OPTIMIZATION OF ELECTRONIC CIRCUITS

E.J.W. TER MATEN, T.G.A. HEIJMEN

NXP Semiconductors, Research, DMS - Physical Design Methods,
Hich Tech Campus 48, 5656 AE Eindhoven, The Netherlands
E-mail: {Jan.ter.Maten,Tino.Heijmen}@NXP.com

C. LIN and A. EL GUENNOUNI

Magma Design Automation,
TUE Campus, Den Dolech 2, Dommel Building Z-Wing 8, 5612 AZ Eindhoven, The Netherlands
E-mail: {Achie,Ahmed}@Magma-DA.com

Keywords: Global Optimization; Derivative Free; Robust Design; Augmented Lagrangian.

1. Introduction

Several types of parameters $\mathbf{p} = (\mathbf{x}, \mathbf{s}, \theta)$ influence the behaviour of electronic circuits and have to be taken into account when optimizing appropriate performance functions $f(\mathbf{p})$: design parameters $\mathbf{x}$, manufacturing process parameters $\mathbf{s}$, and operating parameters θ.

For optimizing one wants to minimise a performance function $f(\mathbf{p})$ while also several constraints have to be satisfied. The performance function $f(\mathbf{p})$ and the constraint functions $c(\mathbf{p})$ can be costly to evaluate and are subject to noise (for instance due to numerical integration effects). For both, the dependency on $\mathbf{p}$ can be highly nonlinear. In circuit simulation, sensitivities of $f(\mathbf{p})$ and $c(\mathbf{p})$ with respect to $\mathbf{p}$ are not always provided (several model libraries do not yet support the calculation of sensitivities). When the number of parameters increases adjoint sensitivity methods become of interest. For transient integration of linear circuits this is described in [3]. Recently, in [9] a more general procedure is described that also applies to nonlinear circuits and retains efficiency by exploiting (nonlinear) techniques from Model Order Reduction.

In this paper we will describe our in-house developed method for optimization and our experiences with it. We restrict ourselves to so-called derivative free methods, so we will not require sensitivities of $f(\mathbf{p})$ and $c(\mathbf{p})$ with respect to $\mathbf{p}$ from the circuit simulator. Also some new directions for further research will be described.

2. Constrained optimization by augmented Lagrangian

In this section we restrict our parameters $\mathbf{p}$ to the design variables $\mathbf{x}$, which can be geometrical quantities like transistor width W and length L. The designer can adjust them during the process of optimizing the performance of an actual design. The performance functions $f(\mathbf{x})$ typically consist of one or more circuit characteristics that the designer would like to maximize or minimize. Examples are: maximization of gain or bandwidth, and mimization of power dissipation or area. The constraints $c(\mathbf{p})$ are also related to circuit performance but these functions have an explicit target. Examples are: bandwidth $\geq$ 800MHz, and 49% $<$ duty cyle $<$ 51%. Formulating a proper set of performance functions and constraints is a non-trivial task. In practice, this requires trial-and-error and a fair amount of tuning. Obtaining the value of a performance function or constraint is done via one or more circuit analyses of the same or different types, e.g. DC, AC, Transient, Noise. Typically, the time required to perform an analysis is a limiting factor in the overall optimization approach.

The search for the optimal values of the optimization variables (OVs) $\mathbf{x}$ can be formulated as a nonlinear constrained optimization problem in n variables with m constraints,

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}), \qquad \mathbf{x} = (x_1, x_2, \ldots, x_n), \\ \text{subject to} \quad & c_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m, \\ & a_j \leq x_j \leq b_j, \, j = 1, \ldots, n, \end{aligned} \tag{1}$$

where x_i denotes the i-th OV. With $\mathbf{x}^\star$ we denote the point where the minimum occurs. The values of the objective function $f(\mathbf{x})$ and the constraining functions $c_i(\mathbf{x})$ are obtained from circuit simulation. The performance and stability of the optimization algorithm is affected by the *scaling* of the OVs, of $f(\mathbf{x})$ and of the $c_i(\mathbf{x})$ [7].

The Nelder-Mead algorithm can be used to minimize

$$\Phi(\mathbf{x}) = f(\mathbf{x}) + \mu \sum_{i=1}^{m} < c_i(\mathbf{x}) >^2 \tag{2}$$

where $< \alpha >= \max(\alpha, 0)$. If the minimum is taken at $\mathbf{x}_\mu$, one has $\lim_{\mu \to \infty} \mathbf{x}_\mu = \mathbf{x}^\star$, which means that the minimization of (2) becomes more and more ill-conditioned. Apart from that, the Nelder-Mead algorithm, which is simple to program, is not that easy to analyse [21]. A similar conditioning problem happens when dealing with logarithmic barrier functions that provide an impassible barrier at the boundary of the feasible region [21].

Recently pattern search methods have been studied [11,12]. The most well-known method of this class is the method of Hooke-Jeeves [8]. The Nelder-Mead method

does not belong to that class (it can also beat pattern search methods). The pattern search methods are nice in the sense that they prove rather weak conditions under which the methods do converge, f.i. for the unconstrained problem $f \in C^1$.
By introducing slack variables $s_i \geq 0$, the augmented Lagrangian penalty function can be written as [18],

$$\Phi_{\mathrm{ALAG,s}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{s}) = \underbrace{f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i [c_i(\mathbf{x}) + s_i]}_{\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})} + \sum_{i=1}^{m} \mu_i [c_i(\mathbf{x}) + s_i]^2, \quad (3)$$

$$= f(\mathbf{x}) + \sum_{i=1}^{m} \{\mu_i [(c_i(\mathbf{x}) - \frac{\lambda_i}{2\mu_i}) + s_i]^2 - \frac{\lambda_i^2}{4\mu_i}\} \quad (4)$$

in which $\mathcal{L}$ is the standard Lagrangian. The parameters λ_i and μ_i are Lagrange multipliers and penalty factors, respectively. Pattern search methods for (3) converge when $f, c_i \in C^2$ [13]. Minimization (4) over the slack variables s_i at optimal value $s_i^\star = \max\left[-c_i(\mathbf{x}) + \frac{\lambda_i}{2\mu_i}, 0\right]$ yields the simplified merit function that is used in [7],

$$\Phi_{\mathrm{ALAG}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \Psi(c_i(\mathbf{x}), \frac{\lambda_i}{2\mu_i}) + \sum_{i=1}^{m} \mu_i \Psi^2(c_i(\mathbf{x}), \frac{\lambda_i}{2\mu_i}) \quad (5)$$

$$= f(\mathbf{x}) + \sum_{i=1}^{m} \mu_i \tilde{\Psi}(c_i(\mathbf{x}), \frac{\lambda_i}{2\mu_i}), \quad (6)$$

where

$$\Psi(y, \xi) = \max[y, \xi], \quad \text{and} \quad \tilde{\Psi}(y, \xi) = \max[y - \xi, 0]^2 - \xi^2$$

The optimal point $(\mathbf{x}^\star, \boldsymbol{\lambda}^\star)$ becomes a stationary point of $\mathcal{L}$ and satisfies the Karush-Kuhn-Tucker (KKT) conditions,

$$\nabla_x \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star) = 0, \quad (7)$$

$$c_i(\mathbf{x}^\star) \leq 0, \quad \lambda_i^\star \leq 0, \quad \lambda_i^\star c_i(\mathbf{x}^\star) = 0, \quad i = 1, \ldots, m.$$

Note that in (6) $\forall \mu_i \neq 0$ $\Psi(c_i(\mathbf{x}^\star), \frac{\lambda_i^\star}{2\mu_i}) = 0$ and $\tilde{\Psi}(c_i(\mathbf{x}^\star), \frac{\lambda_i^\star}{2\mu_i}) = 0$. The KKT conditions imply that the projected gradient $\mathcal{P}(\nabla_x \Phi_{\mathrm{ALAG}}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu})) = 0$, where

$$[\mathcal{P}(\nabla_x F(\mathbf{x}))]_j = \begin{cases} [\nabla_x F(\mathbf{x})]_j & \text{if } a_j < x_j < b_j \\ \min[\,[\nabla_x F(\mathbf{x})]_j, 0\,] & \text{if } a_j = x_j \\ \max[\,[\nabla_x F(\mathbf{x})]_j, 0\,] & \text{if } b_j = x_j \end{cases} \quad (8)$$

The reduction in dimension has been paid back by loss of smoothness due to the 'max' function. However, any algorithm can check for $c_i = \lambda_i/(2\mu_i)$ and in general this does not occur at a KKT point [4]. Thus, when locally enough started,

pattern search methods will also converge for (6).
A conjecture is that under mild conditions there are $|\mu_i| < \infty$ such that $\Phi_{\mathrm{ALAG}}(.,.,\boldsymbol{\mu})$ has a local minimum in $(\mathbf{x}^\star, \boldsymbol{\lambda}^\star)$. Also the location becomes independent of μ_i (when large enough). This happens for instance when $\forall i$ $c_i(\mathbf{x}) \equiv \tilde{c}_i(x_i)$ [a simple 1-D example is $f(x) = x^2$, $c(x) = x - 1$, for which $(\mathbf{x}^\star, \boldsymbol{\lambda}^\star) = (1, -2)$]. This indicates the better conditioning of the problem (6) when compared to (2).
For updating $\boldsymbol{\lambda}$ we observe the equalities [cf also Primal-Dual methods]:

$$\begin{aligned}\nabla_x \Phi_{\mathrm{ALAG,s}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{s})\Big|_{\mathbf{s}=\mathbf{s}^\star} &= \nabla_x \Phi_{\mathrm{ALAG}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= \nabla_x \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}') \quad \text{for} \quad \lambda_i' = \lambda_i - \max[2\mu_i c_i(\mathbf{x}), \lambda_i].\end{aligned}$$

The basic method used to solve the optimization problem is the Method of Multipliers (Algorithm 2.1). In step 3 a trust region approach is applied on a re-

Algorithm 2.1 Method of Multipliers [7]

1: Start: $k = 1$, $x_i = x_i^{(0)}$, $\lambda_i = \lambda_i^{(0)} = 0$, $\mu_i = \mu_i^{(0)}$
2: Set $\Phi(\mathbf{x}) = \Phi_{\mathrm{RSM}}(\mathbf{x}, \boldsymbol{\lambda}^{(k-1)}, \boldsymbol{\mu}^{(k-1)}; \mathbf{x}^{(k-1)})$
3: Minimize: $\mathbf{x}^{(k)} = \mathrm{argmin}_{\mathbf{x}} \Phi(\mathbf{x})$
4: If (update μ_i) $\mu_i^{(k)} = 10 * \mu_i^{(k-1)}$ (but $< \mu_{\max}$!)
5: Else update $\lambda_i^{(k)} = \lambda_i^{(k-1)} - \max[2\mu_i^{(k-1)} c_i(\mathbf{x}^{(k-1)}), \lambda_i^{(k-1)}]$
6: Endif
7: Test for optimality
8: If not optimal set $k = k + 1$ and goto 2.

sponse surface model around $\mathbf{x}^{(k-1)}$ to solve the minimization problem (for details see [7]). In step 4 we keep $\mu_i^{(k)} < \mu_{\max}$. This is based on the basic observation made above: in the end we have trust that only the $\lambda_i^{(k)}$ should be able to do the work. Note that in step 5 linear convergence of the $\lambda_i^{(k)}$ can be improved by applying vector extrapolation techniques. The overall method is derivative free (no gradients of f and of c_i are required). Also the method is not sensitive to noise on f and of c_i. Convergence of a related (but simpler, because no grid is used) algorithm is discussed in [18], assuming that $f, c_i \in C^2$ and that one assures $c_i(\mathbf{x}) \neq \lambda_i/(2\mu_i)$: error control is on $||\mathcal{P}(\nabla_x \Phi_{\mathrm{ALAG}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}))||$ and on $||\Psi(c_i(\mathbf{x}), \frac{\lambda_i}{2\mu_i})||$.
In our case [7] the whole approach is applied on a grid (i.e. all $\mathbf{x}^{(k)}$ are projected to the nearest grid point), that subsequently is refined, like in [5]. This prevents the algorithm from going too fast to a small scale (and then gets stuck in a local

minimum). However, in practice equally appreciated, is that it also allows to store and re-use expensive parts of $f(\mathbf{x})$ and of $c_i(\mathbf{x})$ during the re-building of the response surface model in the optimization process, when $\boldsymbol{\lambda}$ or $\boldsymbol{\mu}$ are updated, and also when refining the grid.

Our trust region error control is described in [7]. Because we only have approximative gradients from the response surface model approximation, for final convergence, error control is based on

$$|f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k-1)})| < \varepsilon_f(1 + |f(\mathbf{x}^{(k)})|), \tag{9}$$

$$||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|| < \varepsilon_x(1 + ||\mathbf{x}^{(k)}||) \tag{10}$$

The parameters $\varepsilon_f, \varepsilon_x$ are decreased when refining the grid. At initialization we apply a Uniform Design approach which is based on number theory. Actually this approach is problem independent. Here improvements can be expected by introducing additional techniques, like pattern search methods [11,12], that add more information from the problem itself to improve initial starts for the above method. Also Kriging techniques using the DACE regression model [2,10,15] can be used in this phase, which additionally offer a more global optimization aspect, when also applied later in the process. The regression model includes effects due to correlation. Based on values $y_i = y(\mathbf{x}^{(i)}) = \mu + \epsilon(\mathbf{x}^{(i)})$, in which the $\epsilon(x_k^{(i)}) \approx \mathrm{N}(0, \sigma^2)$ are normally distributed and have correlation matrix $R = \mathrm{Corr}[\epsilon(\mathbf{x}^{(i)}), \epsilon(\mathbf{x}^{(j)})] = \exp[-\sum_k \theta_k |x_k^{(i)} - x_k^{(j)}|^{p_k}]$, an exact interpolant $\hat{Y}(\mathbf{x})$ can be derived, together with an estimation for the variance $s^2(\mathbf{x})$ [clearly $s^2(\mathbf{x}^{(i)}) = 0$]. The EGO algorithms [10,20] maximize the expected improvement for $Y(\approx \mathrm{N}(\hat{Y}, s^2)$. However, experience shows that only a small reduction in the number of overall function evaluations was obtained when including this approach in the start-up phase of our simulator. In this case all parameters were involved. Note that the method becomes expensive when the number of parameters increase. It may make sense to limit the Kriging approach to the parameters for which the augmented Lagrangian varies only moderately.

3. Discrete optimization

Considerable interest has been shown for discrete optimization methods since continuous optimization methods were well established in the late 1980s. In general, discrete optimization methods are aimed at solving the so-called mixed-discrete nonlinear programming problem, where the term mixed-discrete indicates that both discrete and continuous design variables are present.

Current discrete optimization methods can be classified as either deterministic or probabilistic [1]. Probabilistic methods have been applied to solve engineering optimization problems for a long time. The most well known probabilistic methods

for both continuous and discrete optimization are Genetic Algorithms [17] and Simulated Annealing [22]. One major advantage of probabilistic methods is their ability to directly deal with discrete design variables. However, these methods are extremely expensive (because of a large number of function evaluations), and thus impractical for the optimization of electronic circuits.
Various approaches have been developed to apply deterministic methods for discrete and mixed-discrete optimization problems [6,19]. The simplest and least expensive method for obtaining a discrete solution is by rounding the continuous solution to the value of the closest discrete values. However, this rounding process can easily result in stagnation and convergence problems since extra local minimas are introduced. Therefore, the applicability of the rounding methods may be limited.
The branch and bound method is probably the best known and most frequently used discrete optimization method. The method is based on the sequential analysis of a discrete tree for each variable, the computational cost grows exponentially with the number of design variables. Therefore, the method is also extremely expensive.
In our in-house developed method, a dynamic rounding approach was adopted. Design variables are rounded to the value of the closest discrete values at each evaluation step. The approach has the advantage that no extra local minimums are generated. Stagnation problems are avoided by the use of a component-wise trust region method when solving the quadratic problems generated by the algorithm. The component-wise trust region method allows us to select the space where the new design variable candidate should be stamped by forcing the trust radius of each discrete design variable to be larger than a desired value.

4. Toward Robust Design

The additional types of parameters, the manufacturing process parameters $\mathbf{s}$ and operating parameters $\boldsymbol{\theta}$, introduce additional requests.

- The first ones, $\mathbf{s}$, reflect statistical process fluctuations, like T_{ox} (oxide thickness), and V_{th0} (threshold voltage), substrate doping, characterized by their distribution functions.
- The operating parameters $\boldsymbol{\theta}$ include supply voltage V_{sup} and temperature T. Here additional constraints are found that require $c(\mathbf{x}, \boldsymbol{\theta}) \leq 0$ should hold for all $\boldsymbol{\theta}$ within some interval.

The *operating parameters* $\boldsymbol{\theta}$ imply simple additional inequality constraints that fit the already defined framework. The dependency with respect to $\boldsymbol{\theta}$ ususally is quite smoothly. An interesting test function to consider the effect of operating

parameters is

$$c(x,\theta) = (\theta-2)^2\frac{x\ \sin(x)}{2}, \quad \text{with}\ \ x\in[0,6],\ \ \theta\in[0,5]. \tag{11}$$

When starting at $(x=5,\theta=0)$ a local maximum at $(x=6,\theta=0)$ is easily found. An even higher value of c may be preferred here as a more robust result with respect to θ when one likes to satisfy $c(x,\theta)\geq 0$. Here Kriging techniques were of help to improve the result. Note that this is a particular form of robust design. The problem can be simplified by restricting θ to a discrete set $\theta_i\in\Theta$. In an outer loop the bounds on the constraints can be adaptively updated. New values of θ_i can be selected based on the Kriging approach to determine next interpolation points.

Most *statistical parameters* $\mathbf{s}$ appear as transistor model parameters and cannot be modified by the designer if a fully qualified production process is assumed. However, production variation must be taken into account when designing a circuit. For robust design one can think to optimize a combination of f and of its (higher) derivatives (H^1-norm optimization, say), which requires the need to evaluate these derivatives (sensitivities). Also spreads of several σ's have to be taken into account. The occurence of process parameters $\mathbf{s}$ also makes the functions f and c_i stochastic. So one needs to consider the effect of the transformed probability density function while also correlations have to be taken into account. With each $s_j\approx N(\mu_j,\sigma_j)$, the common normal density function $\phi(\mathbf{s})=C\exp[-\frac{1}{2}\beta^2(\mathbf{s})]$ involves a distance function $\beta^2(\mathbf{s})=(\mathbf{s}-\boldsymbol{\mu})^T\Sigma_s^{-1}(\mathbf{s}-\boldsymbol{\mu})$, which can be used to measure the distance from the nominal value. If $\mathcal{A}_s=\{\mathbf{s}\in\Omega_s|\mathbf{c}(\mathbf{s})\in\Omega_c\}$, where Ω_c indicates the acceptance region for the constraint function values, the worst-case point s_i^{wc} can be defined as

$$s_i^{wc} = \mathrm{argmax}_{s\in\partial\mathcal{A}_{c_i}}\,\phi(s) \quad = \quad \mathrm{argmin}_{s\in\partial\mathcal{A}_{c_i}}\,\beta^2(s)$$

In the worst-case analysis in [16] first linearization with respect to θ is done, and a θ_{WC} is derived. Next, this value is used in the linearization with respect to x, and a x_{WC} is determined. This process is iteratively repeated in a Gauss-Seidel-like manner. For several problems this algorithm gives reasonable results and yield estimations (as is confirmed by our own experiments). However, the algorithm appears not to be robust on the above constraint function c, which gives way for further research. Also, because of linearizations, normality of distributions is an essential ingredient.

As IC technologies scale down to finer feature sizes, it becomes increasingly difficult to control the relative process variations, especially when introducing new technologies. Hence large variations can be observed. Furthermore, these variations are different for each chip fabrication location. Depending on the physical

design properties of the circuit, the designer can introduce additional correlations to the given distributions, targeting improved circuit performance. Note that with the introduction of the $\mathbf{s}$ parameters, one must add additional constraints to the problem formulation. These are called yield constraints and their targets are expressed in terms of probabilistic quantities, typically the number of standard deviations from the mean value.

With fundamentally increasing variation magnitudes, non-normality of performance distributions must be taken into account. The APEX algorithm [14] addresses this point. A response surface model of f is built that is quadratic in the process variations. The method relies (yet) on the explicit calculation of the stochastic moments of f and on AWE-techniques (with their drawbacks) to efficiently approximate the transfer function. Here more robust model order reduction techniques may be applied.

A design that is capable of achieving the performance targets under the given constraints, across all so-called environmental corner conditions can be called robust. Robustness of a design starts to play an increasingly important role in modern circuit design, as design margins are decreasing and process variability is increasing. The use of traditional nominal-design optimization techniques is falling short because they tend to give seemingly good results, but these results collapse under process variations. Therefore, automation of the robust design challenge is in high demand.

5. A Robust Design Example

We now describe a simplified but realistic example of robust design. The purpose of this example is to get a better idea of the intricacies and challenges of optimization applied to circuit design.

The design is a high-frequency divide-by-two circuit fabricated in a state-of-the-art Complementary Metal-Oxide Semiconductor (CMOS) technology. The primary target of the optimization job is to increase the maximum input frequency at which the circuit still correctly divides by two, i.e. the output frequency is half the input frequency across all specified environmental conditions. Denoting with $\omega(v) = \text{freq}(v)$, the frequency of a (scalar) signal $v = v(\mathbf{x})$, that depends on parameters x_j, the problem can in terms of (1) be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & -\omega(v_{in}(\mathbf{x})), \\ \text{subject to} \quad & \omega(v_{out}(\mathbf{x}))/\omega(v_{in}(\mathbf{x})) - 0.5 \leq 0 \\ & 0.5 - \omega(v_{out}(\mathbf{x}))/\omega(v_{in}(\mathbf{x})) \leq 0 \\ & a_j \leq x_j \leq b_j, \qquad j = 1, \ldots, n, \end{aligned} \tag{12}$$

In this formulation, the x_j are coupled to the widths of selected transistors in the divider circuit, where typically the number of selected transistors is at least $2n$ to enforce so-called "matched pairs" of transistors. The lengths of the transistors are kept fixed at a reasonably small value guided by design experience.

The first practical issue that comes into play is how to "measure" the frequency $\omega(v)$. As the circuit exhibits a non-linear large-signal behavior, we need to use a transient analysis to measure its response. This calls for a robust measurement expression that *always* gives a meaningful result, even if the simulation does not converge. Devising robust and accurate simulator expressions is a non-trivial task that requires design and simulation tool knowledge.

Because we also want to have a *robust* solution that is inherently insensitive to process variations, we typically resort to using the well-known Monte Carlo (MC) analysis (placed as a loop around the former transient analysis). To limit the overall simulation and optimization time, a limit of ten MC trials per circuit evaluation with a fixed set of p_j values was enforced. Ten MC trials are not sufficient to guarantee so-called sign-off accuracy, but it appears to be sufficient for typical optimization purposes.

Another important aspect of applied optimization is the fact that the original objective(s) and constraints typically need to be transformed and tuned into a set of new equations that give more directional information to the optimization algorithm; if this process is done well, most algorithms will usually give better results in fewer iterations. This is also the case for this design example. We add two more constraints to prevent getting an output voltage swing that is too small to be properly detectable by the succeeding circuit. Design-specific knowledge and feedback from the circuit behavior during initial optimization runs is normally used to refine the optimization setup further.

The complete optimization problem is:

$$\begin{aligned}
\text{minimize} \quad & -\omega(v_{in}(\mathbf{x},\mathbf{s},\boldsymbol{\theta})), && (13)\\
\text{subject to} \quad & \omega(v_{out}(\mathbf{x},\mathbf{s},\boldsymbol{\theta}))/\omega(v_{in}(\mathbf{x},\mathbf{s},\boldsymbol{\theta})) \leq 0.5 && (14)\\
& \omega(v_{out}(\mathbf{x},\mathbf{s},\boldsymbol{\theta}))/\omega(v_{in}(\mathbf{x},\mathbf{s},\boldsymbol{\theta})) \geq 0.5 && (15)\\
& \min_t\left(v_{out}(\mathbf{x},\mathbf{s},\boldsymbol{\theta})(t)\right) \geq 0.9 && (16)\\
& \max_t\left(v_{out}(\mathbf{x},\mathbf{s},\boldsymbol{\theta})(t)\right) \leq 0.1 && (17)\\
& a_j \leq x_j \leq b_j, \quad j = 1,\ldots,n.
\end{aligned}$$

Solving this problem with the optimization algorithm described in this paper required about 200 iterations, depending on the stopping criterion and other tuning parameters. In each iteration the algorithm evaluates the circuit performance by running a 10-trial MC simulation in which a transient analysis is embedded. The

overall throughput time is the number of iterations times the time required for a full circuit evaluation; approximately 15 hours on a contemporary Linux system with a single CPU. From this information one can directly derive the average CPU time per 10-trial MC simulation: about 5 minutes. Note also that the time required for an MC simulation can become prohibitive quite quickly. This has motivated recent research in this area with the goal of finding alternative methods to efficiently simulate performance variability under process parameter variability, e.g. [14].
Table 1 shows the optimization results in tabular format. The information on the

Table 1. Optimization results (all values in μm)

Specification	Target	Initial	Optimized
(13)	maximize	10.5 GHz	17.1 GHz
(14) + (15)	= 0.5 V	[0.4999994, 0.5000011]	[0.4999991, 0.5000001]
(16)	> 0.9 V	[1.01077, 1.018599]	[0.9612433, 1.014029]
(17)	< 0.1 V	[-0.017398, -0.011710]	[-0.016031,-0.011626]
power dissipation		1.25 mW, [1.14, 1.41]	1.36 mW, [1.21, 1.51]

optimization variables x_j is given in Table 2. To illustrate the increase in robust-

Table 2. Optimization variables data (in μm)

OV	Initial	$[a_j, b_j]$	Final
x_1	2.5	[0.5, 7.5]	1.187
x_2	1.0	[0.5, 3.0]	1.195
x_3	4.0	[0.5, 12]	5.974
x_4	2.5	[0.5, 7.5]	1.638
x_5	4.0	[0.5, 12]	5.436
x_6	10.0	[0.5, 20]	14.13

ness of the divider circuit we show the results of a 100-trial MC simulation before and after optimization. Fig. 1(a) shows the unoptimized circuit performance at the maximized input frequency of 17.1GHz (from 10.5GHz). All graphs represent histograms; in typewriter order the graphs show bin count versus output frequency, safeguarded frequency ratio (with voltage thresholds), raw frequency ratio, minimum ”high” output voltage, maximum ”low” output voltage, and average power dissipation. Fig. 1(b) shows the performance of the divider circuit after optimization. From these plots we can draw the following conclusions:

- The performance of the divider circuit has been improved by increasing the maximum input frequency by more than 60%, with high robustness;

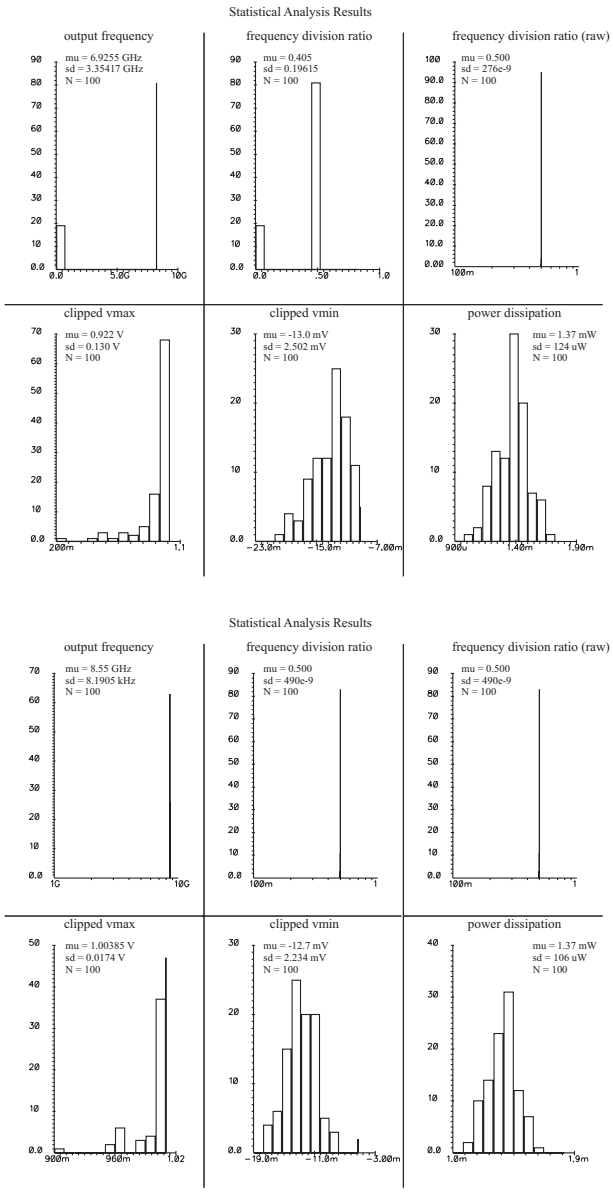

Fig. 1. (a) The divide-by-two circuit operating at 17.1GHz before optimization (top); and (b) the circuit operating at 17.1GHz after optimization (bottom).

- The optimized divider circuit can operate equally well at a maximized input frequency of 17.1GHz, with only a relatively small increase in power dissipation compared to 10.5GHz operation;
- Attaining increased robustness did not deteriorate any other metric.

References

1. J.S. Arora, M.W. Haug: *Methods for optimization of nonlinear problems with discrete variables: A review*, Struct. Opt., 8, pp. 69–85, 1994.
2. W.C.M. van Beers, J.P.C. Kleijnen: *Kriging interpolation in simulation: a survey*, In: R.G. Ingalls, M.D. Rossetti, J.S. Smith, B.A. Peters (Eds.): *Proc. of the 2004 Winter Simulation Conference*, Assoc. for Comput. Mach., New York, 2004.
3. A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, C. Visweswariah, C.W. Wu: *JiffyTune: cicuit optimization using time-domain sensitivities*, IEEE Trans. Computer-Aided Design, 17, pp. 1292–1309, 1998.
4. G. Di Pillo, L. Palagi: *Nonlinear programming: introduction, unconstrained and constrained optimization*, In: P.M. Pardalos, M.G.C. Resende (Eds): *Handbook of applied optimization*, Oxford Univ. Press, pp. 263–298, 2002.
5. C. Elster, A. Neumaier: *A grid algorithm for bound constrained optimization of noisy functions*, IMA J. Numer. Anal., Vol. 15, pp. 585–608, 1995.
6. A.A. Groenwold, N. Stander, J.A. Snyman: *A pseudo-discrete rounding method for structural optimization*, Struct. Opt., 11, pp. 218–227, 1996.
7. T. Heijmen, C. Lin, J. ter Maten, M. Kole: *Augmented Lagrangian algorithm for optimizing analog circuit design*, in: A. Buigis, R. Ciegis, A.D. Fit: *Progress in industrial mathematics at ECMI 2002*, Mathematics in Industry 5, Springer, pp. 179–184, 2003.
8. R. Hooke, T.A. Jeeves: *Direct search solution of numerical and statistical problems*, J. of the ACM, 8, pp. 212–229, 1961.
9. Z. Ilievski, H. Xu, A. Verhoeven, E.J.W. ter Maten, W.H.A. Schilders, R.M.M. Mattheij: *Adjoint Transient Sensitivity Analysis in Circuit Simulation*, Presented at SCEE-2006, Scientific Computing in Electrical Engineering, Sinaia, Romania, 2006.
10. D.R. Jones, M. Schonlau, W.J. Welch: *Efficient global optimization of expensive black-box functions*, J. of Global Optimization, Vol. 13, pp. 455–492, 1998.
11. T.G. Kolda, R.M. Lewis, V. Torcson: *Optimization by direct search: new perspectives on some classical and modern methods*, SIAM Review, Vol. 45-3, pp. 385–482, 2003.
12. T.G. Kolda, V. Torcson: *On the convergence of asynchronous parallel pattern search*, SIAM J. Optim, Vol. 14-4, pp. 939–964, 2004.
13. R.M. Lewis, V. Torczon: *A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, SIAM J. Optim., 12-4, pp. 1075–1089, 2002.
14. X. Li, J. Le, P. Gopalakrishnan, L.T. Pileggi: *Asymptotic probability extraction for non-normal distributions of circuit performance*, Proc. ICCAD 2004.
15. S.N. Lophaven, H.B. Nielsen, J. Søndergard: *DACE – A Matlab Kriging toolbox*, Techn. Report IMM 2002-12, TU Denmark, 2002.
16. M. Pronath, H. Graeb, K. Antreich: *On parametric test design for analog integrated circuits considering error in measurement and stimulus*, In: K. Antreich, R. Bulirsch,

A. Gilg, P. Rentrop (Eds.): *Modeling, simulation and optimization of integrated circuits*, In. Series of Numer. Maths., Vol. 146, pp. 283–301, 2003.
17. S. Rajeev, C.S. Krishnamoorthy: *Discrete optimization of structures using genetic algorithms* J. of Struct. Engineering, 118-5, pp. 1233–1250, 1992.
18. J.F. Rodriguez, J.E. Renaud, L.T. Watson: *Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data*, Structural Optim., Vol. 15, pp. 141–156, 1998.
19. E. Salajegheh, G.N. Vanderplaats: *Optimum design of trusses with discrete sizing and shape variables* Struct. Optim. 6, pp. 79–85, 1993.
20. M.J. Sasena: *Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations*, PhD-Thesis, Univ. of Michigan ,2002.
21. M.H. Wright: *What, if anything, is new in optimization*, In J.M. Ball, J.C.R. Hunt (Eds): *ICIAM 99 – Proc. of the fourth Int. Congress on Industrial and Applied Mathematics*, Oxford Univ. Press, pp. 259–270, 2000.
22. Y. Young Soon, K. Gi Hwa: *Stochastic search techniques for global optimization of structures* In C.K. Choi, H. Sugimoto, C.B. Yun (Eds): *Proceedings of the Korea-Japan Joint Seminar on Structural Optimization, Seoul, Korea*, pp. 87–97, 1992.

Acknowledgement: The first author acknowledges the EU Marie Curie RTN project COMSON, grant n. MRTN-CT-2005-019417.