and the upper bound for the i-th variable, respectively. The optimization parameters in IC design are usually MOS transistor channel lengths and widths and their multiplier factors. Often resistances and capacitances, and sometimes bias currents or voltages are also included as optimization variables.

In practice GOP described by Eq. (1) often can not be solved analytically. When the CF value is obtained through numerical simulations or measurements no analytical expression for the CF is available. Since such optimization problems arise in virtually every field of research, industry, and economy, a very large number of different optimization methods was developed to solve the problem numerically. They are often designed to solve a very specific class of optimization problems, therefore the selection of the appropriate method for a specific problem is very important.

The optimization methods can be classified by three major criteria. According to gradient information we can classify the methods as gradient methods, which require derivatives of the CF, and direct search methods that rely only on the CF values at different points in the search space. The next criteria is the search mechanism. Deterministic methods use a systematic approach to search through the parameter space. The alternative are stochastic methods that rely on some kind of a random search process. The final criterion is the nature of the final solution. Local optimization methods are designed to find the nearest minimum as quickly as possible even if it is not the best minimum in the search space. Global methods on the other hand perform a thorough search and can find the true global minimum with high probability.

Not all classes of optimization methods are appropriate for IC optimization. In IC design the CF is obtained through numerical simulations, which always introduce numerical noise in the CF. The device characteristics are nonlinear therefore the CF itself is also expected to be highly nonlinear. Due to nonlinearity we can expect several local optimal solutions in the given parameter space S. Another problem is the size of S (the dimensionality of the problem). The complexity of modern circuits is increasing rapidly resulting in a large number of optimization parameters. The circuit is expected to be robust. It must meet design specifications under different environmental conditions and manufacturing process variations. All these facts make the optimization of ICs extremely difficult and time consuming, and many optimization methods inefficient and unreliable. The fastest gradient methods require derivatives of the CF, which are usually not available in IC design. Noise also reduces their usability. Large parameter space requires a stochastic search mechanism and a large number of local minima implies that fast local methods are not the best choice. A direct stochastic global optimization method is needed to successfully solve IC optimization problems.

In this paper we present a new global optimization method (DESA) and apply it to IC design problems. It is a hybrid between differential evolution (DE) and simulated annealing (SA) algorithm. Both of these methods have received a lot of attention in the recent years and were successfully applied to many practical problems. SA and DE also have their drawbacks. A combination of both methods is expected to maintain the good features of the original methods while avoiding their weaknesses.

This paper is organized as follows. In sections 2 and 3 the basic SA and DE algorithms are briefly described. Section 4 gives a detailed description of the DESA algorithm. Section 5 contains the experimental setup and the optimization results for several real-world IC design

problems. Section 6 contains the concluding remarks.

2 Simulated annealing (SA)

Simulated annealing (SA) is a stochastic global optimization algorithm that performs random sampling of the search space [1]. Its main feature is the mechanism controlling the transition from the current point (x) to a new point (x^n), generated by a random perturbation (δx) of point x. The transition mechanism is known as the Metropolis criterion and is defined as

$$P = \min(1, e^{-\frac{f(x^n)-f(x)}{T}}) \tag{2}$$

where $f(x^n)$ and $f(x)$ are the CF values at the trial and the current point, respectively. T is the current value of the temperature parameter, which controls the probability of accepting the transition from the x to x^n. Downhill transitions are always accepted, while uphill transitions depend on the value of T. At the beginning of the search T is set to a large value and uphill transitions are accepted with high probability. During the optimization T is reduced according to the specified cooling schedule which reduces the probability of accepting uphill transitions. When T has a very low value most uphill transitions are rejected and SA runs almost as a descent method.

One of the attractive features of SA is the fact that its convergence to the global minimum can be proved under certain assumptions. In [2] for example there are several convergence results for various SA algorithms. However such algorithms require an appropriate mechanism for generating the random step δx and a very slow cooling schedule, which often makes them too slow for practical purposes. Modified versions are often used to speed up the convergence. The convergence results no longer apply to modified versions but the algorithms still exhibit good performance.

3 Differential evolution (DE)

Differential evolution (DE) is a parallel direct search method that uses a population of M points to search for a global minimum of a function over a continuous search space [3]. For every point (target point x^{it}) in the current population a so called mutated point (x^m) is generated by adding a weighted difference of two randomly selected points (x^{ic1} and x^{ic2}) from the current population to the third point (x^{ic3}). The weight factor w is given by the user and usually belongs to the (0,1] interval. Then crossover between the current target point and the mutated point is applied to generate a trial point (x^g). Several crossover schemes have been reported in the literature. Binomial crossover where the crossover is applied independently to every variable with probability P_c, is often used. If the CF value at x^g is lower than at x^{it}, x^g will replace x^{it} in the next generation. The process is repeated until the maximal number of generations is reached. Eq. (3) describes the basic trial point generation mechanism

$$\begin{array}{l} \text{For every } it = 1, 2, ...M \\ \text{Select randomly } ic1, ic2, ic3 \in \{1, 2, ...M\} \\ ic1 \neq ic2 \neq ic3 \neq it \\ x^m = x^{ic3} + (x^{ic1} - x^{ic2}) \cdot w \\ \text{Select randomly } I_k \in \{1, 2, ...N\} \\ \text{For every } i = 1, 2, ...N \\ x^g(i) = \begin{cases} x^m(i) & \text{if } U[0,1] < P_c \text{ or } i = I_k \\ x^{it}(i) & \text{otherwise} \end{cases} \end{array} \tag{3}$$

where $U[0, 1]$ denotes a uniformly distributed random variable from [0,1] interval. I_k ensures that the generated trial point is not identical to some member of the current population.

4 DESA algorithm

SA and DE have their strengths but also drawbacks. SA is known to have good global search capabilities but its convergence is very slow due to inefficient random sampling of the search space. The selection of the appropriate sampling mechanism and more importantly the cooling schedule is also very difficult. DE on the other hand has a robust and efficient search mechanism but only solutions with lower CF value are accepted into the next generation. It can get trapped in a local minimum without any chances of escaping. DE also requires a large population to maintain the diversity of the generated trial points.

With DESA we hope to combine good global search capabilities of SA and efficient search mechanism of DE. The method uses a population of M samplers, and a combination of random sampling and the original DE operator. The method also uses the original Metropolis criterion to allow uphill transitions. Since the annealing schedule is one of the most problematic aspects of SA, we use a different approach. Instead of having a single sampler and decreasing the temperature with time we have multiple samplers operating at different constant temperatures. Temperature changes are achieved by exchanging the points between different samplers. There have already been several attempts to use such an approach to avoid the difficulties of selecting the appropriate cooling schedule [4, 5]. We also include a random sampling mechanism to maintain population diversity. For this purpose every sampler has a fixed parameter called the range which is used in the generation of random moves. Now the i-th sampler g^i (where $i = 1, 2, ...M$) can be fully defined by the following features:

1. temperature T^i, which is used in the Metropolis criterion
2. range R^i, which is used for random step generation
3. crossover probability P_c^i, which is used in DE operator

4. a point x^i in the search space S

Since different optimization parameters in IC design have values that can differ by several orders of magnitude, we normalize all optimization variables to the [0,1] interval. A detailed description of the DESA method is given in the following subsections.

4.1 User defined input parameters

The method uses some parameters that must be set by the user. They are the number of samplers $M \geq 4$ (population size), minimal temperature $T^M > 0$ (temperature for the last sampler), minimal range parameter $R^M > 0$ (range parameter for the last sampler), crossover probabilities for the first and the last sampler P_c^1, $P_c^M \in [0, 1]$, and the stopping distance $D_{stop} > 0$. Default values for these parameters are $M = 20$, $T^M = 10^{-6}$, $R^M = 10^{-6}$, $P_c^1 = 0.1$, $P_c^M = 0.5$ and $D_{stop} = 10^{-4}$.

4.2 Initialization of population

The initial population can be generated randomly but in our method we use an approach that allows more thorough exploration of the search space. Every optimization variable interval is first divided into M equal subintervals. Then M points are randomly generated so that every subinterval for every optimization variable is included in the initial population. This is very important in algorithms that use crossover operators. The values of parameters inside subintervals are chosen randomly.

4.3 Initialization of method parameters

At the beginning of the optimization run some additional method parameters must also be set. These parameters are the temperature, the range parameter and the crossover probability for every sampler. All of them are initialized in the same way. We use the values of the parameters for the first and the last sampler and an exponential function to calculate the values for the remaining samplers.

$$\begin{aligned} c_t &= \frac{1}{M-1} \cdot \log(\frac{T^1}{T^M}) \\ T^i &= T^1 \cdot e^{-c_t \cdot (i-1)}, i = 1, 2, ...M \end{aligned} \tag{4}$$

T^1 is the maximum temperature and is set to the CF difference between the worst and the best point in the initial population. The same procedure is then repeated for the range parameter R^i with $R^1 = 1$, and for the crossover probabilities P_c^i.

4.4 Trial point generation mechanism

In every iteration a single point is selected for improvement. We select the worst point in the current population but any point that is not the best point can be selected here. We denote the sampler that holds this target point with the superscript it. A trial point is generated using a combination of an operator similar to the original DE operator and a random move. First x^g is generated according to Eq. (3). Then a random step r is generated according to the Cauchy probability distribution with the range parameter R^{it}. A trial point x^g is then updated with the random step r. The procedure is given by Eq. (5).

$$\begin{aligned} r(i) &= R^{it} \cdot \tan(\pi \cdot (U[0, 1] - 0.5)) \\ x^g(i) &= x^g(i) + r(i) \\ i &= 1, 2, ...N \end{aligned} \tag{5}$$

All variables $x^g(i)$ that violate box constraints are set to a random value between $x^{it}(i)$ and the violated boundary value.

4.5 Acceptance criterion

In this phase of the algorithm the generated trial point x^g is submitted to the Metropolis criterion (Eq. (2)) with temperature T^{it}. If the Metropolis criterion is satisfied, x^g replaces x^{it} in the next generation. Better points are always accepted. If the trial point x^g is worse than the current target point, the transition depends on the sampler that holds the target point. If the target point x^{it} is located at the sampler with a high temperature (i.e. T^{it} is large), x^g will have a high probability of being accepted. If the target point is located at a low temperature, this probability will be low. With this mechanism the chances for the algorithm to escape from a local minimum are increased.

4.6 Acceleration

The method can use many different mechanisms to speed up the convergence. In our case we used a fairly simple procedure. Every time a new best point is found, we apply this mechanism. We construct a quadratic model function based on three collinear points in the search space. The first point is a randomly selected point from the current population. Points with higher CF values have higher probability of being selected. The probability of selecting the point il is given by Eq. (6).

$$P(il) = \frac{\frac{f(x^{il}) - f(x^{best})}{f(x^{worst}) - f(x^{best})}}{\sum_{j=1}^{M} \frac{f(x^j) - f(x^{best})}{f(x^{worst}) - f(x^{best})}} \tag{6}$$

x^{best} and x^{worst} denote the points from the current population with the lowest and highest CF values, respectively.

The second point is the centroid of the population points (c) and is calculated according to Eq. (7).

$$c = \frac{1}{M} \sum_{i=1}^{M} x^i \tag{7}$$

These two points define a search direction $d = c - x^{il}$. The third point p^3 is obtained by making a random move from x^{il} in the direction d. If the obtained

quadratic model function is not convex, the best of these three points is returned. For the convex case the minimum of the model function is returned. If the minimum of the model function violates box constraints, it is first contracted towards x^{il} until the violation is removed. The returned point replaces x^{il} if it has lower CF value.

4.7 Temperature transition

One of the main problems of the original SA algorithm is the selection of the appropriate cooling schedule. If the cooling is too fast the algorithm can get trapped in a local minimum and if the cooling is too slow the optimization takes too long to be of any use for practical purposes. In DESA the cooling schedule is not needed because temperature changes are achieved by simply exchanging points between samplers which operate at different but fixed temperatures. After every trial point generation, replacement, and acceleration phase we randomly select a sampler g^{is} from the population. Then samplers g^{it} and g^{is} exchange their points in search space with probability given by Eq. (8).

$$P = \min(1, e^{-(\frac{1}{T^{is}} - \frac{1}{T^{it}}) \cdot (f(x^{is}) - f(x^{it}))}) \qquad (8)$$

This mechanism is quite different from the original idea of SA. Here the idea is to always send better solutions to samplers with higher T and R (i.e. if $T^{is} > T^{it}$ and $f(x^{is}) \geq f(x^{it})$) but also allow the occasional transition of a better point to a sampler with lower T. If the point x^{is} is worse than x^{it} the situation is reversed. The mechanism reduces the probability of the exchange if the difference in the temperature values of the two samplers is too large. Samplers with small values of T work almost as a downhill method accepting mostly points that reduce the CF value. Their value of R is small so the random component in trial point generation is also small and the applied operator is very much like the original DE operator. These samplers execute an algorithm very similar to DE. When a good point is found, the next point to be improved is likely to be at samplers with higher T and R so the algorithm runs at least for a while like a random search allowing longer jumps through the search space and making uphill transitions with higher probability. If an acceptable solution is not found the point eventually ends up at samplers with small T and R and the whole process is repeated. This scheme also performs a kind of reannealing and further improves the chances of escaping from a local minimum.

4.8 Stopping conditions

Several termination criteria can be used in our method. In practice the time available for the optimization is always limited so the maximal number of function evaluations is a logical choice for termination. The maximal distance between points in the population and the current best point is also used in the termination condition. When this distance falls below a user-defined stopping distance D_{stop} the algorithm is terminated. The third termination criterion is the CF value difference between the best and the worst point in the current population. When this difference becomes smaller than the user-defined minimal temperature (T^M) the algorithm is terminated. In IC design all the design goals are satisfied when the CF value reaches zero (see [6]). This can also be used in the termination condition.

5 Optimization of integrated circuits

Many different circuit simulators exits and one of the most popular is SPICE. The original SPICE is an open source program and due to its huge success the core can still be found in many modern CAD tools. SPICE OPUS [7] is a version of the original SPICE but unlike many other versions it is designed to allow automatic circuit optimization. Several optimization methods are already implemented in SPICE OPUS. The method that produced good results in IC optimization is a version of the simplex algorithm referred to as the constrained simplex method (COMPLEX) [8]. The original COMPLEX method is a local procedure therefore a multistart concept was implemented in SPICE OPUS to improve the methods global search capabilities. Every time the COMPLEX method reaches its termination conditions, a new simplex is initialized in the unexplored parts of the search space and the process is repeated until the maximal number of CF evaluations is reached [9]. The concept has proved to be fairly successful in IC design but it is sometimes slow and unreliable. DESA method is new and was implemented in C language as a part of SPICE OPUS and is expected to achieve better performance than the multistart COMPLEX method.

In IC optimization the definition of the CF must account for all circuit properties for which the design goals are set. Usually there are many conflicting design goals (i.e. for gain, bandwidth, power consumption, etc.). This makes the optimization very difficult. In addition, the circuit is expected to satisfy the design goals under different environmental conditions. In SPICE OPUS This is achieved by simulating the circuit across several corner points. Every combination of the environmental parameters (such as the temperature, circuit load, power supply voltage, model parameters, etc.) is represented by a corner point. For every corner point simulations are conducted resulting in real values that measure circuit properties in different corners. The worst values of circuit properties are then used to construct the penalty functions. The value of the CF is obtained as a weighted sum of all penalty functions [6]. When all design goals are satisfied, all penalty functions and thus the CF itself have the value zero.

5.1 Test cases

The performance of DESA method was tested on 7 real-world IC design problems and compared with the performance of the multistart COMPLEX method. We will describe in detail only the first case (damp1) which is a differential amplifier circuit. The circuit topology is depicted by Fig. 1.

There are 27 optimization variables:

- 3 resistors → 3 optimization variables

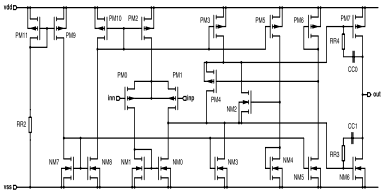

Fig. 1 The topology of the damp1 test circuit

- 2 capacitors $\rightarrow$ 2 optimization variables
- transistors NM0 and NM1 should be identical $\rightarrow$ 2 optimization variables (width and length)
- transistors NM3, NM5, NM7, and NM8 should be identical $\rightarrow$ 2 optimization variables (width and length)
- transistors PM0 and PM1 should be identical $\rightarrow$ 2 optimization variables (width and length)
- transistors PM2, PM3, PM5, and PM10 should be identical $\rightarrow$ 2 optimization variables (width and length)
- transistors PM9 and PM11 should be identical $\rightarrow$ 2 optimization variables (width and length)
- transistors NM2, NM4, NM6, PM4, PM6, and PM7 $\rightarrow 6 \cdot 2 = 12$ optimization variables (widths and lengths)

In this case we do not optimize transistor multipliers.

The properties which we are interested in are: circuit area, current consumption, AC gain, unity gain bandwidth, bandwidth, phase margin, gain margin, maximal derivative of gain magnitude, output voltage swing, DC gain, settling time, overshoot, slew rate, rise time, and fall time.

In order to measure these properties we need to perform the following analyses: an operating point analysis, a DC analysis, an AC analysis, and a transient analysis.

We only consider a single corner point with typical model parameters and ambient temperature of 27 degrees centigrade.

The remaining cases will be described more briefly. The second case (damp1-5c) optimizes the same circuit as the first case, with the same optimization parameters and design goals, but considers five different corner points to account for varying environmental conditions.

The third case (lfbuffer) is a circuit with 36 optimization parameters (32 transistors, 1 capacitor and 1 resistor). It requires an OP, an AC, a DC, and a transient analysis to measure 13 circuit properties chosen as design goals. The case considers a single corner point.

The fourth case (lfbuffer-5c) is the same as the third one but considers five different corner points.

The next case (nand) is a simple NAND gate element with only 3 optimization parameters (4 transistors) and considers 3 corners. The case requires an OP analysis and two transient analyses to measure 9 circuit properties.

The delay case (delay) has 12 optimization parameters (6 transistors) and a single corner point. It requires an OP analysis and a transient analysis to obtain 6 considered circuit properties.

The last case (damp2) is another amplifier circuit with 15 optimization parameters (9 transistors, 1 capacitor and 1 resistor) and 14 corner points. We perform an OP, a DC, two AC, a transient, and a noise analysis to measure 13 circuit properties.

All considered test cases were optimized using the default parameter values for both methods. The methods were compared in terms of the final solution quality and the number of CFE needed to reach the solution. Since the optimization is extremely time consuming, every circuit was optimized only once.

5.2 Results

Optimization results are given in Table 1. For every case the number of design variables (VARS), the number of design goals (GOALS), and the number of corner points (CORNERS) is given. The table shows the number of CFE needed to reach a point with the given CF value, the CF value of the final solution, the number of CFE needed to find the final solution, and the number of CFE when the method was terminated. For the multistart COMPLEX method the number of restarts needed to reach a particular solution is given in parentheses. Circuits for which the final CF value is zero have satisfied all the design goals and the optimization was stopped at that time even though the convergence has not occurred yet.

Tab. 1 IC optimization results

Case		DESA	multistart COMPLEX
damp1	CFE for CF < 1	5 843	5 208 (2)
	CFE for CF < 0.5	7 818	22 699 (6)
VARS=27	final CF	0	0.087
GOALS=15	CFE for minimal CF	63 863	68 471 (15)
CORNERS=1	final CFE	63 863	100 000 (24)
damp1-5c	CFE for CF < 10	2 490	1 356 (1)
	CFE for CF < 5	3 598	21 281 (5)
VARS=27	final CF	1.806	3.425
GOALS=15	CFE for minimal CF	250 021	231 312 (55)
CORNERS=5	final CFE	300 000	300 000 (73)
lfbuffer	CFE for CF < 10	1 781	414 (1)
	CFE for CF < 1	9 523	1 339 (1)
VARS=36	final CF	0	0.512
GOALS=13	CFE for minimal CF	79 377	3 310 (1)
CORNERS=1	final CFE	79 377	100 000 (24)
lfbuffer-5c	CFE for CF < 10	1 941	989 (1)
	CFE for CF < 5	5 852	13 523 (4)
VARS=36	final CF	2.330	4.313
GOALS=13	CFE for minimal CF	229 658	157 612 (41)
CORNERS=5	final CFE	300 000	300 000 (78)
nand	CFE for CF < 500	282	40 (1)
	CFE for CF < 200	507	94 (1)
VARS=3	final CF	166.641	166.686
GOALS=9	CFE for minimal CF	2 208	5 818 (47)
CORNERS=3	final CFE	2 986	10 000 (81)
delay	CFE for CF $< 20 \cdot 10^3$	38 698	660 (1)
	CFE for CF $< 10 \cdot 10^3$	84 302	21 101 (28)
VARS=12	final CF	0	6 183.500
GOALS=6	CFE for minimal CF	183 687	114 794 (143)
CORNERS=1	final CFE	183 687	200 000 (250)
damp2	CFE for CF < 20	989	420 (1)
	CFE for CF < 10	11 282	3 926 (3)
VARS=15	final CF	5.926	7.487
GOALS=13	CFE for minimal CF	251 244	326 011 (231)
CORNERS=14	final CFE	365 343	500 000 (352)

It can be seen from the table that DESA outperforms the multistart COMPLEX method on all considered cases

in terms of the final solution quality (final CF). Since the multistart COMPLEX method is designed to run without limitations on the number of CFE, it was manually stopped once the maximal number of CFE was reached. For DESA the convergence can occur earlier, depending on the stopping criteria described in section 4.8. The multistart COMPLEX method was allowed to run at least as long as DESA.

For the first case (damp1) DESA was able to find a solution that satisfied all the design specifications using 63 863 CFE. The multistart COMPLEX method performed 15 restarts with approximately the same number of CFE. Since it did not satisfy the design goals, it was allowed to perform the additional 9 restarts but was still unable to find a better solution even after 100 000 CFE.

In the second case (damp1-5c) neither method was able to satisfy all the design goals. Multiple considered corner points mean that a more robust circuit is required which may not be possible to obtain with the given topology, parameter bounds, or design goals. The problem with several corners is much more complex and more CFE are required to find good solutions. In this case both methods were run until the CFE limit was reached. The limit was set to 300 000 CFE. Although the design goals were not completely satisfied, DESA was still able to find a better solution than the multistart COMPLEX method.

In the third case (lfbuffer) DESA was again able to find the global minimum. It required 79 377 CFE. The progress of the multistart COMPLEX method was very fast and it found a fairly good solution in the first run. But even after it was allowed to perform more CFE than DESA, it was unable to find the global minimum.

In the fourth case (lfbuffer-5c) both methods failed to satisfy all the design goals. They were both terminated after the CFE limit was reached, which was set to 300 000. Despite 41 restarts the multistart COMPLEX method was unable to find a better solution than DESA.

The next case (nand) was the smallest among all the considered cases. Again neither method was able to find a solution that would satisfy all the design goals. DESA reached the termination condition after 2 208 CFE. The multistart COMPLEX method was allowed to perform 10 000 CFE (81 restarts) before it was terminated but its final CF value was still slightly higher.

For the delay case, DESA found the global minimum after 183 687 CFE. The initial progress of the multistart COMPLEX method was considerably faster. It was able to perform many restarts before DESA got even close to solutions with low CF values. But slow initial progress implies that the method searches more thoroughly through the search space. This improves its chances of finding the global minimum. The multistart COMPLEX method was allowed to run even longer but despite 250 restarts the final CF values was still rather high.

The last case (damp2) is very complex due to a large number of corner points. Both methods failed to satisfy all the design goals. DESA reached its termination conditions after 365 343 CFE. The multistart COMPLEX method again exhibited fast initial progress. Even after using considerably more CFE than DESA and despite 352 restarts it was unable to find a better solution.

We will discuss the optimization results in detail for the first case only (damp1). Table 2 shows the desired and measured properties at the final solution for both methods. One can see that both methods were able to satisfy most of the design goals. The multistart COMPLEX method found the solution where some of the properties are even better than those found DESA. This however comes at a price of not satisfying the output swing requirement. DESA on the other hand was able to satisfy all the design goals with a considerably lower number of CFE.

Tab. 2 Results for damp1 case

Measurement	goal	DESA	multistart COMPLEX
circuit area	$< 10^{-8}\mathrm{m}^2$	$8 \cdot 10^{-9}\mathrm{m}^2$	$5.95 \cdot 10^{-9}\mathrm{m}^2$
current consumption	$< 1\mathrm{mA}$	$437 \cdot 10^{-6}\mathrm{A}$	$530 \cdot 10^{-6}\mathrm{A}$
AC gain	$> 70\mathrm{dB}$	$70.4\mathrm{dB}$	$70\mathrm{dB}$
unity gain bandwidth	$> 5\mathrm{MHz}$	$17.2\mathrm{MHz}$	$15.7\mathrm{MHz}$
bandwidth	$> 500\mathrm{Hz}$	$1.38\mathrm{kHz}$	$2.23\mathrm{kHz}$
phase margin	$> 60^o$	65.9^o	90.3^o
gain margin	$> 10^o$	33.4^o	15.6^o
max.derivative of gain magnitude	< 0	$-104 \cdot 10^{-9}$	$-39.9 \cdot 10^{-9}$
output voltage swing	$> 1.6\mathrm{V}$	$1.6\mathrm{V}$	$1.57\mathrm{V}$
DC gain	$> 60\mathrm{dB}$	$69.3\mathrm{dB}$	$66.7\mathrm{dB}$
settling time	$< 300\mathrm{ns}$	$174\mathrm{ns}$	$168\mathrm{ns}$
overshoot	$< 1\%$	$696 \cdot 10^{-3}\%$	$885 \cdot 10^{-3}\%$
slew rate	$> 5 \cdot 10^6\mathrm{V/s}$	$7.44 \cdot 10^6\mathrm{V/s}$	$7.48 \cdot 10^6\mathrm{V/s}$
rise time	$< 200\mathrm{ns}$	$64.4\mathrm{ns}$	$64.1\mathrm{ns}$
fall time	$< 200\mathrm{ns}$	$57.5\mathrm{ns}$	$81.1\mathrm{ns}$

In order to demonstrate the difficulties of IC optimization we calculated the profile of the CF for the damp1 test case. Centered at some initial point we performed a sweep through all 27 optimization parameters. Fig. 2 shows the profile when sweeping only through 3 parameters but it is enough to see the problems of IC optimization. The curves intersect at a point with x-axis value zero representing the center point of the profile. The CF is highly nonlinear and several local minima are clearly visible in the profile. The sensitivity of the CF to different parameters varies considerably and noise is also easy to notice. All these facts make fast gradient descent methods unreliable and inefficient, and the entire optimization task extremely difficult. When there are several corner points to consider, the task becomes even more challenging. The quality of the initial point (center point of the profile) is quite bad. The CF value is not zero which means that the solution does not satisfy the design goals. What is more, changing a single parameter can already produce a point with lower CF value.

Fig. 3 shows the profile of the same CF for the same optimization parameters but it is centered at the final solution found by the DESA method. One can still notice

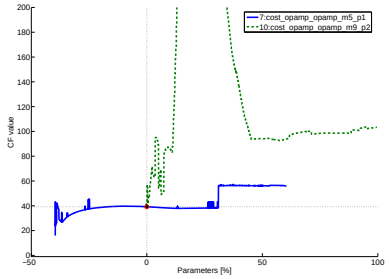

Fig. 2 The profile of the CF, when sweeping through two of the optimization parameters (centered at the initial solution).

noise, nonlinearity, and local minima, but the quality of the final solution (point with x-axis value zero) is also clear. Changes of any single parameter do not produce a point with CF value lower than at the center point of the profile.

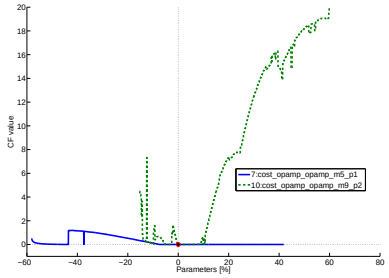

Fig. 3 The profile of the CF, when sweeping through two of the optimization parameters (centered at the final solution found by DESA method).

6 Conclusions

IC optimization is a very difficult and time consuming task and requires a careful selection of the appropriate optimization method. Noisy CF with several local minima and large search space dictate the use of direct stochastic global optimization methods. In this paper a new hybrid global optimization algorithm (DESA) was presented. It combines good global search capabilities of the simulated annealing algorithm and the efficient search mechanism of differential evolution. DESA is fairly simple to implement and has only a few parameters. Since it is a global optimization method, it is expected to require a large number of CFE. DESA was implemented in C language as a part of SPICE OPUS simulation and optimization tool. Experiments were conducted on seven real-world cases of IC design to evaluate the performance of the method. Comparison was made with a version of the simplex algorithm (multistart COMPLEX method) which is already integrated as a part of SPICE OPUS and produced good results for IC design problems. Experimental results have shown that DESA outperformed the multistart COMPLEX method in terms of global search capabilities on all considered test cases.

7 Acknowledgment

The research has been supported by the Ministry of Higher Education, Science and Technology of the Republic of Slovenia within programme P2-0246 - Algorithms and optimization methods in telecommunications.

8 References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:1277–1292, 1983.

[2] R. L. Yang. Convergence of the simulated annealing algorithm for continuous global optimization. *Journal of optimization theory and applications*, 104(3):691–716, 2000.

[3] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimizazation*, 11:341–359, 1997.

[4] G. L. Bilbro. Fast stochastic global optimization. *IEEE Trans. Syst., Man, Cybern*, 24(4):684–689, 1994.

[5] D. R. Thompson and G. L. Bilbro. Sample-sort simulated annealing. *IEEE Trans. Syst., Man, Cybern (B)*, 35(3):625–632, 2005.

[6] A. Bűrmen, D. Strle, F. Bratkovič, J. Puhan, I. Fajfar, and T. Tuma. Automated robust design and optimization of integrated circuits by means of penalty functions. *AEU-International journal of electronics and communication*, 57(1):47–56, 2003.

[7] http://www.fe.uni-lj.si/spice/.

[8] M. J. Box. A new method of constrained optimization and a comparison with other methods. *Computer Journal*, 8:42–52, 1965.

[9] J. Puhan, A. Bűrmen, and T. Tuma. Analogue integrated circuit sizing with several optimization runs using heuristics for setting initial points. *Canadian journal of electrical and computer engineering*, 28(3-4):105–111, 2003.