Journal of Circuits, Systems, and Computers
Vol. 17, No. 1 (2008) 123–140
© World Scientific Publishing Company

A COMBINED SIMPLEX–TRUST-REGION METHOD FOR ANALOG CIRCUIT OPTIMIZATION

ÁRPÁD BŰRMEN*, TADEJ TUMA and IZTOK FAJFAR

Faculty of Electrical Engineering,
University of Ljubljana,
Tržaška Cesta 25, SI-1000 Ljubljana, Slovenia
**arpadb@fides.fe.uni-lj.si*

Revised 31 July 2007

The analog-integrated circuits industry is exerting increasing pressure to shorten the analog circuit design time. This pressure is put primarily on the analog circuit designers that in turn demand automated circuit design tools evermore vigorously. Such tools already exist in the form of circuit optimization software packages but they all suffer a common ailment — slow convergence. Even taking into account the increasing computational power of modern computers the convergence times of such optimization tools can range from a few days to even weeks. Different authors have tried diverse approaches for speeding up the convergence with varying success. In this paper authors propose a combined optimization algorithm that attempts to improve the speed of convergence by exploiting the positive properties of the underlying optimization methods. The proposed algorithm is tested on a number of test cases and the convergence results are discussed.

Keywords: Computer-aided design; analog circuits; optimization algorithms.

1. Introduction

The ever-increasing pressure to shorten the time-to-market exerted by the integrated circuit (IC) industry is fueling the development of design automation methods and software.[1] In the beginning of each IC design process, designers get a list of the so-called design requirements formulated as value intervals for every circuit performance measure (e.g., gain, phase margin, delay, etc.).

The requirement that the performance measure intervals must be obeyed for a whole range of operating conditions and manufacturing process variations further increases the complexity of the design problem. This demand is the basic characteristic of a robust IC.

One usually practices a robust design by evaluating the circuit characteristics for various extreme combinations of operating conditions and manufacturing process variations (corners). By keeping the circuit performance measures within their prescribed intervals across a cleverly chosen set of corners, a robust IC is obtained.

In the design process the designer first chooses the circuit topology that has the potential to fulfill all of the design requirements for the prescribed range of operating conditions and manufacturing process variations. The next step is the circuit sizing where the values of the elements are chosen for the given topology. This step can be automated by means of parametric optimization.

Parametric optimization is a process where a real-valued function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is minimized. This function is often referred to as the cost function (CF). Transforming the search for a better circuit into the search for the lowest value of f translates the initial design problem handled by humans into an optimization problem that can be handled by a computer. A suitable transformation can be constructed by means of penalty functions. Such CF formulation has been shown to yield robust designs.[2–4]

The basic idea of the penalty function approach is to add up the contributions of individual performance measures transformed by penalty functions. The form of each penalty function is determined by the type of the design requirement for a particular performance measure (e.g., measure value must be below some goal, above some goal, etc.). Generally the penalty is proportional to the amount the goal is violated. Let $\mathcal{C}$ denote the set of corner points, m_i the ith performance measure, and n_i the corresponding penalty function. The CF can then be constructed as

$$f(\boldsymbol{x}) = \sum_i n_i(m_i(c_i^{\text{worst}}, \boldsymbol{x})) \,, \tag{1}$$

where $c_i^{\text{worst}} \in \mathcal{C}$ denotes the corner point in which the ith measure reaches its worst value.

Once f is constructed and can be evaluated by means of simulation and some basic arithmetics, it can be used as the input for an optimization algorithm. The goal of the optimization algorithm is to find the set of circuit parameter values $\boldsymbol{x} \in \mathbb{R}^n$ for which the value $f(\boldsymbol{x})$ is as low as possible, using as few CF evaluations as possible.

As circuit simulators provide no information on the value of the CF gradient, gradient-based optimization algorithms are not appropriate for circuit sizing. Ideally, the result of the optimization algorithm should be the global minimum of the CF. Unfortunately, dedicated global algorithms require many CF evaluations, while local algorithms easily get trapped in a local minimum caused by the numerical noise introduced into the CF during simulation.

A reasonable compromise between local and global behavior was found in the family of simplex algorithms.[5–8] As these algorithms manipulate at least $n + 1$ points at the same time, they exhibit a fair amount of noise independence. Another advantage is that they do not require CF gradient information. The initial set of points is usually chosen randomly causing the algorithm to exhibit some global behavior. Initially, a rapidly decreasing sequence of CF values is produced which together with the simplicity of implementation makes simplex algorithms very popular in the optimization community. Unfortunately, simplex algorithms are plagued by convergence problems. Convergence problems usually manifest themselves as slow progress in the vicinity of a local minimum.[9–11]

The goal of this paper is not to find a new global optimization method. Instead, we try to extend the simplex algorithm in a way that the convergence to a local minimum will be guaranteed (which is currently not the case). By ensuring local convergence we expect the algorithm to become faster (spend less function evaluations for finding the same result). Furthermore we also expect slightly better final results for cases where the original simplex algorithm stagnates. The latter is often encountered with higher dimensional problems (10 and above).

To overcome the shortcomings of the Box simplex algorithm, a two-stage algorithm is proposed. In the first stage the simplex algorithm finds the vicinity of a good local minimum. The second stage utilizes a local optimization algorithm to find the exact location of the minimum. Any local algorithm with a strong convergence theory could be used in the second stage. Due to the fact that the gradient of the CF is not available, a good candidate for the second part of the search is an algorithm from the family of trust-region methods.[12–14]

In the following sections we describe the modified Box simplex algorithm and the trust-region algorithm, and in the section describing the combined algorithm we discuss the accompanying switch condition. We test the combined algorithm first on some simple mathematical functions to prove its feasibility, and then on real-world integrated circuit-sizing problems. The conclusions are given in the last section.

2. A New Two-Stage Algorithm

In the combined two-stage algorithm the goal of the first stage is to bring the iterates in the neighborhood of a CF local minimum. Any method that performs well in practice can be used for this purpose. Our method of choice is the modified Box simplex algorithm.[6] The second stage pinpoints the minimum. The method used in the second stage must have a strong convergence theory. A method from the trust-region family is used in the proposed combined algorithm.[12–14]

The Box simplex algorithm was chosen due to the results obtained in our past research.[7,8] These results have shown that despite the strong theoretical background of gradient-based algorithms they tend to produce poor results on real-world circuit optimization problems. Surprisingly, a simple heuristic algorthm like the Box simplex algorithm not only outperformed them, but even showed some global search capabilities which can be traced back to the random choice of the initial simplex.

Simplex algorithms build on the idea of moving a set of $M > n$ points (also referred to as the simplex) through the n-dimensional space in order to find a point with a CF value as low as possible. The worst point of the simplex is the one that is the candidate for the replacement. Using operations such as reflection through the centroid of the remaining $M - 1$ points and contraction toward the best point the algorithm tries to improve the worst point. This process is repeated until the simplex becomes small enough. The best point remains in the simplex as long as no better point is found.

The manipulation of M points instead of a single one results in a fair amount of noise resistance. By choosing these points randomly in the beginning of the search some global search capabilities are also obtained. Although simplex algorithms produce good results on practical optimization problems there is no guarantee that they are capable of finding even a local minimizer of the CF. The main reason for their popularity is their simplicity and the rapid initial decrease of the CF value they produce.

Trust-region algorithms form a local model of the CF. The model is assumed to be valid within a part of the search space also referred to as the trust region. The minimum of the model in the trust region is calculated. If the CF value at the minimum predicted by the model is lower than the CF value at the current point, the search moves to the newly found minimum. If the CF value predicted by the model agrees well with the true CF value, the trust region is expanded; otherwise, it is contracted. The search terminates when the trust region is sufficiently small.

Trust-region algorithms are typical local search algorithms that require no derivative information. Unlike the simplex algorithms their convergence to a local minimum of the CF is guaranteed under mild assumptions.[12–14]

2.1. *The modified Box simplex algorithm*

Let superscripts denote vector components in Cartesian coordinates. The original Box simplex method is capable of handling box constraints (explicit constraints) of the form[6]

$$b^i \leq x^i \leq B^i \quad i = 1, 2, \ldots, n\,. \tag{2}$$

Values b^i and B^i are lower and upper bounds imposed on parameter values and define a box $\mathcal{B}$ in $\mathbb{R}^n$.

The normalized distance between two points ($\boldsymbol{x}$ and $\boldsymbol{y}$) is obtained as

$$d(\boldsymbol{x}, \boldsymbol{y}) = 100 \cdot \sqrt{\sum_{i=1}^{n} \left(\frac{x^i - y^i}{B^i - b^i} \right)^2}\,. \tag{3}$$

The modified Box simplex algorithm can be summarized in the following steps:

(1) Build the initial simplex containing the initial point $\boldsymbol{x}_0$ and $M - 1 \geq n$ additional randomly chosen points from $\mathcal{B}$. Evaluate the CF at the M points of the initial simplex. Choose $\alpha > 1$ and $\beta \in (0, 1)$.
(2) Sort the points of the simplex such that $f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}_1) \leq \cdots \leq f(\boldsymbol{x}_{M-1})$.
(3) Compute the centroid of the $M - 1$ best points: $\overline{\boldsymbol{x}} = (M-1)^{-1} \sum_{i=0}^{M-2} \boldsymbol{x}_i$.
(4) Mirroring. Mirror the worst point across the centroid to obtain $\boldsymbol{x}_\mathrm{r} = \overline{\boldsymbol{x}} + \alpha(\overline{\boldsymbol{x}} - \boldsymbol{x}_{M-1})$. If any component of $\boldsymbol{x}_\mathrm{r}$ violates box constraints, it gets truncated to the violated constraint value.

(5) If $f(\boldsymbol{x}_\mathrm{r}) < f(\boldsymbol{x}_{\boldsymbol{M-1}})$, replace $\boldsymbol{x}_{M-1}$ with $\boldsymbol{x}_\mathrm{r}$ and go to step 12.
(6) Set $\boldsymbol{x}_\mathrm{c} := \boldsymbol{x}_\mathrm{r}$.
(7) Contraction toward centroid. While $f(\boldsymbol{x}_\mathrm{c}) \geq f(\boldsymbol{x}_{M-1})$ and $d(\boldsymbol{x}_\mathrm{c}, \overline{\boldsymbol{x}}) \geq \gamma$, set $\boldsymbol{x}_\mathrm{c} := \overline{\boldsymbol{x}} + \beta(\boldsymbol{x}_\mathrm{c} - \overline{\boldsymbol{x}})$.
(8) If $f(\boldsymbol{x}_\mathrm{c}) < f(\boldsymbol{x}_{M-1})$, replace $\boldsymbol{x}_{M-1}$ with $\boldsymbol{x}_\mathrm{c}$ and go to step 12.
(9) Set $\boldsymbol{x}_\mathrm{c} := \boldsymbol{x}_\mathrm{r}$.
(10) Contraction toward best point. While $f(\boldsymbol{x}_\mathrm{c}) \geq f(\boldsymbol{x}_{M-1})$ and $d(\boldsymbol{x}_\mathrm{c}, \boldsymbol{x}_0) \geq \gamma$, set $\boldsymbol{x}_\mathrm{c} := \boldsymbol{x}_0 + \beta(\boldsymbol{x}_\mathrm{c} - \boldsymbol{x}_0)$.
(11) If $f(\boldsymbol{x}_\mathrm{c}) < f(\boldsymbol{x}_{M-1})$, replace $\boldsymbol{x}_{M-1}$ with $\boldsymbol{x}_\mathrm{c}$; otherwise replace it with $\boldsymbol{x}_0$.
(12) Check the stopping condition. Go back to step 2 if the condition is not satisfied.

The stopping condition is based on the simplex size $\overline{d}$. When the simplex size falls below γ_s, the algorithm stops. Before the simplex size is evaluated the points are sorted according to their CF value and the new centroid ($\overline{\boldsymbol{x}}$) is evaluated, upon which $\overline{d}$ is obtained from

$$\overline{d} = \frac{1}{M-1} \sum_{i=0}^{M-2} d(\overline{\boldsymbol{x}}, \boldsymbol{x}_i)\,. \tag{4}$$

The following values of parameters were used: $M = 2n$, $\alpha = 1.3$, $\beta = 0.5$, and $\gamma = \gamma_\mathrm{s} = 0.001$.

2.2. *The l_∞ trust-region algorithm*

Trust-region methods solve for the minimum of a model function representing the CF on a subset of the search space also referred to as the trust region ($\mathcal{T}$). In our implementation the search space is normalized in such a manner that interval $[0, 100]$ corresponds to the intervals $[b^i, B^i]$ of individual circuit parameters. In this case the most appropriate trust region is square in shape. For a square trust region, the l_∞ norm must be used. The l_∞ norm is defined as

$$\|\boldsymbol{x}\|_\infty = \max_{i=1,\ldots,n} |x^i|\,. \tag{5}$$

The trust region $\mathcal{T}$ is then defined as

$$\mathcal{T}(\boldsymbol{x}_\mathrm{c}, r) = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_\mathrm{c}\|_\infty \leq r\}\,, \tag{6}$$

where $\boldsymbol{x}_\mathrm{c}$ is the center of the trust region and r is its radius.

The CF is evaluated at selected points in the trust region and based on the results a model of the CF $m(\boldsymbol{x})$ is built. The model is minimized over the trust region resulting in point $\boldsymbol{x}_m$. The CF is evaluated at $\boldsymbol{x}_m$. Based on the obtained amount of CF decrease, the algorithm decides whether the trust-region center $\boldsymbol{x}_c$ should move to $\boldsymbol{x}_m$. The agreement between the model and the CF at $\boldsymbol{x}_m$ is the basis for increasing or decreasing the trust-region radius r. After $\boldsymbol{x}_c$ and r are updated, a new model is built and the whole procedure is repeated until r becomes small enough.

We can summarize the trust-region algorithm in the following steps:

(1) Set $\boldsymbol{x}_c = \boldsymbol{x}_0$ and choose an initial trust-region radius $r > 0$. Choose $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq \gamma_2 < 1$.
(2) Choose a finite set of interpolation points $\mathcal{P} \subset \mathcal{B} \bigcap \mathcal{T}(\boldsymbol{x}_c, r)$. Evaluate CF at the points from $\mathcal{P}$ and build a model $m(\boldsymbol{x})$ of the CF.
(3) Minimize $m(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \mathcal{B} \bigcap \mathcal{T}(\boldsymbol{x}_c, r)$. Let $\boldsymbol{x}_m$ denote the resulting point.
(4) Evaluate $f(\boldsymbol{x}_m)$ and

$$\rho = \frac{f(\boldsymbol{x}_c) - f(\boldsymbol{x}_m)}{m(\boldsymbol{x}_c) - m(\boldsymbol{x}_m)} .$$

(5) If $\rho \geq \eta_1$ let $\boldsymbol{x}_c = \boldsymbol{x}_m$.
(6) Update r to r_{new}:

$$r_{\text{new}} \in \begin{cases} [r, \infty) ; & \rho \geq \eta_2 , \\ [\gamma_2 r, r) ; & \rho \in [\eta_1, \eta_2) , \\ [\gamma_1 r, \gamma_2 r) ; & \rho < \eta_1 . \end{cases} \tag{7}$$

(7) Check the stopping condition. Go back to step 2 if the condition is not satisfied.

Trust-region methods are often associated with quadratic models. Unfortunately such models require $(n + 1)(n + 2)/2$ sample points in order to determine the complete set of model parameters. In circuit optimization the number of parameters is very often 20 or more, which means that 231 or more points must be evaluated for a single model construction step. Furthermore, to obtain the model coefficients one must solve a linear system of order $(n+1)(n+2)/2$. If one takes into account that a single optimization run with the simplex algorithm typically takes a few thousand CF evaluations one can see that there can only be a few trust-region iterations before the combined algorithm starts to perform worse than the ordinary simplex algorithm.

A much better choice is a linear model of the form $m(\boldsymbol{x}) = \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{g}$ is the gradient vector of the model function. Linear models require only $n + 1$ points with known CF values and they can be constructed without solving a linear system. In our example, one of the $n+1$ points is the trust-region center ($\boldsymbol{x}_c$). The remaining n points are chosen by perturbing individual vector components to the edge of $\mathcal{B} \bigcap \mathcal{T}(\boldsymbol{x}_c, r)$ that lies further away from $\boldsymbol{x}_c$.

The stopping condition is based on the trust-region radius (r). When r falls below γ_{t}, the algorithm is stopped. The algorithm is somewhat more greedy than the ones described in the literature. At the beginning of step 2 the trust-region center ($\boldsymbol{x}_c$) is moved to the point with the best-yet CF value. This takes advantage of possible CF improvements obtained during the evaluation of points from $\mathcal{P}$ used for constructing $m(\boldsymbol{x})$.

The following values of parameters were used: $\gamma_{\mathrm{t}} = 0.001$, $\eta_1 = 0.01$, $\eta_2 = 0.9$. The trust-region radius update was obtained from

$$r_{\mathrm{new}} = \begin{cases} 2.5 \cdot r\,; & \rho \geq \eta_2\,, \\ 0.25 \cdot r\,; & \rho \in [\eta_1, \eta_2)\,, \\ 0.25 \cdot r\,; & \rho < \eta_1\,. \end{cases} \tag{8}$$

2.3. *The combined simplex–trust-region algorithm*

We can summarize the combined algorithm in the following steps:

(1) Choose the constants for both optimization methods.
(2) Run the modified Box simplex method until the neighborhood of a minimum is reached.
(3) Choose the initial trust-region radius and center.
(4) Run the trust-region method until the minimum is reached.

In the process of deciding when the neighborhood of a CF minimum is found, one has two sources of information that can be used. The first is the sequence of CF values as they were obtained by the simplex algorithm. The second one is the position of the simplex points.

The CF value decreases as the simplex algorithm progresses toward a CF minimum. Unfortunately, the initial CF value and the dynamics of the decrease are very different from case to case. Even if the CF is constructed in a standard way (see Refs. 2, 3, and 1) the differences are fairly large. Applying some monotonically increasing function to the CF value before it enters the decision-making process of the algorithm produces the same path through the search space, but the value of the CF can be completely different. This can misguide the combined algorithm into switching to a trust-region method too early or too soon if the point of switch is based on the CF value.

On the other hand the simplex shrinks as it closes in on a CF minimum. This shrinking is the cause that the simplex algorithm eventually stops (the stopping condition is based on the simplex size $\overline{d}$). The rate at which the simplex is shrinking gradually decreases as the algorithm approaches a local minimum.

In the combined algorithm, the simplex size and the rate of simplex size change are the triggers for the termination of the simplex algorithm. The rate of simplex size change is calculated from a sequence of 2-tuples $(i, \overline{d}_i)$, where i is the consecutive number of the CF evaluation and $\overline{d}_i$ the corresponding simplex size. Such 2-tuples are generated every time the simplex size is checked to see if the algorithm should terminate.

The criterion for transition to the trust-region algorithm is not checked until there are at least $j > M\mu_1$ 2-tuples in the sequence. Suppose the simplex algorithm is at the jth CF evaluation. Let $\mathcal{I}_j$ denote the set of indices i for which

$\max(j - M\mu_2, 0) \le i \le j$. Least squares fitting is used to obtain coefficients k_j and a_j

$$\min_{k_j, a_j} \sum_{i \in \mathcal{I}_j} (\overline{d}_i - (k_j i + a_j))\,. \tag{9}$$

The obtained slope of a fitted line (k_j) is usually negative as the simplex is mostly shrinking. The switch condition can then be expressed as

$$-k_j < k_{\mathrm{s}} n^{-1/2} \vee \overline{d}_j < \overline{d}_{\mathrm{s}}\,. \tag{10}$$

To make sure that the transition from the Box simplex algorithm to the trust-region algorithm always happens, one must make sure that the former terminates after a finite number of iterations. This can be achieved by making the CF piecewise-constant (e.g., using grid-restrainment).[15]

Definition 1. A grid restrainment operator $\mathcal{R}$ is a map $\mathbb{R}^n \mapsto \mathbb{R}^n$ such that $\boldsymbol{y} = \mathcal{R}(\boldsymbol{x})$ and

$$y^i = b^i + s^i \left\lfloor \frac{x^i - b^i}{s^i} + \frac{1}{2} \right\rfloor .$$

s^i is the maximal distance between the neighboring grid points in the ith dimension. The range of $\mathcal{R}$ is a grid. It is easy to prove that the intersection of a grid and a compact set is a finite set of points.

Now, let the Box simplex algorithm use the following function instead of $f(\boldsymbol{x})$:

$$\tilde{f}(\boldsymbol{x}) = f(\mathcal{R}(\boldsymbol{x}))\,. \tag{11}$$

$\tilde{f}$ is a piecewise-constant approximation of f. As the values of s^i decrease, $\tilde{f}$ becomes more accurate. The finiteness of the Box simplex algorithm (when its termination condition is expressed by Eq. (10)) is the consequence of the following theorem.

Theorem 1. *The range of $\tilde{f}$ on any compact domain is a finite subset of $\mathbb{R}$.*

Proof. Since the intersection of a grid and a compact set is a finite set, the range of $\mathcal{R}$ on a compact domain must be a finite set too. There exist finitely many values of f that correspond to this set. □

The final result can be summarized in the following theorem.

Theorem 2. *Suppose that f has compact level sets and that the Box simplex algorithm optimizes $\tilde{f}$ instead of f. Then there can only be finitely many function evaluations before the algorithm terminates due to Eq.* (10).

Proof. The compactness of level sets of f implies the same for the level sets of $\tilde{f}$. This in turn guarantees the existence of a ball $\mathcal{C}$ that includes the level set of the worst point in the initial simplex.

A point is accepted into the simplex only if it decreases the $\tilde{f}$ value at the worst point of the simplex. Therefore, the simplex remains within $\mathcal{C}$ regardless of how long the algorithm is running.

$\mathcal{C}$ is compact; so, $\tilde{f}$ has only finitely many different values in $\mathcal{C}$ (see previous theorem). Now recall again that a point is accepted into the simplex if its corresponding CF value is better than the one at the worst point. Together with the fact that there are only finitely many different values of $\tilde{f}$ in $\mathcal{C}$, this results in a finite number of points being accepted into the simplex. After that only contraction steps can occur that reduce the simplex diameter below any positive value in a finite number of CF evaluations. So, the second part of Eq. (10) is satisfied sooner or later. $\square$

When the switch condition (10) is satisfied, the Box algorithm assumes that it has found the neighborhood of a minimum and terminates. Next, the trust-region algorithm is started. The initial trust-region center ($\boldsymbol{x}_c$) is chosen to be at the best-yet point obtained by the simplex algorithm. The initial trust-region radius (r) is set to 100. This way the initial trust-region spans the complete search space, and the chance of getting stuck in a local minimum caused by numerical noise in the CF is reduced.[16]

The point of switch to the trust-region algorithm is determined by the values assigned to μ_1, μ_2, k_s, and $\overline{d}_\mathrm{s}$. The value of μ_1 must be large enough so that the simplex algorithm has a chance to probe around the search space. μ_2 specifies the amount of averaging for calculating the rate of simplex size change. If the value of μ_2 is too small, the intermediate increase in the size of the simplex (when a promising direction of search is found) can cause a premature switch to the trust-region algorithm.

The value of d_s specifies the point at which the trust-region algorithm replaces the simplex algorithm regardless of the rate of simplex size change. Typically this constant should be set to a value around 1.0 (the simplex size is 1% of the total search space size). Finally, k_s specifies the critical rate of simplex size change per iteration. When the rate of simplex size change drops below k_s, the trust-region algorithm takes over. Values around 0.2 (0.2% change in simplex size per iteration) tend to give good results.

The convergence properties of the two employed algorithms are very different. While there exists a strong convergence theory for trust-region algorithms[12–14] the convergence properties of simplex algorithms are almost unknown.[17] Furthermore, there exist several counter-examples.[9–11] On the other hand, simplex algorithms perform remarkably well on practical circuit optimization problems.[2–4,7,18] They rapidly decrease the CF value and exhibit resistance to noise. Due to the initial random choice of the simplex points, the obtained local minimum is quite good when compared to the results of other local algorithms.[7]

As the trust-region algorithm is run in the second part of the search, the convergence properties of the combined algorithm are determined by the convergence properties of the trust-region algorithm. This way the combined algorithm has mathematically provable convergence. The simplex algorithm which identifies the vicinity of a good local minimum in the first part of the search is followed by the trust-region algorithm which finds the actual minimum.

3. Testing the Combined Algorithm

The combined algorithm is a general optimization method. The use of a trust-region algorithm in the second part of the search makes it locally convergent. In our past research, the modified Box simplex algorithm (used in the first stage of the combined algorithm) emerged as most suitable for fast circuit optimization.[7] It also performed well on IC sizing problems.[2–4,18] Therefore, the combined algorithm was compared to the modified Box simplex algorithm. If the new algorithm outperforms the Box simplex algorithm, it outperforms several other algorithms[7] too.

We implemented both algorithms in the SPICE OPUS circuit simulation and optimization tool.[8]

3.1. *Test problems*

The comparison was done on 11 test problems comprising six circuits and three analytical functions.

The combined algorithm was first tested on three simple analytical cases. The first one (lin) was a linear function of 10 variables with a minimum in one of the corners of the search space. The second one (sq1) was a convex quadratic function of 10 variables with a minimum in the center of the search space. The third analytical case was a concave quadratic function of 10 variables (sq2) with a maximum in the center of the search space and local minima in its corners.

The nand test case (Fig. 1, bottom left) requires the optimizer to minimize the area, delay, and rise/fall time of a NAND gate. The delay test case (Fig. 1, top left) attempts to achieve the desired delay with a delay circuit and keep the power consumption and circuit area as low as possible.

The remaining test cases are various differential amplifiers that constitute the basic building blocks of most analog-integrated circuits. For all these circuits the goal is to maximize the performance of the circuit (e.g., gain, bandwidth, phase margin, etc.) while keeping the circuit area and power consumption as low as possible. The simplest amplifier is the damp1 test case (Fig. 1, top right) — a differential amplifier with single-ended output. The only amplifier with a differential output in our test suite is the ddamp case (Fig. 1, bottom right). The damp2 (Fig. 2, top) and lfbuf (Fig. 2, bottom) are differential amplifiers with single-ended outputs.

To illustrate a typical optimization problem, the damp2c test case will be described in more detail. The case has 27 optimization parameters. There are fewer optimization parameters than device parameters in the circuit (due to matched devices). Such devices are identical. Every MOS transistor group contributes two parameters (width and length). The MOS transistor groups are (NM0, NM1), NM2,

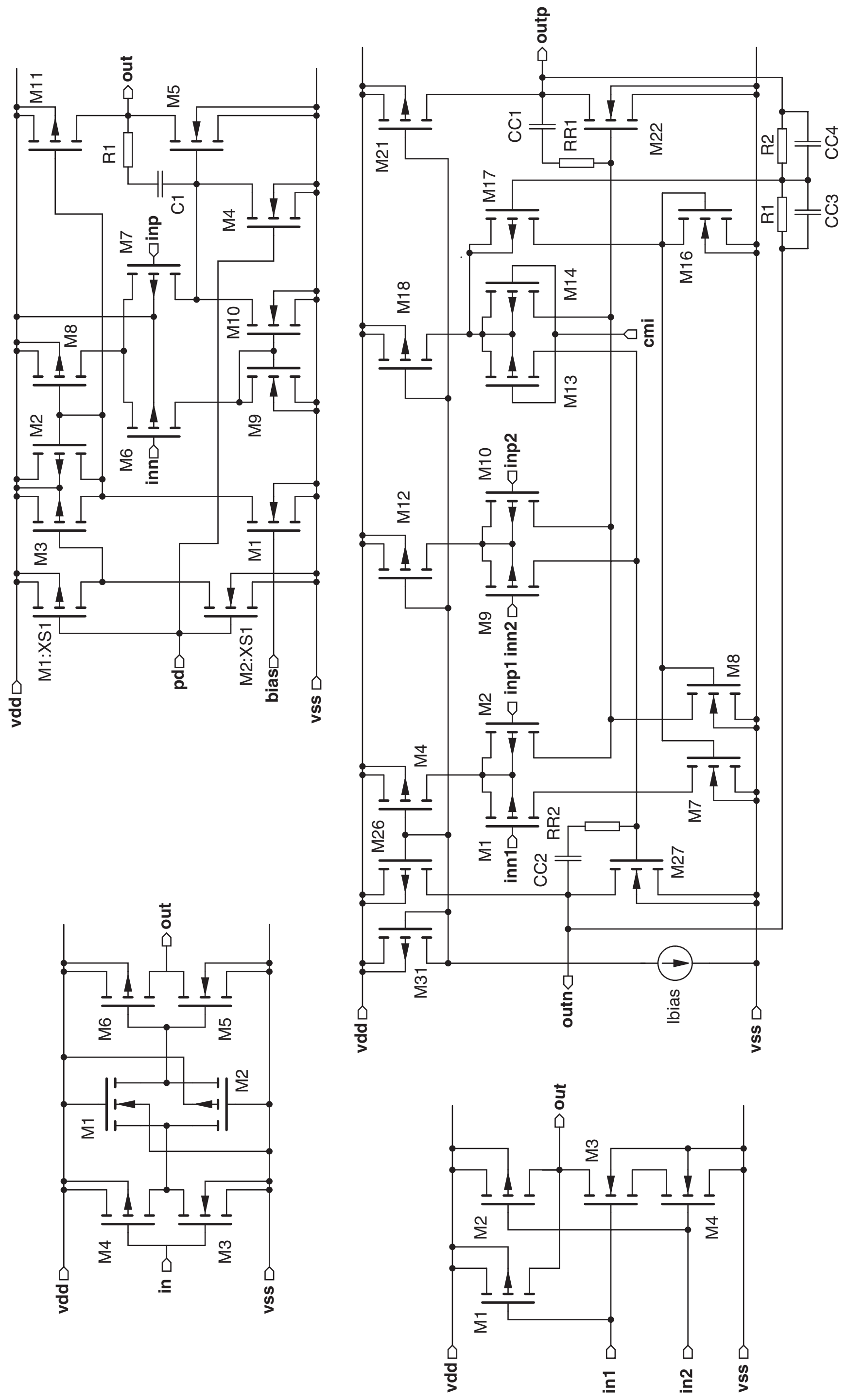

Fig. 1. Topologies of the delay (top left), nand (bottom left), damp1 (top right), and ddamp (bottom right) test circuits.

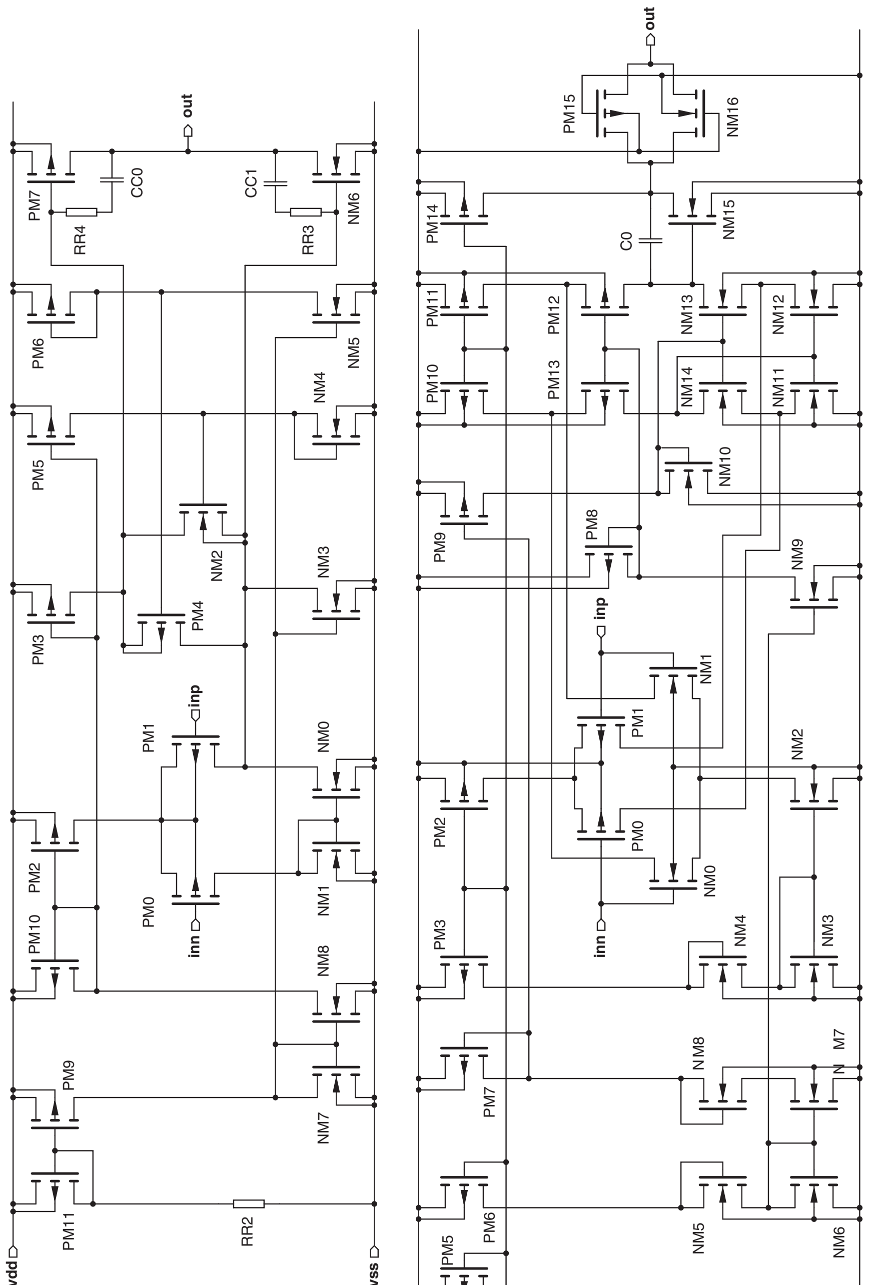

Fig. 2. Topologies of the damp2 (top) and lfbuf (bottom) test circuits.

(NM3, NM5, NM7, NM8), NM4, NM6, (PM0, PM1), (PM2, PM3, PM5, PM10), (PM9, PM11), PM4, PM6, and PM7. Additionally, three resistors (RR2, RR3, and RR4) and two capacitors (CC0 and CC1) are also subject to optimization contributing a total of five parameters. Together with the MOS transistors, the total parameter count is $22 + 5 = 27$.

The MOS transistor widths and lengths are optimized in the range from 0.3 to 50 μm and from 0.18 to 5 μm, respectively. Capacitors and resistors are optimized in the range from 10 to 20 pF and from 10 to 100 kΩ. The step for MOS width and length, capacitance, and resistance is 0.01 μm, 1 fF, and 1 Ω, respectively.

The amplifier load is set to 10 pF capacitance in parallel with 10 kΩ resistance. A total of five design corners are accounted for in the process of circuit evaluation.

- typical MOS model, 27°C, 1.8 V supply voltage,
- fast MOS model, 50°C, 2.0 V supply voltage,
- slow MOS model, 0°C, 1.6 V supply voltage,
- fast MOS model, 0°C, 1.6 V supply voltage, and
- slow MOS model, 50°C, 2.0 V supply voltage.

Four analyses are performed for every circuit in every corner — an operating point analysis, a DC sweep analysis, an AC analysis, and a transient analysis. The values of performance measures in their respective worst corner points (Table 2) are obtained from the results of these analyses. The CF value is obtained by adding up the CF contributions of these measurement values across all corners. The CF contribution of a measurement is calculated using the following formula

$$n_i = \begin{cases} M_{\mathrm{W}} \ \mathrm{ramp}\left(\dfrac{M - M_{\mathrm{G}}}{M_{\mathrm{G}}}\right) & \text{for minimizing } M\,, \\ M_{\mathrm{W}} \ \mathrm{ramp}\left(\dfrac{M_{\mathrm{G}} - M}{M_{\mathrm{G}}}\right) & \text{for maximizing } M\,, \end{cases} \tag{12}$$

where ramp(x) is the unit ramp function and M is the measurement value. The values of goal (M_{G}) and weight (M_{W}) are listed in Table 2.

The number of optimization parameters (n), the number of SPICE analyses (A) needed for a single CF evaluation, and the number of MOS transistors (M) in the circuit are listed in Table 1. The last two properties are not specified for the analytical test cases. The damp2c and lfbufc circuits are the same as damp2 and lfbuf, except that the optimization is performed across several corner points. This increases the number of required analyses per CF evaluation and changes the shape of the CF due to the inclusion of multiple corner points.

4. Results

We chose the parameters of the combined algorithm empirically with limited numerical trials: $\mu_1 = 10$, $\mu_2 = 100$, $k_{\mathrm{s}} = 0.22$, and $\overline{d}_{\mathrm{s}} = 1.5$. Every test problem was run with the simplex algorithm and with the combined algorithm. Table 1 lists

Table 1. Performance of the combined algorithm versus the performance of the modified simplex algorithm.

		Simplex		Combined	
Test problem	$n/A/M$	N	f_{final}	N	f_{final}
lin*	10/—/—	5066	− 999.993	888	− 1000.000
sq1	10/—/—	2130	1.44e−6	1028	3.70e−3
sq2*	10/—/—	6871	− 24999.1	363	− 25000.0
nand	3/9/4	295	166.644	172	166.735
delay*	12/2/6	1334	13284.6	1215	12975.1
ddamp	14/44/18	1543	104.426	1453	91.2954
damp1*	15/31/13	1716	9.58849	1365	9.58514
damp2*	27/4/20	5719	2.64345	3369	2.05055
damp2c	27/20/20	4732	7.36757	2314	7.02317
lfbuf*	36/4/32	9152	0.763508	3786	0.707504
lfbufc	36/20/32	5759	4.18654	3028	4.01531

n, A, M, N, and f_{final} are the number of optimization parameters, the number of analyses per candidate circuit, the number of MOS transistors in the circuit, the number of evaluated candidate circuits, and the final CF value, respectively.

Table 2. Resulting circuit performance for the damp2c test case.

Performance measure	Target	M_{G}	M_{W}	Simplex	Combined
Cost	min	—	—	7.37	7.02
Area (μm^2)	min	10000	1	7310	7160
Supply current (mA)	min	1	3	0.992	0.984
AC gain (dB)	max	70	5	63.6	64.2
Unity gain bandwidth (MHz)	max	5	3	3.6	4.11
Bandwidth (Hz)	max	500	1	2160	2130
Phase margin (°)	max	60	5	41.1	40.4
Gain margin (dB)	max	10	5	14.5	13.5
Swing (V)	max	1.6	5	0.893	0.896
DC gain (dB)	max	60	5	43.3	43.4
Settling time (ns)	min	300	1	275	333
Overshoot (%)	min	1.0	1	0.765	0.857
Slew rate (V/μs)	max	5	1	3.4	3.8
Rise time (ns)	min	200	1	141	128
Fall time (ns)	min	200	1	315	301

M_{G} is the measurement goal and M_{W} is the measurement weight. The last two columns list the worst measurement values across all corner points for the simplex and the combined algorithm.

the number of CF evaluations (N) and the final CF value (f_{final}) reached by both the algorithms. The time needed by the algorithm to reach the solution is directly proportional to the number of CF evaluations.

On all of the test problems the combined algorithm terminates after fewer CF evaluations than the simplex algorithm. On one problem (sq1) the final CF value obtained by the combined algorithm is significantly worse than the one obtained by the simplex algorithm. This happened due to the inappropriately chosen trust-region termination condition. By changing γ_{t} to $1.4 \cdot 10^{-5}$ the combined algorithm takes 2120 CF evaluations to reach the final CF value of $1.65 \cdot 10^{-6}$ which is almost the same as the result obtained by the simplex algorithm. By further decreasing γ_t the combined algorithm finds a better solution, but requires more CF evaluations than the simplex algorithm. The good performance of the simplex algorithm can be explained by the single-minimum located in the center of the search space. A simplex algorithm with a random initial simplex quickly finds such solutions by repeated shrinking.

For the nand test problem the final CF value obtained by the combined algorithm is slightly worse than the one obtained by the simplex algorithm. The two solutions (compared in terms of the optimization parameters) are close in the search space. The values of the three optimization parameters found by the simplex algorithm are 2.16, 0.68, and 1.38 μm. The combined algorithm resulted in 2.12, 0.65, and 1.32 μm. The combined algorithm terminated prematurely due to numerical noise in the CF. Nevertheless the combined algorithm found only a slightly worse solution with significantly fewer CF evaluations. For all other test problems, the CF value obtained by the combined algorithm was better.

In Table 1, asterisks mark the test problems where the switch to the trust-region algorithm occurred due to the rate of simplex size change. This occurs in about one-half of the problems, which indicates that the use of both the simplex size and the rate of simplex size change in the switch condition is justified.

In seven out of the eight IC design test problems the combined algorithm outperformed the simplex algorithm both in terms of the final CF value and in terms of the number of CF evaluations. This provides a fairly good reason to believe that the combined algorithm generally outperforms the simplex algorithm on circuit optimization problems. It is reasonable to expect that replacing the simplex algorithm with the combined algorithm will generally improve the final results of circuit optimization runs and decrease the number of CF evaluations for up to 50%.

Table 2 compares the performance measures of the damp2c circuit optimized with the simplex algorithm and the combined algorithm. Of the 14 performance measures, the circuit that resulted from the combined algorithm run has nine performance measures that are better. The comparison of the device parameter values (Table 3) reveals that the circuit obtained with the combined algorithm is very close to the circuit obtained by the simplex algorithm (of 27 parameters, 15 are almost identical and four are in the same order of magnitude). The first one, however, was obtained in less than half the time.

5. Conclusion

The ever-increasing pressure to shorten the time-to-market exerted by the IC industry is fueling the development of design automation methods and software. Automated selection of element parameter values for a predefined topology (automated circuit sizing) is the simplest of the techniques that have the potential to significantly shorten the time-to-market. At the core of every software for automated circuit sizing there is always a parametric optimization algorithm.

Table 3. The comparison of resulting circuits for the damp2c test problem.

	NM0 W	NM0 L	NM2 W	NM2 L	NM3 W	NM3 L	NM4 W	NM4 L
Simplex	26.1	0.33	33.0	2.52	0.84	2.62	34.4	1.58
Combined	26.1	0.33	34.4	2.55	0.82	2.67	34.9	1.47

	NM6 W	NM6 L	PM0 W	PM0 L	PM2 W	PM2 L	PM9 W	PM9 L
Simplex	1.17	2.45	5.12	0.37	37.2	2.65	20.4	1.37
Combined	26.1	0.33	34.4	2.55	0.82	2.67	34.9	1.47

	PM4 W	PM4 L	PM6 W	PM6 L	PM7 W	PM7 L	CC0	CC1
Simplex	48.1	4.68	23.2	0.83	6.51	4.97	5.20	3.89
Combined	26.1	0.33	34.4	2.55	0.82	2.67	5.08	3.40

	RR2	RR3	RR4
Simplex	31.8	50.9	8.44
Combined	32.1	45.8	9.16

The units for width and length, capacitance, and resistance are μm, pF, and kΩ, respectively.

One of the methods that performed remarkably well on circuit optimization problems is the Box simplex algorithm. Despite its good performance, it still has two major shortcomings: slow convergence in the neighborhood of a minimum and lack of convergence theory. Combining this algorithm with a mathematically well-established algorithm from the family of trust-region algorithms, one can expect to overcome these shortcomings.

We implemented the combined algorithm in the SPICE OPUS circuit simulator and optimizer. We test it on a set of 11 test problems. On most of them (9), the combined algorithm outperformed the Box simplex algorithm both in terms of the number of CF evaluations (by up to 50%) and the final CF value. This makes us believe that replacing the Box simplex algorithm with the combined algorithm will generally benefit the automation of the circuit-sizing process.

As circuit optimization is a lengthly task, parallelization of the combined algorithm has the potential to even further speed up the search for a circuit that satisfies the design requirements. Both simplex[19,20] and trust-region[21] algorithms were already a subject of research in the field of algorithm parallelization. It is also possible to distribute evaluations of the same circuit in different operating conditions (corners) on multiple computers.[22] Therefore, we can reasonably expect that parallelizing the combined algorithm will be a fairly simple task.

Acknowledgments

The research was co-funded by the Ministry of Education, Science, and Sport (Ministrstvo za Šolstvo, Znanost in Šport) of the Republic of Slovenia through the program P2-0246 Algorithms and optimization methods in telecommunications.

References

1. G. Gielen and R. Rutenbar, Computer-aided design of analog and mixed-signal integrated circuits, *Proc. IEEE* **88** (2000) 1825–1854.
2. A. Bűrmen, D. Strle, F. Bratkovič, J. Puhan, I. Fajfar and T. Tuma, Penalty function approach to robust analog IC design, *Informacije MIDEM — J. Microelectron. Electron. Comp. Mater.* **32** (2002) 149–156.
3. 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-Int. J. Electron. Commun.* **57** (2003) 47–56.
4. A. Bűrmen, J. Puhan and T. Tuma, Defining cost functions for robust IC design and optimization, *IEEE Designer's Forum: Design, Automation, and Test in Europe — Conf. Exhibition*, eds. D. Sciuto and D. Verkest, Munich, Germany (2003), pp. 196–201.
5. J. Nelder and R. Mead, A simplex method for function minimization, *Comput. J.* **7** (1965) 308–313.
6. M. Box, A new method of constrained optimization and comparison with other methods, *Comput. J.* **7** (1965) 42–54.
7. J. Puhan, T. Tuma and I. Fajfar, Optimisation methods in SPICE: A comparison, *ECCTD '99: Proc.*, eds. C. Beccari *et al.*, Stresa, Italy (1999), pp. 1279–1282.
8. J. Puhan and T. Tuma, Optimisation of analog circuits with spice 3f4, *Proc. 1997 ECCTD*, Budapest, Hungary (1997), pp. 2/177–180.
9. V. Torczon, Multi-directional search: A direct search algorithm for parallel machines, PhD. thesis, Rice University (1989).
10. M. H. Wright, Direct search methods: Once scorned, now respectable, *Numerical Analysis, Pitman Research Notes in Mathematics*, eds. D. F. Griffiths and G. A. Watson (Addison Wesley Longman Limited, 1995), pp. 191–208.
11. K. I. M. McKinnon, Convergence of the Nelder–Mead simplex method to a nonstationary point, *SIAM J. Optimization* **9** (1999) 148–158.
12. A. Conn, K. Scheinberg and P. Toint, Recent progress in unconstrained nonlinear optimization without derivatives, *Math. Program.* **79** (1997) 397–414.
13. A. Conn, K. Scheinberg and P. Toint, On the convergence of derivative-free methods for unconstrained optimization, *Approximation Theory and Optimization: Tributes to M. J. D. Powell*, eds. M. Buhmann and A. Iserles (Cambridge University Press, Cambridge, 1997), pp. 83–108.
14. A. Conn, N. Gould and P. Toint, *Trust-Region Methods* (SIAM, USA, 2000).
15. A. Bűrmen, J. Puhan and T. Tuma, Grid restrained Nelder–Mead algorithm, *Comput. Optimization Appl.* **34** (2006) 359–375.
16. A. Nussdorfer, A. Bűrmen, J. Puhan and T. Tuma, On cost function properties in analog circuit optimization, *Informacije MIDEM — J. Microelectron. Electron. Comp. Mater.* **34** (2004) 95–101.
17. J. Lagarias, J. Reeds, M. Wright and P. Wright, Convergence of the Nelder–Mead simplex method in low dimensions, *SIAM J. Optimization* **9** (1998) 112–147.
18. A. Bűrmen, J. Puhan and T. Tuma, Robust design and optimization of operating amplifiers, *IEEE Int. Conf. Industrial Technology*, Maribor, Slovenia (2003), pp. 745–750.
19. J. Dennis and V. Torczon, Parallel implementations of the Nelder–Mead simplex algorithm for unconstrained optimization, *Proc. SPIE — The Int. Soc. Optical Eng.* **880** (1988) 187–191.
20. L. Coetzee and E. Botha, The parallel downhill simplex algorithm for unconstrained optimization, *Concurrency: Practice and Experience* **10** (1998) 121–137.

21. P. Hough and J. Meza, A class of trust-region methods for parallel optimization, *SIAM J. Optimization* **13** (2002) 264–282.
22. A. Bűrmen, J. Puhan and T. Tuma, Parallel sizing of robust analog ICs, *Informacije MIDEM — J. Microelectron. Electron. Comp. Mater.* **34** (2004) 88–94.