Enhancing Analog Yield Optimization for Variation-aware Circuits Sizing

Ons Lahiouel, Mohamed H. Zaki and Sofiène Tahar
Dept. of Electrical and Computer Engineering, Concordia University, Montréal, Québec, Canada
Email: {lahiouel, mzaki, tahar@ece.concordia.ca}

***Abstract*—This paper presents a novel approach for improving automated analog yield optimization using a two step exploration strategy. First, a global optimization phase relies on a modified Lipschitizian optimization to sample the potential optimal sub-regions of the feasible design space. The search locates a design point near the optimal solution that is used as a starting point by a local optimization phase. The local search constructs linear interpolating surrogate models of the yield to explore the basin of convergence and to rapidly reach the global optimum. Experimental results show that our approach locates higher quality design points in terms of yield rate within less run time and without affecting the accuracy.**

I. Introduction

Novel and robust techniques for automated analog yield optimization are urgently needed to shorten time-to-market and avoid costly re-design iterations. However, identifying an optimized nominal design solution faces immense challenges. The difficulty mainly lies in locating the global optimum of a multivariate and nonlinear optimization problem. On the other hand, the increasingly large process variation has an adverse impact on the performances and renders the transistor-level yield estimation computationally expensive [5].

Existing techniques face problems in terms of sub-optimal solution, low flexibility and computational cost [6]. Indeed, deterministic optimizations highly depend on a near optimal initial design point and often reach local minimum [6]. In addition, stochastic optimizations suffer from a slow rate of convergence [5] and a limited coverage of the search space. Also, they necessitate multiple runs and are very sensitive to various search parameters. Corner-based and worst-case design methods can be efficient due to the limited number of simulations they require. However, while corner models can be either inaccurate or too pessimistic [6], the inherent error in the gradients computation required for worst case methods limits their usefulness [8]. This paper addresses the above issues by providing designers with an optimization strategy able to improve sizing robustness with minimum computational cost.

DIRECT (DIviding RECTangles) [3] is a modified Lipschitzian optimization [2] approach that eliminates the need to specify a Lipschitz constant. Its search strategy implies a high guarantee on the convergence to the global optimum. In fact, the method was created to solve difficult global optimization problems with bound constraints. Thus, it has received a great attention from the optimization community to solve modern large-scale, multidisciplinary engineering problems [7]. The method requires no knowledge about the starting point or the objective function gradient. Instead, it samples points in the search domain, and relies on the iteration history to decide about the next potential sample location.

In this paper, we propose an approach to enhance analog yield optimization for variation-aware circuits sizing, which incorporates Lipshitizian optimization and local Radial Basis Function (RBF) model-based search methods. Compared with existing techniques: (1) it is able to minimize the risk of missing potentially optimal design points; (2) it does not require multiple runs and only a few parameters need to be set; and (3) it minimizes the number of yield evaluations.

The rest of the paper is organized as follows: Section II details our yield optimization methodology. In Section III, we provide experimental results for a folded cascode amplifier. Finally, Section IV concludes the paper.

II. Proposed Methodology

An overview of our proposed framework for variation-aware circuits sizing is shown in Figure 1. We assume a continuous set of interval-valued sizing solutions that characterizes the feasible design space D_0. Any design point $x \in D_0$ is guaranteed to satisfy the specification in nominal condition [4]. The focus of this paper is to find a design point $x^* \in D_0$ that maximizes the yield. First, a global search step uses a parallelized Lipshitizian algorithm that trades off between the computational cost and the solution optimality. This step partitions and samples the centers of potentially optimal subregions of D_0 and locates the area near the global optimum. Second, a local search mechanism is used to rapidly reach the optimal design point starting from the best solution computed by the global search. Besides, it employs previously evaluated points to locally model the objective function and drives the optimization. At each optimization iteration, Monte Carlo simulation in HSPICE is employed to estimate the yield rate.

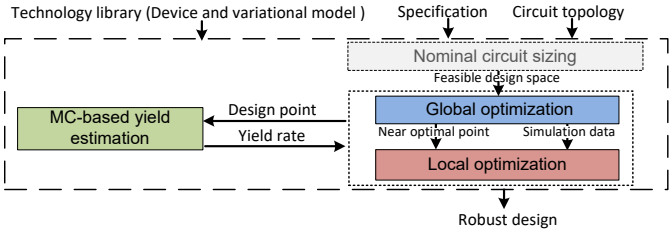

Figure 1. Framework for variation-aware circuits sizing

A. Parallel Global Optimization

The aim of the global search is to locate a design point $\overline{x^*}$ near to the optimal solution x^* of the problem:

$$\mathbf{x}^* \quad = \quad \max_{\mathbf{x}\in\mathbf{D_0}} \mathbf{g}(\mathbf{x}) = \min_{\mathbf{x}\in\mathbf{D_0}} \mathbf{f}(\mathbf{x}) \tag{1}$$

where $g : D_0 \longrightarrow R_+$ is a positive real-valued yield function and $f = -g$ is smooth and continuous in D_0.

Alg. 1 Parallel global optimization

Require: D_0: Design search space, f: Objective function
1: $D_0^{1\to S} \leftarrow$ Divide(D_0)
2: **for all** $ind = 1 \to S$ **do in parallel**
3: $\quad D_0^{ind} \leftarrow$ normalize(D_0^{ind})
4: $\quad$ Initialize: $\overline{x^*_{ind}} \leftarrow$ center(D_0^{ind}), $f^{min}_{ind} \leftarrow f(\overline{x^*_{ind}})$, $c_0 \leftarrow \overline{x^*_{ind}}$, $f(c_0) \leftarrow f^{min}_{ind}$, $\Gamma_{ind} \leftarrow \{c_0, f(c_0)\}$
5: $\quad$ **while** *stopping criteria* is unsatisfied **do**
6: $\quad\quad$ Identify S: all potential optimal hyperrectangles H_j
7: $\quad\quad$ **for all** $H_j \in S$ **do**
8: $\quad\quad\quad$ Identify M: the dimensions with max. side length d,
9: $\quad\quad\quad$ Evaluate **in parallel** $f(c_j \pm \alpha e_m)$, $m \in M$, $\alpha = d/3$
10: $\quad\quad\quad$ $\Gamma_{ind} \leftarrow \Gamma_{ind} \cup \{c_j \pm \alpha e_m, f(c_j \pm \alpha e_m)\}$
11: $\quad\quad\quad$ Update $\overline{x^*_{ind}}$, f^{min}_{ind}
12: $\quad\quad\quad$ Evaluate w_m and divide H_j according to w_m
13: $\quad\quad$ **end for**
14: $\quad$ **end while**
15: **end for**
16: **return** $\overline{x^*}$=min$_{\overline{x^*_{ind}}, f^{min}_{ind}}(f^{min}_{ind})$, $\Gamma = \bigcup_{ind=1}^{S} \Gamma_{ind}$

The approach is summarized in Algorithm 1. It is based on the DIRECT method [3] that is a variant of the Lipschitzian optimization. Hence, it is effective in finding the basin of convergence [2]. Also, it can operate in high-dimensional space as it uses an especially easy-to-manage partitioning of the search spaces into hyperrectangles, where only their center points are sampled.

In order to decrease the optimization running time and to conduct a refined exploration of the search space, we start by subdividing the yield optimization process into S subproblems that we invoke simultaneously (Line 1). Each subproblem is limited to a subregion of D_0 that is transformed into the unit hypercube (Line 3). The near optimal point is initialized by sampling the center of the search space (Line 4). Then, the set of potentially optimal hyperrectangles S is identified (Line 6). A hyperrectangle H_j is said to be potentially optimal if there exists a rate of change constant $\overline{K} > 0$ such that:

$$\begin{aligned} f(c_j) - \overline{K}d_j &\leq f(c_i) - \overline{K}d_i, \forall i \in I \\ f(c_j) - \overline{K}d_j &\leq f^{min}_{ind} - \gamma|f^{min}_{ind}| \end{aligned} \tag{2}$$

where I is the set of all indices of all hypererctangles, c_j is the center of H_j and d_j is the size of H_j defined as the distance from the center to the vertices of H_j [3]. The mathematical formula of d_j can be found in [3]. f^{min}_{ind} is the current best function value. The first inequality in Equation 2 expresses the decision to choose the hyperrectangle which promises the best improvement (i.e., decrease) in the objective function. It also ensures that as soon as a larger hyperrectangle with a lower function value at the center exists, the algorithm switches the search to that more promising (i.e., potential) region. The parameter γ in the second inequality guarantees that there is a sufficient decrease in the objective function. Once H_j is identified as potentially optimal, it is divided into smaller hyperrectangles (Lines 7 to 13). The divisions are performed only along its longest sides. It starts by determining the set M of all dimensions of maximal length (Line 8). Then, the function f is evaluated in parallel at $c_j \pm \alpha e_m$, where α is one-third the maximum side-length, and e_m, $m \in M$ is the mth unit vector (i.e., a vector with a one in the mth position and zeros elsewhere) (Line 9). The first division is performed perpendicular to the side with the lowest w_m, where:

$$w_m \quad = \quad min\{f(c_j + \alpha e_m), f(c_j - \alpha e_m)\}, \ m \in M \tag{3}$$

The new hyperrectangle that has center c_j is divided perpendicular to the direction of the second lowest w_m. The process is repeated until H_j is divided in all directions $m \in M$. The subdivision ensures that previous function evaluations are at the center of the new hyperrectangles (Figure 2).

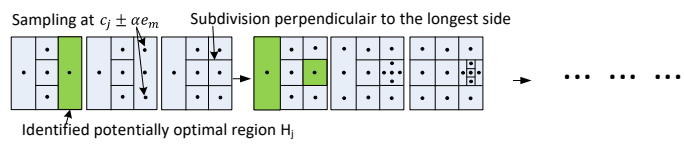

Figure 2. Two global optimization iterations

The global convergence of Algorithm 1 may come at the cost of a slow optimization at the final phase. In this work, we overcome this limitation by stopping the search when the hyperrectangle with the lowest objective function is sufficiently small. At this stage, the subdivisions become clustered near the global solution and the algorithm enters a refinement stage. The *stopping criteria* is given as $d_j < \sigma d_0$. It stops the search when the size of the hyperrectangle H_j with the best objective function at its center c_j (i.e., d_j) reaches a certain percentage of the original unit hypercube size d_0. $0 < \sigma < 1$ is adjusted for a tradeoff between computational cost and the solution optimality. It should ensure that no region is omitted and minimizes the risk of a premature termination.

The outputs of of Algorithm 1 are the best solution $\overline{x^*}$ reached by the subproblems and the simulation data Γ (Line 16), where Γ is composed of all sampled center points and their function evaluations.

B. Local Optimization

After a design point $\overline{x^*}$ in the basin of convergence is identified, a local search is invoked to speed up the convergence. The local search iteratively builds and optimizes a linear and non expensive RBF model of the objective function within a neighborhood of a current iterate, as given in Equation 4 [9]:

$$\begin{aligned} \min \quad & m_k(x_k + s), \quad x_k + s \in B_k \\ B_k \quad &= \quad \{x_k + s, s \in R^n : \|s\|_2 \leq \triangle_k\} \\ m_k(x_k + s) \quad &= \quad \sum_{i=1}^{|\Psi|} \lambda_i \phi(\|s - y_i\|_2) + p(s) \\ f(y_i) \quad &= \quad m_k(y_i), \quad \forall y_i \in \Psi \end{aligned} \tag{4}$$

where x_k is the current state, B_k is the so called trust region for an implied (center, radius) pair (x_k, $\triangle_k > 0$) and $\|.\|_2$ is the l_2 norm. The model m_k approximates f within a neighborhood of the current trust region B_k. It is a linear combination of RBFs (Line 3 in Equation 4). $\phi : R_+ \rightarrow R$ is a univariate RBF. λ_i are the linear model coefficients, which are determined by requiring that the model m_k interpolates the function f at a set of linearly independent data points $\Psi = \{y_i, f(y_i)\}$ (Line 4 in Equation 4) at which the values of f are known, including the current iterate x_k. The interpolation results in a system of linear equations [9]. $p(s)$ is a low order polynomial tail that guarantees the uniqueness of m_k (i.e., the model unknown coefficients λ_i). $|\Psi|$ is the cardinality of Ψ. The model m_k can be formed using as little as $n+1$ data points [9]. Algorithm 2 illustrates the method at each iteration.

Alg. 2 Local optimization

Require: Γ: Available simulation data points, $x_0 = \overline{x^*}$: Starting point, f: Objective function

1: **while** $\|\nabla m_k(x_k)\| \geq \varepsilon$ **do**
2: Select $\Psi \in \Gamma$ in the neighborhood of B_k.
3: Build m_k interpolating f at Ψ
4: minimize m_k within B_k and compte s_k
5: Evaluate $f(x_k + s_k)$ and update Γ
6: Compute ρ_k and adjust B_k
7: **end while**
8: **return** x^*: optimal solution

The current iterate x_k is usually surrounded by several neighbored points which have been evaluated previously in Algorithm 1. These simulation data points are reused to accelerate the local optimization phase. That is, at each iteration, the algorithm selects a set of data points $\Psi \in \Gamma$ within a neighborhood of the trust region B_k (Line 2). If the neighboring points are not enough for linear interpolation, new points in the neighborhood of x_k are properly generated [9]. Then, the model m_k that interpolates f is built (Line 3) and the unknown model coefficients λ_i are determined. The approximated solution s_k (i.e., the step) is computed by optimizing m_k over the trust region B_k (Line 4). The yield is evaluated at $x_k + s_k$ and the set Γ is updated (Line 5). In fact, any evaluated design point is saved in Γ, which allows the algorithm to gain additional insight into the function in the next iterations.

The pair (x_k, $\triangle_k$) of the trust region B_k is adjusted according to the ratio of the achieved versus the predicted improvement (i.e., decrease of the objective function f), $\rho_k = \frac{f(x_k)-f(x_k+s_k)}{m_k(x_k)-m_k(x_k+s_k)}$ (Line 6). if ρ_k is sufficiently positive, then the iteration is successful; the next iteration point $x_{k+1} = x_k + s_k$ will be taken and the trust-region radius $\triangle_k$ is enlarged. If ρ_k is not sufficiently positive, then the iteration was not successful; the current x_k will be kept and the trust-region radius is reduced. The process is repeated until the model gradient $\|\nabla m_k(x_k)\|$ is smaller than a threshold parameter ε. That is, the sequence of x_k converges to a stationary point. The convergence criteria proof can be found in [9].

III. Application - Folded Cascode Amplifier

In this section, we present the results of the application of our yield optimization technique on a folded cascode amplifier [6] circuit as shown in Figure 3.

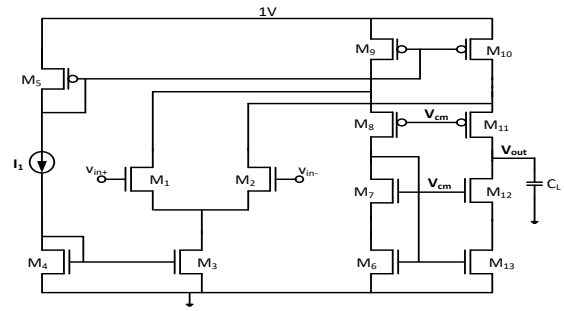

Figure 3. Fully differential folded-cascode amplifier

In the experiments, the circuit is designed in a commercial TSMC 65 nm process and simulated in HSPICE with BSIM4 transistor models. We use the TSMC 65 nm transistor mismatch model [1]. The local mismatch variables are considered as the process parameters including the oxide thickness $\triangle t_{ox}$, threshold voltage under zero bias $\triangle V_{th}$, channel width $\triangle w$ and channel length $\triangle L$. Each process parameter follows a truncated normal distribution [1]. All presented methods are implemented via a link between MATLAB and HSPICE. Also, the yield of each feasible design point is evaluated using MC analysis with the LHS technique and 2000 samples. In Algorithm 1, we set $\gamma = 10^{-3}$ and we subdivide the optimization into $S = 4$ subproblems. Algorithm 2 uses sequential quadratic programming (SQP) and cubic RBF models.

After applying the symmetry constraints, the number of independent design variables is 9. The local mismatch variables of each transistor pair are considered, which results in 48 process parameters. Table I provides the specifications for the gain Av, gainbandwidth GBW, power P_{DC} and slew rate SR.

Table I
Set of specifications for the folded cascode amplifier

Perf metrics	Spec	Perf metrics	Spec
$Av(dB)$	≥ 50	$GBW(MHz)$	≥ 40
$P_{DC}(mW)$	≤ 1	$SR(V/\mu s)$	≥ 60

The results of the proposed approach are reported in Table II. Alg1 refers to the parallel global optimization (i.e., Algorithm 1) and Alg2 refers to the local optimization (i.e., Algorithm 2). The number of yield evaluations and the yield values reached by each phase are reported in Columns 2 and 3, respectively. We also perform the optimization with different stopping criteria parameter values σ of the global optimization step. The relative error of the yield estimation at the optimized design point x^* is computed by evaluating its relative deviation to the yield provided by 70000 MC simulations in HSPICE at the same design point and given as $Rel\ Err = \frac{|g(70000-sim)-g(2000-sim)|}{|g(70000-sim)|} \times 100$.

Table II
EXPERIMENTAL RESULTS FOR THE CASCODE AMPLIFIER

σ	Yield Eval (#)			Yield (%)		Rel Err (%)
	Alg1	Alg2	Total	Alg1	Alg2	
0.2	189	45	234	73.05	75.02	0.21
0.1	247	11	258	81.03	84.35	0.21
0.005	602	9	611	83.54	84.34	0.20

Using $\sigma = 0.1$, the proposed method locates the best yield solution. In this case, Alg1 reaches a near optimal solution with 247 yield evaluations. The local optimization needs to perform only 11 yield evaluations to converge to a higher quality design point. A close solution is reached with $\sigma = 0.005$. However, it requires 2X more yield evaluations. In fact, the value $\sigma = 0.1$ (i.e., 10% of the original search space size) offers a good tradeoff between the solution optimality and the required number of yield estimations.

The proposed method finds a lower yield percentage with $\sigma = 0.2$. In this case, the sampling and subdivision strategy did not accurately locate the basin of convergence. Consequently, Alg2 fails to locate a high yield solution. In fact, the result of the local refinement requires a good starting point. However, in all experiments, it uses a small number of yield evaluation. Its low computational cost is achieved thanks to the optimization of a non expensive and local model of the yield and the simulation data reuse strategy.

Table III
EXPERIMENTAL RESULTS OF ALGORITHM1 APPLIED SOLELY

σ	Yield Eval (#)	Yield (%)	Rel Err (%)
0.0001	961	84.34	0.20
0.0003	871	84.02	0.21

We apply Alg 1 solely to locate the most robust design point with the optimum yield. The results are reported in Table III. Alg 1 applied with a low σ value succeeds in locating a good solution. However, it requires almost 4X higher number of yield evaluations, when compared to our approach with $\sigma = 0.1$. This observation confirms the slow convergence of the modified Lipshitiz optimization, despite its good search ability. The integration of a local refinement phase significantly decreases the number of yield evaluations and accelerates the optimization.

We compare our experimental results with high-ability algorithms including Genetic Algorithm (GA), Differential Evolution (DE) algorithm and GA-SA (Genetic Algorithm-Simulated Annealing), employed to optimize the yield for the cascode amplifier circuit. GA-SA uses GA as the global exploration mechanism and the simulated annealing (SA) algorithm to perform a local refinement. For all three methods, the feasible design space D_0 (i.e., the search space) is the same as the one used in the proposed method. The evaluation of the yield is accomplished using MC simulations in HSPICE. For both GA and DE, the population size is 80 and the crossover rate is 0.8 [6]. The population is initialized by randomly selecting values of the design variables within D_0.

We executed 20 runs of each algorithm starting from 20 different initializations. Table IV shows the best results in terms of yield quality among the 20 runs. We also include the result of the proposed method with $\sigma = 0.1$.

Table IV
COMPARISON WITH SIMULATION-BASED STOCHASTIC SEARCH METHODS

Method	Yield Eval (#)	Yield (%)	Rel Err (%)	Time [h]
Proposed	258	84.35	0.19	37.33
GA	495	60.61	0.20	110.01
DE	485	65.98	0.20	107.77
GA-SA	508	68.11	0.19	112.88

Our method is able to locate a higher yield rate with less computational time. The reduced computational time comes from: (1) the reduction of the search space allowed by the problem subdivision and the parallel computation; and (2) alleviating the slow convergence problem of the global search by the integration of a non expensive and linear local model-based optimization. Furthermore, the search ability of our approach obviously outperforms the stochastic optimization-based method thanks to an exhaustive exploration of potentially optimal regions. It can also be observed that neither DE nor the hybrid approach GA-SA is able to perform a reliable optimization, even though multiple runs were tried and the best optimization result is presented.

IV. CONCLUSION

In this paper, we proposed a novel method for analog yield optimization using a partition-based global search algorithm, which samples the most potential region of the feasible design space. A surrogate model-based local search is then integrated to highly speedup the convergence. Its efficiency is elevated by the reuse of existing simulation data of the global search phase. Compared with simulation-based stochastic optimization, our method identifies more robust design points with less run-time and without affecting the accuracy.

REFERENCES

[1] TSMC 65*nm* CMOS Process Technology. http://www.tsmc.com, 2016.

[2] R. Horst and P. Pardalos. *Handbook of global optimization*. Kluwer Academic Publishers, 1995.

[3] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[4] O. Lahiouel, M. H. Zaki, and S. Tahar. Towards enhancing analog circuits sizing using SMT-based techniques. *Design Automation Conference*, pages 1–6, 2015.

[5] B. Liu, F. V. Fernandez, and G. E. Gielen. Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques. *IEEE Transactions on CAD*, 30(6):793–805, 2011.

[6] B. Liu, G. Gielen, and F. Fernandez. *Automated Design of Analog and High-frequency Circuits*. Springer, 2014.

[7] R. Paulavičius and J. Žilinskas. Simplicial Lipschitz optimization without the Lipschitz constant. *Journal of Global Optimization*, 59(1):23–40, 2014.

[8] R. Schwencker, F. Schenkel, M. Pronath, and H. Graeb. Analog circuit sizing using adaptive worst-case parameter sets. *Design, Automation and Test in Europe Conference*, pages 581–585, 2002.

[9] S. M. Wild, R. G. Regis, and C. A. Shoemaker. Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, Oct. 2008.