Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing based on Computational Intelligence Techniques

Bo Liu, Francisco V. Fernández and Georges Gielen, *Fellow, IEEE*

Abstract—**In nanometer CMOS technologies, worst-case design methods and response-surface-based yield optimization methods face challenges in accuracy. Monte-Carlo (MC) simulation is general and accurate for yield estimation, but its efficiency is not high enough to make MC-based analog yield optimization, which requires many yield estimations, practical. In this paper, techniques inspired by computational intelligence are used to speed up yield optimization without sacrificing accuracy. A new sampling-based yield optimization approach, which determines the device sizes to optimize yield, is presented, called the Ordinal Optimization (OO)-based Random-Scale Differential Evolution (ORDE) algorithm. By proposing a two-stage estimation flow and introducing the OO technique in the first stage, sufficient samples are allocated to promising solutions, and repeated MC simulations of non-critical solutions are avoided. By the proposed evolutionary algorithm that uses Differential Evolution for global search and a random-scale mutation operator for fine tunings, the convergence speed of the yield optimization can be enhanced significantly. With the same accuracy, the resulting ORDE algorithm can achieve approximately a tenfold improvement in computational effort compared to an improved MC-based yield optimization algorithm integrating the infeasible sampling and Latin-hypercube sampling techniques. Furthermore, ORDE is extended from plain yield optimization to process-variation-aware single-objective circuit sizing.**

Index Terms—**Yield optimization, variation-aware analog sizing, ordinal optimization, differential evolution**

I. INTRODUCTION

Industrial analog integrated circuit design not only calls for fully optimized nominal design solutions, but also requires high robustness and yield in the light of varying supply voltage and temperature conditions, as well as inter-die and intra-die process variations [1-3]. Especially in nanometer CMOS technologies, random and systematic process variations have a large influence on the quality and yield of the manufactured analog circuits. As a consequence, in the high-performance analog and mixed-signal design flows, the designer needs guidelines and tools to deal with these factors impacting circuit yield and performances in an integrated manner in order to avoid costly re-design iterations [4].

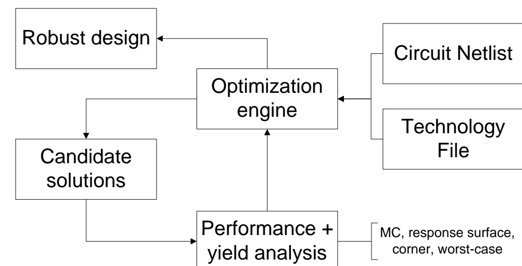

Fig. 1. General flow of yield optimization methods

Yield optimization includes system-level hierarchical optimization [5] and building-block-level yield optimization [6-7]. At the building block level, there exist parametric yield optimization [6-8] and layout-related yield optimization [9-11], e.g. critical area yield analysis [9]. This paper focuses on parametric yield optimization at the building-block level.

The yield optimization flow is summarized in Fig. 1. In the optimization loop, the candidate circuit parameters are generated by the optimization engine; the performances and yield are analyzed and fed back to the optimization engine for the next iteration. Yield analysis is a critical point in the yield optimization flow. Among the factors that impact yield, statistical inter-die and intra-die process variations play a vital role [8]. Previous yield optimization methods include device model corner-based methods [3,12], performance-specific worst-case design (PSWCD) methods [6,7], response-surface-based methods [2,15] and Monte-Carlo (MC)-based methods.

- Device model corner-based methods [3,12] use the same slow/fast parameter sets to decide the worst-case parameters for all circuits for a given technology. They

Manuscript received Jan.21, 2010; revised May 29 and Sept. 22, 2010. This research was supported by a special bilateral agreement scholarship of Katholieke Universiteit Leuven, Belgium and Tsinghua University, P. R. China, and by the TIC-2532 Project, funded by Consejeria de Innovación, Ciencia y Empresa, Junta de Andalucia, Spain. This paper was recommended by Associate Editor Peng Li.
Georges Gielen and Bo Liu are with the ESAT-MICAS, Katholieke Universiteit Leuven, Leuven, Belgium. (e-mail: {Georges.Gielen, Bo.Liu} @esat.kuleuven.be, liu_bo765@yahoo.com.cn). Francisco Fernández is with IMSE, CSIC and University of Sevilla, Sevilla, Spain. (e-mail: Francisco.Fernandez@imse-cnm.csic.es).

are efficient due to the limited number of simulations needed. But their drawback is that the worst-case performance values are pessimistic as the corners correspond to the tails of the joint probability density function of the parameters, resulting in considerable over-design. Also, the slow/fast values obtained for a single performance, e.g. delay, and the worst-case process parameters for other performances may be different. Secondly, the actual yield may be low if the intra-die variations are ignored. If the intra-die variations were considered, the number of simulations would be extremely large. The limitations of device model corner-based methods for robust analog sizing are discussed in [1,13].

- The PSWCD methods [6-7,13-14] represent an important progress in robust sizing of analog ICs. Instead of using the same slow/fast parameter sets for all the circuits, the PSWCD methods decide on the worst-case parameters for specific performances of each circuit and nominal design. Determining the performance-specific worst-case parameters is critical for this kind of method. Although the search for the WC point typically uses some nonlinear optimization formulation, most PSWCD methods [13] linearize the performances at the worst-case point, which can introduce inherent errors. Some PSWCD methods build a response surface between the inter-die parameters and the performances [14] (RSM PSWCD). The inter-die parameters are independent of the design parameters, but intra-die variations have correlations to the design parameters. Hence, intra-die variations cannot be considered in these methods. If considering intra-variations, the total number of the process variation variables increases significantly with the number of the devices. While some PSWCD methods calculate an approximate estimation of the yield, others do not calculate yield. Instead, they calculate a range of the process parameters for a given yield, in which the specifications are met. In this case, the estimated yield is not available explicitly and the method has to be run repeatedly with different target values (e.g. yield>95%-99%) to know the best yield that can be achieved.
- In response-surface-based methods, first macro-models over the yield and the designable and process parameters are established through regression methods and these are subsequently used to estimate the yield in the sizing process. Macro-models can be classified into white-box models and black-box models. A white-box model analytically expresses the yield as a function of the design and process parameters. Some additional parameters are used for regression purposes. Black-box models, on the other hand, do not consider analytical expressions of the yield, but construct a regression model according to the input (i.e. design points, process parameters) and the output (i.e. yield) data. Accurate yield-aware performance macro-models can make a sizing tool explore design alternatives with little computational cost. However, response-surface-based methods suffer from the trade-off between the accuracy and the complexity of the model, as well as the accuracy and the number of samples (CPU time) to create the model.
- MC-based methods have the advantages of generality and high accuracy [16], so they are the most reliable and commonly used technique for yield estimation. Nevertheless, a large number of simulations are needed for MC analysis, therefore preventing its use within an iterative yield optimization loop (Fig. 1). Some speed enhancement techniques for MC simulations based on Design of Experiments (DOE) techniques have been proposed, such as the Latin Hypercube Sampling (LHS) method [17,18] or the Quasi-Monte-Carlo (QMC) method [19,20], to replace Primitive Monte-Carlo (PMC) simulation. These speed improvements are very significant, but our experiments show that the computational load is still too large for yield optimization if only using DOE methods in real practice (see section IV).

Currently, PSWCD methods and response-surface-based methods are the most popular approaches in the repeated iterations within yield optimization loops, while some form of Monte-Carlo yield estimation is most popular in design verification.

Therefore, in this paper we address the efficiency of MC-based yield optimization by proposing a different (but complementary) approach exploiting techniques from computational intelligence: while keeping high accuracy, we dramatically increase the efficiency of yield optimization by (1) optimally allocating the computing budget to candidate solutions in order to avoid non-critical MC simulations; and (2) enhancing the convergence speed of the search strategy by means of a random-scale (RS) mutation operator in combination with the differential evolution (DE) algorithm to decrease the amount of expensive MC simulations.

Based on the above ideas, we then present the Ordinal Optimization (OO)-based Random-Scale Differential Evolution (ORDE) algorithm for analog yield optimization. The method aims to:

- be general enough to be applied to any analog circuit in any technology process and for any distribution of the process parameters;
- simultaneously handle inter-die and intra-die variations in nanometer technologies;
- provide highly accurate results comparable to Monte-Carlo analysis;
- use an order of magnitude less computational effort compared with the improved MC-based method

integrating the infeasible pruning and Latin Hypercube sampling techniques (Section III (A)) and as such making the computational time of accurate yield optimization practical.

The remainder of the paper is organized as follows. Section II reviews basic concepts of yield optimization. Section III introduces the components and the general framework of ORDE. Section IV tests ORDE on practical examples. Comparisons with response-surface-based methods are also performed. In Section V, the ORDE algorithm is extended from plain yield optimization to process-variation-aware single-objective analog sizing, which optimizes a target design objective (e.g. power) subject to a minimum yield requirement. The concluding remarks are presented in Section VI.

II. Basics of Yield Optimization

The aim of yield optimization is to find a circuit design point d^* that has a maximum yield, considering the manufacturing and environmental variations [8]. In the following, we will elaborate the design space *D*, process parameter space *S* with distribution *pdf(s)*, environmental parameter space Θ and specifications *P*.

The design space *D* is the search space of the circuit design points, *d*, which can be transistor widths and lengths, resistances, capacitances and bias voltages and currents. Each one has an upper and lower bound, which is determined by the technological process or the user's setup. The process parameter space *S* is the space of statistical parameters reflecting the process fluctuations, e.g. oxide thickness T_{ox} and threshold voltage V_{th}. Process parameter variations can be inter-die or intra-die. For an accurate model, both types should be considered. The environmental variables Θ include temperature and power supply voltage. The specifications *P* are the requirements set by the designer, which can be classified into performance constraints (e.g. DC gain > 70dB) and functional constraints (e.g. transistors must work in the saturation region).

The yield optimization problem can be formulated as to find a design point d^* that maximizes yield (in the case of plain yield optimization) [13]:

$$d^* = \arg\max_{d \in D} \{Y(d)\} \quad (1.1)$$

or that minimizes some function *f* (e.g. power, area) subject to a minimum yield requirement *y* (in the case of yield-aware sizing) [17]:

$$d^* = \arg\min_{d \in D} \{f(d, s, \theta)\} \; s \in S, \theta \in \Theta$$
$$s.t. \; Y(d) \geq y \quad (1.2)$$

Yield is defined as the percentage of manufactured circuits that meet all the specifications considering process and environmental variations. Hence, yield can be formulated as:

$$Y(d) = E\{YS(d, s, \theta) \mid pdf(s)\} \quad (2)$$

where *E* is the expected value. $YS(d, s, \theta)$ is equal to 1 if the performance of d meets all the specifications considering *s* (process fluctuation) and θ (environmental variation); otherwise, $YS(d, s, \theta)$ is equal to 0. In most analog circuits, circuit performances change monotonically with the environmental variables θ. Then, the impact of environmental variations can be handled by simulations at the extreme values of the environmental variables. For instance, if the power supply may experience some variations, e.g, 10%, the largest degradation is obtained by simulating at the extreme values: $(1 \pm 10\%) \times$ nominal value. Process variations, on the other hand, are much more complex: directly simulating the extreme values (classical worst-case analysis [1]) may cause serious over-design. This work therefore focuses on the impact of statistical process variations (space *S*) in yield optimization.

III. The ORDE Algorithm

A. The Use of Infeasible Pruning and DOE in ORDE

To satisfy the first three goals (be general enough, able to handle both inter-die and intra-die variations, high accuracy) from Section I, MC analysis is selected. The speed enhancement technique, DOE, for MC-based yield estimation is used. The DOE method implemented in ORDE is LHS. However, the key contributions of ORDE are not related to a particular sampling mechanism, therefore, other speed acceleration methods like the recently proposed QMC [19,20] can be integrated. In the yield optimization process, some candidate solutions will appear that cannot satisfy the specifications even for nominal values of process parameters. Their yield values will be too low to become a useful candidate solution. Hence, there is not much sense in applying the MC-based yield estimation to these solutions. In ORDE, we call them infeasible solutions and assign them a zero yield value. Their violation of constraints is calculated, and the constrained optimization algorithm will minimize the constraint violations and move the search space to feasible solutions (i.e. design points that satisfy the specifications for nominal process parameters). This technique is named "infeasible pruning" in this paper. The selected feasible solutions are handled by ordinal optimization, which is described below.

B. Basics of ORDE

Many recent analog circuit sizing and yield optimization methodologies are based on evolutionary computation (EC), which relies on the evolution of a set of candidate solutions, commonly called population, along a set of iterations, commonly called generations [19]. The computational effort at each iteration and the necessary number of iterations are two key factors that affect the speed of the yield optimization. We solve these two problems by optimally allocating the computing budget to each candidate in the population (reducing the computational effort at each iteration) and by improving the search mechanism (decreasing the necessary number of iterations, and, hence, decreasing the number of expensive MC simulations). Therefore, the total computational effort can be reduced considerably. In this paper, we use two computational intelligence techniques to implement these two key ideas.

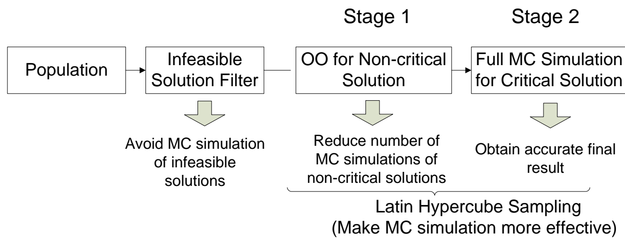

Fig. 2. Two-stage yield estimation flow

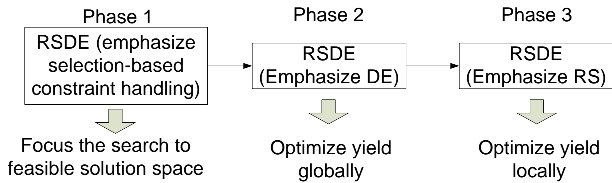

Fig. 3. Yield optimization flow

Our yield estimation flow is depicted in Fig. 2. In order to optimally allocate the computing budget at each iteration, instead of assigning the same number of MC simulations to feasible solutions, the yield estimation process is divided into two stages. In the first stage, the fitness ranking of the candidate solutions and a reasonably accurate yield estimation result for good (critical) solutions are important. For medium or bad (non-critical) candidate solutions, their ranking is important, but accurate yield estimation is not. The reason is that the function of the yield estimation for non-critical candidates is to guide the selection operator in the EC algorithm, but the candidates themselves are likely not to be selected as the final result or even not enter the second stage of the yield optimization flow. Hence, the computational efforts spent on feasible but non-optimal candidate solutions can be strongly reduced. On the other hand, the estimations for these non-critical candidates cannot be too inaccurate either. After all, correct selection of candidate solutions in the yield optimization algorithm is necessary. In the first stage, the yield optimization problem is therefore formulated as an ordinal optimization problem, aimed at identifying critical candidate solutions by allocating a sufficient number of samples to the MC simulation of these solutions, while reasonably few samples are allocated to non-critical solutions [22]. Notice that this approach is intended to assign a different number of MC simulations to the yield estimations of the different candidates. This is different, and compatible, with the efficient sampling addressed with any DOE technique. In the second stage of the ORDE method, an accurate result is highly important, so the number of simulations within each yield estimation is increased in the second stage to obtain an accurate yield value.

Another key technique of ORDE is to decrease the necessary number of iterations of the optimization flow shown in Fig. 3. Instead of using conventional EC algorithms, we design a selection-based random-scale differential evolution (DE) algorithm (RSDE), which is a combination of three different techniques. Each technique plays a significant role in each phase. The first phase emphasizes a selection-based method to focus the search into the feasible solution space, defined by the nominal values of the process parameters. We use the DE framework [23] (a powerful and fast global optimization algorithm) for global search (emphasized in the second phase) and a random-scale operator for fine tunings (emphasized in the third phase).

In the following, the basic components of ORDE will be introduced first, and the general framework will then be presented.

C. Introducing Ordinal Optimization into Yield Optimization

Ordinal optimization (OO) has emerged as an efficient technique for simulation and optimization, especially for problems where the computation of the simulation models is time consuming [22]. OO is based on two basic tenets: (a) Obtaining the "order" of a set of candidates is easier than estimating an accurate "value" of each candidate. The convergence rate of ordinal optimization is exponential. This means that the probability that the observed best solution is among the true best solutions grows as $O(e^{-\alpha n})$ where α is a positive real number and n is the number of simulations [22]. In contrast, the convergence rate of methods aimed at estimating the right value instead of the order, e.g. the direct Monte Carlo method, is at most $O(1/\sqrt{n})$ [24]. (b) An accurate estimation is very costly but a satisfactory value can be obtained much easier.

Therefore, OO fits the objectives of the first stage of yield estimation of ORDE (see Fig. 2) quite well. In the

first stage, a bunch of good designs are selected through evolution and sent to the second stage. The requirement is a correct selection with a reasonably accurate yield estimation and with the smallest computational effort. According to OO, a large portion of the simulations should be conducted with those critical solutions in order to reduce the estimator variance. On the other hand, limited computational effort should be spent on non-critical solutions that have little effect on identifying the good solutions, even if they have large variances. This leads to the core problem in ordinal optimization: allocating the computing budget, which can be formulated as follows. Given a pre-defined computing budget, how should it be distributed among the candidate designs?

Consider the yield evaluation function. For a single simulation (e.g. a sample of process parameters), we define $YS(d,s)$ =1 if all the circuit specifications are met, and $YS(d,s)$ =0 otherwise. Because the MC simulation determines the yield as the ratio of the number of functional chips to all fabricated chips, the mean value of $YS(d,s)$, corresponds to the yield value, $Y(d)$. Let us consider a total computing budget equal to T simulations. In ORDE, T is determined by the number of feasible solutions (i.e. solutions that meet the performance constraints for nominal values of the process parameters) at each generation. Here, we set $T = sim_{ave} \times M1$, where $M1$ is the number of feasible solutions and sim_{ave} is the average budget for each candidate set by the user. The budget allocation problem consists in determining the number of simulations $n_1, n_2, \cdots, n_{M1}$ of the $M1$ candidate solutions such that $n_1 + n_2 + \cdots n_{M1} = T$. For this problem, several algorithms have been reported in the specialized literature [25,26]. An asymptotic solution to this optimal computing budget allocation problem is proposed in [25]:

$$n_b = \sigma_b \left(\sum_{i=1, i\neq b}^{M1} n_i^2 / \sigma_i^2\right)^{1/2} \tag{3}$$
$$n_i / n_j = \left(\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}}\right)^2, \quad i, j \in \{1, 2, \cdots, M1\}, i \neq j \neq b$$

where b is the best design of the $M1$ candidate solutions (represented by the highest estimated yield value based on the available samples for each candidate). For each candidate solution, some samples are allocated. For each sample, the corresponding $YS(d,s)$ can be computed (0 or 1). By these $YS(d,s)$, we can calculate their mean (estimated yield, $Y(d)$) and $\sigma_1^2, \sigma_2^2, \cdots, \sigma_{M1}^2$, which are the finite variances of the $M1$ solutions, respectively. They measure the accuracy of the estimation. Parameter $\delta_{b,i} = Y_b(d) - Y_i(d)$ represents the deviations of the estimated yield value of each design solution with respect to that of the best design. The interpretation of (3) is quite intuitive. If $\delta_{b,i}$ is large, the estimated yield value of design i is bad, and according to $n_i / n_j = (\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}})^2$, n_i becomes small, i.e., we should not allocate many simulations to this design. However if σ_i is large, it means that the accuracy of the yield estimation is low, and we should allocate more simulations to this design to obtain a better yield estimate. Therefore, the quotient $\sigma_i / \delta_{b,i}$ represents a trade-off between the yield value of design i and the accuracy of its estimation. Therefore, an OO-based yield analysis algorithm can be designed as follows:

Algorithm 1. Ordinal optimization for analog yield analysis

Step 0: Let k =0, and perform n_0 simulations for each feasible design, i.e. $n_i^k = n_0, i = 1, 2, ..., M1$.

Step 1: If $\sum_{i=1}^{M1} n_i^k \geq T$, stop the OO for yield analysis.

Step 2: Consider Δ additional simulations (refer to [22] for the selection of the Δ and n_0 values) and compute the new budget allocation $n_i^{k+1}, i = 1, 2, ..., M1$ by eqn. (3). If $n_i^{k+1} \geq n_{\max}$, then $n_i^{k+1} = n_{\max}$.

Step 3: Perform additional $\max\{0, n_i^{k+1} - n_i^k\}$ simulations for design d_i, $i = 1, 2, ..., M1$. Let $k = k+1$ and go to step 1.

Parameter n_0 is the initial number of simulations for each candidate solution, selected to provide a very rough idea of the yield. More simulations are allocated according to the quality of the candidate later on. Parameter $n_{\max}$ is the upper limit of the number of simulations for any candidate. The value of $n_{\max}$ must call for a balance between the accuracy and the efficiency.

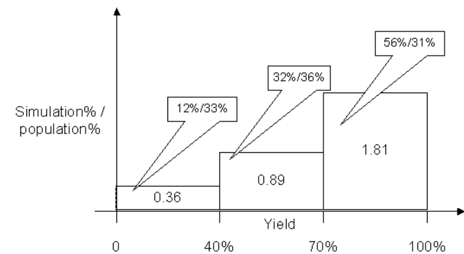

Fig. 4. The function of OO in a typical population

A typical population from example 2, described in section IV later on, is selected to show the benefits of OO (see Fig. 4): candidates with a yield value larger than 70% correspond to 31% of the population, and are assigned 56% of the simulations. Candidates with a yield value smaller than 40% correspond to 33% of the population, and are only assigned 12% of the simulations. The total number of simulations is 10.2% of those of the infeasible pruning (IP)+LHS method applied to the same candidate designs, because repeated MC simulations of non-critical solutions are avoided.

The above technique is used until the yield converges close to the desired value. For example, if the desired target yield is 99%, the threshold value between the first and the second stage can be 97%. Candidates with an estimated yield larger than the threshold value enter the second stage. In this stage, all the candidates are assigned the specified maximum number ($n_{\max}$) of samples to guarantee the accuracy of the final result, while other candidates in the population still remain in the first stage and still use the estimation method described previously. Note that the two stages are therefore not separated in time, but rather use different yield estimation methods.

The threshold value must be properly selected. A too low threshold value may cause low efficiency, as OO would stop when the yield values of the selected points are not promising enough (e.g. a 50% yield threshold for a requirement of 90% yield) and shifts the yield estimation and selection tasks to the second stage, which is more CPU expensive. A too high threshold value (e.g. a threshold equal to the yield requirement) may cause low accuracy. The reason is that in most cases the points selected by OO are promising (it can compare the candidates and do the selection correctly) but the estimated yield values are not sufficiently accurate for the final result. Assigning the threshold to be two percentage points below the required target yield represents an appropriate trade-off between efficiency and accuracy.

D. Brief Introduction to the DE Algorithm

In addition to introducing OO to decrease the computational effort at each iteration, decreasing the necessary number of iterations is another key objective. The DE algorithm [23] is selected as the global search engine. The DE algorithm outperforms many EC algorithms in terms of solution quality and convergence speed [23]. DE uses a simple differential operator to create new candidate solutions and a one-to-one competition scheme to greedily select new candidates.

The *i*-th candidate solution in the *Q*-dimensional search space at generation *t* can be represented as

$$d_i(t) = [d_{i,1}, d_{i,2}, \cdots, d_{i,Q}] \tag{4}$$

At each generation *t*, the *mutation* and *crossover* operators are applied to the candidate solutions, and a new population arises. Then, *selection* takes place, and the corresponding candidate solutions from both populations compete to comprise the next generation. The operators are now explained in detail.

For each target candidate solution, according to the *mutation* operator, a *mutant vector* is built:

$$V_i(t+1) = [v_{i,1}(t+1), \ldots, \ v_{i,Q}(t+1)] \tag{5}$$

It is generated by adding the weighted difference between a given number of candidate solutions randomly selected from the previous population to another candidate solution. In ORDE, the latter one is selected to be the best individual in the current population. The mutation operation is therefore described by the following equation:

$$V_i(t+1) = d_{best}(t) + F(d_{r1}(t) - d_{r2}(t)) \tag{6}$$

where indices r_1 and r_2 ($r_1, r_2 \in \{1, 2, \ldots, M\}$) are randomly chosen and mutually different, and also different from the current index *i*. Parameter $F \in (0, 2]$ is a constant called the scaling factor, which controls the amplification of the differential variation $d_{r1}(t) - d_{r2}(t)$. The population size *M* must be at least 4, so that the mutation can be applied. The base vector to be perturbed, $d_{best}(t)$, is the best member of the current population, so that the best information can be shared among the population.

After the *mutation* phase, the *crossover* operator is applied to increase the diversity of the population. Thus, for each target candidate solution, a *trial vector* is generated as follows:

$$U_i(t+1) = [u_{i,1}(t+1), \ldots, u_{i,Q}(t+1)] \tag{7}$$

$$u_{i,j}(t+1) = \begin{cases} v_{i,j}(t+1), if\ (rand(i,j) \le CR)\ or\ j = randn(i), \\ d_{i,j}(t), otherwise, \end{cases} \tag{8}$$

where $j = 1, 2, \ldots, Q$ and $rand(i, j)$ is an independent random number uniformly distributed in the range [0,1]. Parameter *randn(i)* is a randomly chosen index from the set $\{1, 2, \ldots, Q\}$. Parameter $CR \in [0,1]$ is a constant called the crossover parameter, which controls the diversity of the population.

Following the *crossover* operation, the *selection* operation decides whether the *trial vector* $U_i(t+1)$ will be a member of the population of the next generation *t*+1 or not. For a minimization problem, $U_i(t+1)$ is compared to the initial target candidate solution $d_i(t)$ by the

following one-to-one-based greedy selection criterion:

$$d_i(t+1) = \begin{cases} U_i(t+1), if\ f(U_i(t+1)) < f(d_i(t)), \\ d_i(t), otherwise \end{cases} \quad (9)$$

where the function f is the objective function, i.e. the function to be minimized or maximized. In this paper, the objective function is the yield in the case of standard yield optimization and some circuit performance in the case of yield-aware sizing. The candidate solution, $d_i(t+1)$, becomes the candidate solution of the new population. Then, the next iteration begins.

E. Random-Scale Operator for Combined Global and Local Search

Although DE is a very powerful and fast global optimization algorithm, it is not so efficient in local tuning to reach the exact optimal solution (other global optimization algorithms, e.g. genetic algorithms, also have the same problem). But local tuning is emphasized in the fine-tuning phase (phase 3 in Fig. 3). Usually, in this phase, there are many candidates which are assigned the maximum number of simulations ($n_{\max}$) to estimate the yield, which is expensive, but otherwise the accuracy would degrade very significantly. Moreover, the global optimization mechanism must be maintained even in this phase, because otherwise the yield optimization has a high risk to be stuck at a premature solution. We therefore propose a combined global and local search mechanism, whose purpose is to enhance the convergence speed while at the same time maintaining the accuracy.

In the EC field, enhancing the local search ability is often achieved by memetic algorithms [27]. In addition to the global optimization engine, memetic algorithms use a population-based strategy coupled with individual search heuristics capable of performing local refinements. Local search methods can be classified into derivative-based methods (e.g. Quasi-Newton method [28]) and derivative-free methods (e.g. Hill Climbing method [29]). In derivative-based methods, calculating the required derivatives, e.g. Hessian matrix, often consumes numerous function evaluations when the number of design variables is large, especially when the derivatives cannot be expressed analytically. Normally, for medium-scale problems (10-20 design variables), derivative-free methods also need more than 20-30 iterations for each candidate, and each iteration needs $n_{\max}$ simulations. As this number has to be multiplied by the size of the population, this procedure becomes very expensive. Hence, for yield optimization, a cheaper method is necessary.

Instead of performing a separate global and local search, our proposed approach is to combine global search and local search into a unified procedure. In eqn. (6), the scaling factor F is a constant for all the candidate solutions. If F is small, the whole evolution process will be slow; if F is large, it is difficult to perform effective local fine-tunings. To solve the problem, a natural idea is to randomize F in eqn. (6) to each differential variation. By randomizing F, the differences of vectors can be amplified stochastically, and the diversity of the population is retained. This introduces two advantages: (1) The algorithm has a lower probability of providing premature solutions because of the reasonable diversity; (2) The vicinity of the *mutant vector* that the standard DE can explore, is investigated by the randomized amplification of the differential variation $d_{r1}(t) - d_{r2}(t)$. In a standard DE search process, the candidates may get stuck in a region and make the evolution quite slow when the global optimization point is nearly reached (but the diversity is also maintained). This is called “stagnation” [30]. Fig. 5 shows the effect of randomizing F. It can be seen that a cloud of potential points centered around the mutant vector with constant scaling factor *F1* have the potential to be investigated.

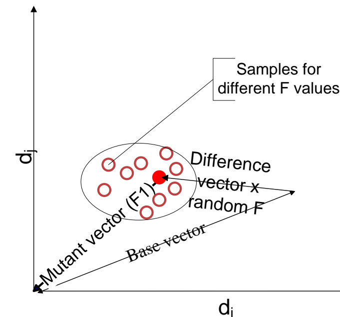

Fig. 5. Mutant vectors obtained by the random-scale operator

In our method, as scaling factor we use a vector $\hat{F}$ composed of Gaussian-distributed random variables with mean value μ and variance σ : $\hat{F}_{i,j} = norm(\mu, \sigma)$, $i = 1, 2, \ldots M$, $j = 1, 2, \ldots Q$. A Gaussian distribution is selected based on the following two considerations: (1) As the purpose of the random scaling factor is to search the vicinity of the mutant vectors by the constant F, it should not be far from it. By using a Gaussian distribution, 68% of the generated samples in $\hat{F}$ are within 1 σ. (2) It should have the ability to escape from the “stagnation”. A Gaussian distribution can also provide 5% of $\hat{F}$ values out of 2 σ. We have also tried uniform and Cauchy distributions for the scaling factor using benchmark problems in the EC field and found that the Gaussian-distributed $\hat{F}$ results in the best average objective function value. For each variable in the search space, the

scaling factor $\hat{F}_{i,j}$ of each differential variation $d_{r1}(t) - d_{r2}(t)$ is different. Eqn. (6) is therefore changed to:

$$V_i(t+1) = d_{best}(t) + \hat{F}_i(d_{r1}(t) - d_{r2}(t)) \qquad (10)$$

By the proposed combined global and local search mechanism, the necessary number of iterations of the yield optimization algorithm decreases significantly (see example 2 in Section IV).

F. Other Components

Besides the two key ideas described above, we use the selection method in [31] to handle the optimization constraints. They include both circuit performance constraints (e.g. gain larger than 80dB) and functional constraints (e.g. transistors must work in the saturation region). The advantages of this selection method and its combination with the DE algorithm for analog sizing have been shown in [32].

G. The General Framework of ORDE

Based on the above components, the overall ORDE algorithm for analog yield optimization can now be constructed. The detailed flow diagram is shown in Fig. 6.

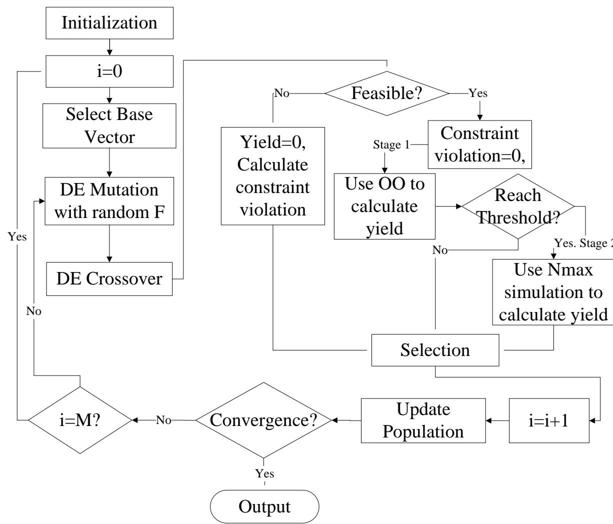

Fig. 6. Flow diagram of ORDE

The ORDE algorithm consists of the following steps.

Algorithm 2. The ORDE algorithm for analog yield optimization

Step 0: Initialize parameters n_0, T, Δ, n_{max} and the DE algorithm parameters (e.g. the population size M, the crossover rate CR). Initialize the population by randomly selecting values of the design variables d within the allowed ranges.
Step 1: Update the current best candidate. If no candidate meets the specifications for nominal process parameters, the best candidate is the one with the smallest constraint violation. Otherwise, the best candidate is the feasible candidate with the largest estimated yield.
Step 2: Perform the mutation operation according to eqn. (10) to obtain each candidate solution's mutant counterpart.
Step 3: Perform the crossover operation between each candidate solution and its corresponding mutant counterpart according to eqn. (8) to obtain each individual's trial individual.
Step 4: Check the feasibility of the trial individual. For feasible solutions, go to step 5.1; for infeasible solutions, go to step 5.2.
Step 5.1: Set constraint violations equal to 0, and use the OO technique described in Algorithm 1 to calculate the yield. If the estimated yield is higher than the threshold value, add additional samples to perform the full MC simulation. Go to step 6.
Step 5.2: Set yield equal to 0, and calculate the constraint violations. No yield is estimated in this step. Go to step 6.
Step 6: Perform selection between each candidate solution and its corresponding trial counterpart according to the rules in [31]: if both of them are not feasible, select the one with smaller constraint violation; if one is feasible and the other is infeasible, select the feasible one; if both are feasible, select the one with higher yield.
Step 8: If the stopping criterion is met (e.g. a convergence criterion or a maximum number of generations), then output d_{best} and its objective function value; otherwise go to Step 1.

IV. EXPERIMENTAL RESULTS AND COMPARISONS

In this section, the ORDE algorithm is demonstrated by two practical analog circuits in 0.35 μm and 90nm CMOS technologies, respectively. To highlight the effects of the two key contributions of ORDE, it will be compared with a reference method that combines the DE optimization engine, the selection-based constraint handling mechanism, infeasible pruning and LHS techniques. In the DE search engine, the population size is 50 and the crossover rate is 0.8. Except for ORDE, the DE scaling factor F in the other experiments is 0.8, which is a common setting [23]. In the random-scale search operator, we choose a Gaussian distribution with μ =0.75 and σ =0.25. The optimization process stops when the reported yield reaches 99%, or when the yield does not increase for 20 consecutive generations. If parameter n_0 is set to a too low value, the yield estimates are too inaccurate, even for the application of eqn. (3). If it is too high, the advantages of OO are lost. A value between 5 and 20 is recommended in [22]. We use n_0=15 in ORDE.

Parameter sim_{ave} is set to 35 in all experiments. It may seem that 35 simulations are too few to get an acceptable accuracy. However, there are 3 considerations that must be taken into account. (1) LHS sampling is used. It has been reported that LHS gets a comparable accuracy to primitive Monte-Carlo simulation (PMC) in circuit yield analysis with just 20%-25% the number of samples of PMC [34]. (2) Parameter sim_{ave} is just an average number of simulations of different candidates. By using OO, the MC simulations are optimally allocated according to the solution qualities, so promising candidate solutions are assigned much more than 35 simulations. According to experiments, some promising candidates are assigned more than 160 LHS simulations. (3) We do not need a very accurate result in the first stage, as the purpose of this stage is correct selection and getting a reasonably accurate yield estimation for promising points. The examples have been run on a PC with 4GB RAM and Linux operating system, in the MATLAB environment. Synopsys HSPICE electrical simulator is used as the circuit performance evaluator. The key techniques in this paper, i.e. the OO for yield optimization and the random-scale operator, are analyzed by statistical results here. The abilities of the DE optimization kernel for analog sizing compared with some other EC algorithms have been reported in [32], and will not be compared here.

A. Experimental Method

There are several aspects that have to be considered when designing the experiments. Firstly, the number of MC simulations for each feasible candidate should be decided. There is not much sense in comparing the efficiency without a good accuracy. The number of MC simulations in the second stage is the main factor that influences the accuracy of the final result. The accuracy of the yield estimates is related to the number of samples according to [1]:

$$n_{MC} \approx \frac{Y(1-Y)k_{\gamma}^{2}}{\Delta Y^{2}} \tag{11}$$

where Y is the yield value and ΔY is the confidence interval, e.g. if the yield estimate is 90%, and $\Delta Y = 1\%$, then the confidence interval is 89%-91%. Parameter k_{γ} reflects the confidence level, e.g. $k_{\gamma} = \pm 1.645$ denotes a 90% confidence level. From eqn. (11), the necessary number of MC simulations can be calculated. However, this corresponds to the primitive MC simulation. According to [34,35], LHS sampling requires 20%-25% the number of samples compared with PMC to get a comparable accuracy. Fig. 7 shows the estimated number of LHS simulations needed for a confidence level of 90%, 95%, and 99% respectively when $\Delta Y = 1\%$. The number of LHS simulations is estimated as 20% of the necessary number of PMC samples. It can be seen that even for a 99% confidence level, for a yield larger than 96%, 500 LHS points are sufficient. For a 90% confidence level, 500 LHS points are even sufficient for a yield larger than 90%. In all the experiments, the threshold to use 500 LHS simulations is 97%, so 500 LHS samples are enough for the required accuracy.

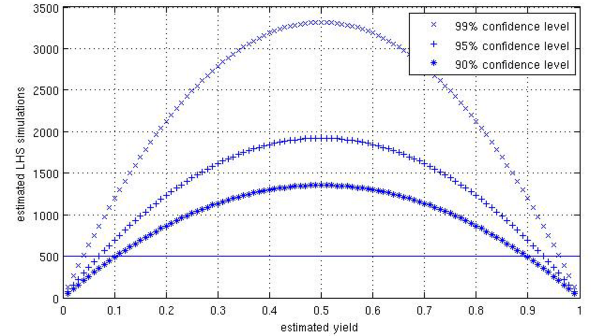

Fig. 7. Necessary numbers of LHS simulations

Secondly, the estimated yield result is influenced by the number of samples. Two experiments using 50 and 500 MC simulations for each feasible candidate can report a solution with "100% yield", but the true yield value can be quite different. To reflect the real accuracy, we calculate the yield estimated by 50,000 LHS MC simulations at the same design point. From eqn. (11), we can calculate that with 99% confidence level and $\Delta Y = 0.1\%$, the corresponding yield value of 50,000 LHS simulations is 96%. The results we test are all higher than 96%. Hence, an estimation result from 50,000 LHS simulations is a very reliable approximation of the real yield value for use as a reference result.

Thirdly, the method to measure the efficiency should be decided. The performance of evolutionary algorithms (EA) is affected by the random numbers used in the evolution operators. The CPU times and the yield results have differences between different runs. To address the stochasticity of the results of the evolution process, all experiments are therefore executed 10 times with different random numbers and the results are analyzed and compared statistically showing typical, best and worst performance. In this way, the comparison in terms of accuracy and efficiency is reliable.

B. Test Example 1

The ORDE algorithm is first tested on a fully differential folded-cascode amplifier, shown in Fig. 8, implemented in a 0.35 μm CMOS process with 3.3V power supply. The specifications are gain $A_0 \geq 70\text{dB}$, gain-bandwidt $\text{GBW} \geq 40\text{MHz}$, phase margin

$PM \geq 60^{\circ}$, output swing $OS \geq 4.6V$ and $power \leq 1mW$. There are 13 design variables. The transistor width has a range of 1 μm to 600 μm; the transistor length has a range of 0.35 μm to 5 μm; the biasing current has a range of 10 μA to 200 μA. The total number of the process variation variables is 80, including 15 transistors $\times$ 4 intra-die variables / transistor = 60 intra-die variables (mismatch) and 20 inter-die variables. Statistical information of the process variables has been extracted from the technology information provided by the foundry.

Experiments with the infeasible pruning (IP) +LHS method have been performed using 300 and 500 LHS MC simulations for each feasible candidate. The results of the yield estimate provided by a 50,000 MC simulation analysis at the same final design point and the total number of simulations are analyzed. The statistical results of 10 independent runs are shown in Table 1, Table 2 and Fig. 9.

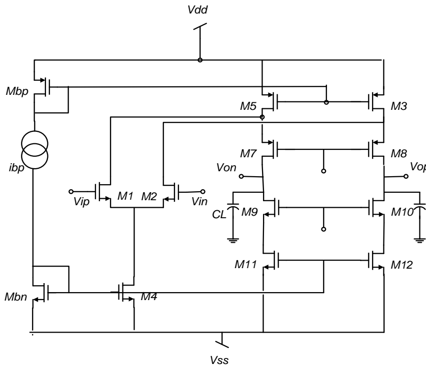

Fig. 8. Fully differential folded-cascode amplifier

Table 1. The yield results (using 50,000 MC simulations) of the solutions obtained by different methods (example 1)

methods	*best*	*worst*	*average*	*variance*
300 simulations (IP + LHS)	98.5%	96.2%	96.8%	0.008%
500 simulations (IP + LHS)	98.7%	96.6%	97.5%	0.007%
ORDE	99.1%	97.3%	98.5%	0.004%

Table 2. Total number of simulations (example 1)

methods	*best*	*worst*	*average*
300 simulations (IP + LHS)	181500	986100	464570
500 simulation (IP + LHS)	357500	3591500	1824520
ORDE	25376	439984	150540

From the inspection of Table 1, the experiment with 500 IP +LHS MC simulations is selected as a benchmark to compare with ORDE. From Fig.9, we can see that the deviations of ORDE from the accurate yield estimate obtained by 50,000 LHS samples are much better than the other methods and the computational cost is much lower. With respect to the number of simulations, shown in Table 2, ORDE costs only 8.25% of the simulations of the infeasible pruning (IP)+LHS method with comparable accuracy. Moreover, in many runs of 300 or 500 simulations with standard DE, the final reported results do not reach the yield requirement, 99%, while 90% of the reported results of ORDE reach 99%. It can be concluded that the random-scale search operator enhances the search ability of DE. The average cost of CPU time of ORDE for this example is about 3 to 4 minutes.

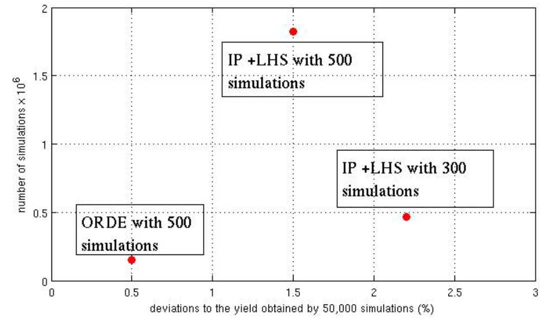

Fig. 9. Comparisons of average yield estimate deviation and number of simulations for different methods for example 1: ORDE clearly has good accuracy and small number of simulations

In the following, a more complex example will be tested and the contribution of OO and the random-scale search operator will be investigated separately.

C. Test Example 2

The ORDE algorithm is now tested on a two-stage fully differential folded-cascode amplifier with common-mode feedback (CMFB), shown in Fig. 10. The circuit is designed in a 90nm CMOS process with 1.2V power supply. The specifications are $A_0 \geq 60\text{dB}$, $\text{GBW} \geq 45\text{MHz}$, $\text{PM} \geq 60^{\circ}$, $\text{OS} \geq 1.9V$, $power \leq 2.5mW$ and $\sqrt{area} \leq 250\mu m$. There exist 21 design variables. The transistor width has a range of 0.12 μm to 800 μm; the transistor length has a range of 0.1 μm to 20 μm; the compensation capacitance has a range of 0.1pF to 50pF; the biasing current has a range of 0.05mA to 50mA. All transistors must be in the saturation region. The total number of process variation variables for this technology is 143, including 24 transistors $\times$ 4 intra-die variables / transistor = 96 intra-die variables and 47 inter-die variables. Statistical information of the process variables was extracted from the technology information provided by the foundry.

Experiments with 300 and 500 simulations for each feasible candidate by the reference IP+LHS method have been done. We separately study the improvement provided by the introduced OO technique and the improvement provided by the random-scale operator. The results of the yield estimation provided by a 50,000 MC simulation analysis at the same final design point and the

total number of simulations are analyzed. The statistical results of 10 independent runs are shown in Table 3, Table 4 and Fig. 11.

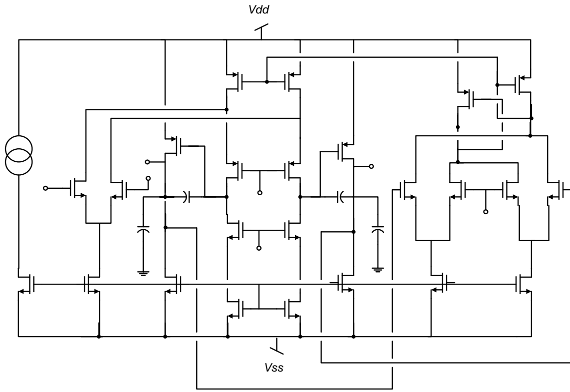

Fig. 10. Two-stage fully differential folded-cascode amplifier

Table 3. The yield results (using 50,000 MC simulations) of the solutions obtained by different methods (example 2)

methods	*best*	*worst*	*average*	*variance*
300 simulations (IP + LHS)	99.0%	97.9%	98.3%	0.002%
500 simulations (IP + LHS)	99.3%	98.2%	98.9%	0.002%
OO+IP+LHS	99.7%	98.1%	98.9%	0.003%
ORDE	99.6%	98.3%	98.9%	0.002%

Table 4. Total number of simulations (example 2)

methods	*best*	*worst*	*average*
300 simulations (IP + LHS)	115500	546900	264130
500 simulations (IP + LHS)	172500	688000	418730
OO+IP+LHS	39828	140537	90209
ORDE	16335	100795	47421

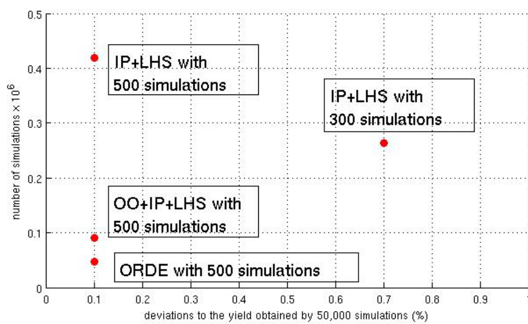

Fig. 11. Comparisons of average yield estimate deviation and number of simulations for different methods for example 2: ORDE clearly has good accuracy and small number of simulations

From the best, worst and average yield values in Table 3, it can be seen that the accuracy with 300 simulations is obviously lower than with 500 simulations. To assess the separate contribution of ordinal optimization (OO) and the random-scale differential evolution operator, two experiments are conducted. The first one only includes OO as well as the IP and LHS techniques. The second experiment corresponds to the use of ORDE (OO and RSDE combined). For statistical characterization, 10 runs of each experiment are performed.

From Fig.11, we can see that the deviations of ORDE from the target value are very close to that of using 500 simulations and the computational cost is much lower. With respect to the number of simulations, shown in Table 4, ORDE costs only 11.32% of the number of simulations of the IP+LHS method with comparable accuracy. These results come from the contribution of both the OO and the random-scale operator. Without the random-scale operator, as can be seen from the result of the OO+IP+LHS method, it spends 21.54% of the simulations of the IP+LHS method. The average CPU time of ORDE for this example is 25 minutes. It can therefore be concluded that ORDE improves the CPU time by an order of magnitude for the same accuracy compared to the improved MC-based method integrating the infeasible pruning and Latin Hypercube sampling techniques.

D. Comparisons to RSM Methods

The advantages and drawbacks of the PSWCD methods have been discussed in Section I. Here, we experimentally compare ORDE with response-surface-based methods.

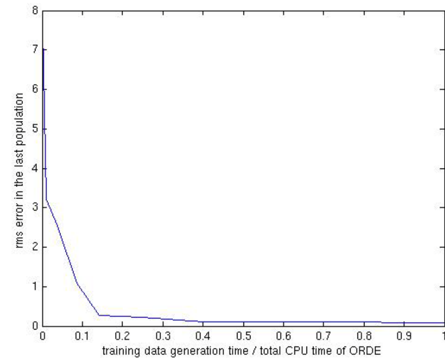

Fig. 12. Result of using NN to approximate yield

In response-surface-based methods, the data obtained from expensive MC simulations at a number of design points is used to generate a regression model able to predict the yield in other design points much cheaper than with a MC simulation, be it at the price of a loss of accuracy. Hence, there exist two trade-offs. The first one is the balance between the accuracy and the complexity of the model. In deep-submicron or nanometer technologies, a sufficiently accurate white-box model may be very complex and makes the regression computationally intractable [8]. The second trade-off is the balance between the accuracy and the number of samples needed to build the model. If sufficient accuracy is required,

sufficient and well distributed training data must be provided. But the computational cost also increases sharply as the density of the samples increases. In the following, we will show the trade-off between accuracy and computational cost of generating the training data using a black-box model.

To assess the loss of accuracy we use the example 1 in Section IV.B and consider a response-surface method based on neural networks (NN), often considered as a powerful regressor [36]. Here, we will use a Backward Propagation NN [36] with 20 neurons in the hidden layer and the Levenberg-Marquardt algorithm [37-38] for training it to approximate the yield. For the source of the training data, we use the data generated during a typical execution of ORDE. It has to be noticed that using these sampling data favors the macro-model as these training data are more significant (because of the key techniques in ORDE, which make the sampling to be more effective) than randomly selected MC simulations, or even those selected by applying only IP and DOE techniques. We will consider the data generated up to a given iteration of ORDE as training data (design parameters as input, and yield values as output), and use the data (yield value) of subsequent iterations as test data to assess the accuracy of the macro-model. At every iteration, we use the data from all previous iterations to train the NN and use this to predict the yield values of the current iteration. The error between the predicted yield values and the real yield values obtained by MC simulations is then calculated and plotted in Fig. 12. The Y axis shows the root-mean-square (RMS) error of the yield predictions. The X axis shows the ratio of the computation time to generate the training data for the NN (by ORDE) to the total computational time of ORDE up to a given iteration. It can be seen that in the beginning the error decreases sharply, but then is levels off. Even when all the data from ORDE are used to train the NN, the RMS error is still 8.06%, while ORDE can provide an error less than 1%. Therefore, response-surface-based methods have difficulties in achieving a sufficient accuracy.

V. Enhancing ORDE for Single Objective Variation-aware Sizing

As can be seen from the previous experiments, the ORDE algorithm meets the goals (be general enough, able to handle both inter-die and intra-die variations, very high accuracy) with significant enhancement on efficiency to make the CPU time practical for yield optimization. On the other hand, in real practice, if the yield requirement can be met, the designers sometimes want to further optimize some objective function (e.g. power or area) while maintaining the target yield, which is shown in eqn. (1.2). To achieve this, we present an extended version of ORDE for single-objective variation-aware sizing.

A. ORDE-based single-objective variation-aware sizing

In single-objective variation-aware sizing, both the objective function f (e.g. power) and the constraint (yield Y) must be considered simultaneously. Hence, we first look at the differences between them. Yield is not a stochastic variable, but we have some uncertainties on its estimation. If we perform an infinite number of MC simulations, yield would have an exact value. The objective function, or specification, is different. If we perform an infinite number of MC simulations, power would still have a probability distribution function, but with an accurate mean and an accurate variance caused by the process variations. Therefore, for yield, we use its expected value to describe it. For the objective function, we use the 3σ value to guarantee the reliability of the expected objective function value, where σ is extracted from the samples.

The main idea of extending ORDE from plain yield optimization to single-objective variation-aware sizing is to add an outer selection procedure considering the objective function value and the yield as constraint. The detailed selection rules are now as follows: for each candidate solution and its corresponding trial counterpart,

(1) if none of them are feasible for nominal process parameters, select the one with the smaller constraint violation;

(2) if one is feasible and the other is infeasible for nominal process parameters, select the feasible one;

(3) if both are feasible for nominal process parameters,

(3.1) if both of them violate the yield constraint, select the one with the smaller yield constraint violation;

(3.2) if one satisfies the yield constraint and the other does not, select the feasible one;

(3.3) if both of them satisfy the yield constraint, select the one with the smaller $\overline{f}(d)+3\sigma_{f(d)}$ (f is the objective function to be minimized, $\overline{f}$ is the mean value).

Using the above selection rule to replace the original selection rule in ORDE, the extended ORDE for single-objective variation-aware sizing can be implemented. We can roughly divide the algorithm into two phases: the yield satisfaction phase and the objective function optimization phase. If we handle the single-objective variation-aware sizing problem as a new task, the yield satisfaction phase will be run first. However, we already have the candidates that satisfy the yield constraint as the plain yield optimization is done first to check if the yield requirement can be met. In this method, we use the last population in the plain yield optimization as the initial population of the extended ORDE to prevent the yield satisfaction phase from running two times.

B. Example

Here we use the example 2 from section IV with the

specifications of $A_0 \geq 60dB$, $GBW \geq 45MHz$, $PM \geq 60^\circ$, $OS \geq 1.9V$, $power \leq 2.5mW$, $\sqrt{area} \leq 250\mu m$ and settling time $\leq 25\mu s$ with 1% error band (this specification needs transient simulation). The yield specification is 99% and the power is the target design objective to be minimized. Five tests with different random seeds are performed. For plain yield optimization without optimizing the power consumption, ORDE satisfies the yield specification, 99%, at a power of 2.38mW. We then use the extended ORDE to minimize power while maintaining yield larger than 99%. The average power value now becomes 1.63mW. The average CPU time is 8919s.

VI. Conclusions

In this paper, the ORDE algorithm has been proposed for efficient yield optimization of analog integrated circuits, considering both inter-die and intra-die process variations. The method is general. ORDE can provide very accurate results with far less computational cost (an order of magnitude smaller) than the MC-based method using infeasible pruning and Latin Hypercube sampling techniques. This improved efficiency makes statistical yield optimization useful in practice. This is achieved by using techniques from computational intelligence, which are as follows: (1) ORDE uses a two-stage yield estimation process with ordinal optimization in the first stage, which determines the simulation effort for each candidate solution "intelligently"; (2) the proposed random-scale operator maintains the diversity and performs combined global and local search, thus enhancing the convergence speed of the search engine; (3) the use of Design of Experiment techniques, infeasible pruning and the selection-based constraint-handling technique also contribute positively to ORDE. Furthermore, ORDE is extended from plain yield optimization to process-variation-aware single-objective analog sizing which has been shown to give good results. Therefore, ORDE and its extended version are both reliable and efficient methods for analog circuit yield optimization, especially for new nanometer technologies with large variability. Moreover, ORDE is based on evolutionary computation and statistical sampling methods, which are very well suited for parallel computation.

Acknowledgment

We sincerely thank Mr. Brecht Machiels, ESAT-MICAS, Katholieke Universiteit Leuven, for valuable discussions.

References

[1] H. Graeb, 2007. "Analog Design Centering and Sizing", Springer.
[2] G. Gielen, et al., 2007. "Automated Synthesis of Complex Analog Circuits", *Proc. of 18th European Conf. on Circuit Theory and Design*, pp. 20-23.
[3] K. S. Eshbaugh, 1992. "Generation of Correlated Parameters for Statistical Circuit Simulation", *IEEE TCAD*, pp. 1198-1206.
[4] M. Buhler, et al., 2006. "DATE 2006 Special Session: DFM/DFY Design for Manufacturability and Yield - influence of process variations in digital, analog and mixed-signal circuit design" *Proc. of DATE*. pp. 387-392.
[5] G. Yu et al., 2008. "Yield-aware Hierarchical Optimization of Large Analog Integrated Circuits", *Proc. of ICCAD*, pp. 79-84.
[6] F. Schenkel, et al., 2001. "Mismatch Analysis and Direct Yield Optimization by Spec-Wise Linearization and Feasibility-Guided Search", *Proc. of DAC*, pp. 858-863.
[7] T. Mukherjee, et al., 2000. "Efficient Handling of Operating Range and Manufacturing Line Variations in Analog Cell Synthesis", *IEEE TCAD*. pp. 825-839.
[8] T. McConaghy, 2008. "Variation-aware Structural Synthesis and Knowledge Extraction of Analog Circuits", Katholieke Universiteit Leuven, Press. (Doctoral thesis)
[9] P. Khademsameni et al., 2002. "Manufacturability Analysis of Analog CMOS ICs through Examination of Multiple Layout Solutions", 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 3-11.
[10] Y. Xu et al., 2009. "Regular Analog/RF Integrated Circuits Design Using Optimization With Recourse Including Ellipsoidal Uncertainty", *IEEE TCAD*, pp. 326-637.
[11] J. Chen et al., "Placement Optimization for Yield Improvement of Switched-Capacitor Analog Integrated Circuits", *IEEE TCAD*, pp. 313-318.
[12] M. Barros et al., 2010. "Analog Circuits Optimization Based on Evolutionary Computation Techniques", *Integration, the VLSI Journal*, pp. 136-155.
[13] R. Schwencker, et al., 2002. "Analog Circuit Sizing using Adaptive Worst-case Parameters Sets", Proc. of DATE. pp. 581-585.
[14] M. Sengupta et al., 2005. "Application-specific Worst Case Corners Using Response Surfaces and Statistical Models", *IEEE TCAD*, pp. 1372-1380.
[15] S. Basu, et al., 2009. "Variation-Aware Macromodeling and Synthesis of Analog Circuits using Spline Center and Range Method and Dynamically Reduced Design Space", *Proc. of 22nd International Conf. on VLSI Design*, pp. 433-438.
[16] A. Mutlu, et al., 2003. "Concurrent Optimization of Process Dependent Variations in Different Circuit Performance Measures", *Proc. of the 2003 International Symposium on Circuits and Systems*, pp. 692-695.
[17] S. Tiwary, et al., 2006. "Generation of Yield-Aware Pareto Surfaces for Hierarchical Circuit Design Space Exploration", *Proc. of DAC*, pp. 31-36.
[18] M. Stein, 1987. "Large Sample Properties of Simulations Using Latin Hypercube Sampling," *Technometrics*, pp. 143-151.
[19] A. Singhee, et al., 2008. "Practical, fast Monte Carlo statistical static timing analysis: Why and how", *Proc. of ICCAD*, pp. 190-195.
[20] A. Singhee, et al., 2009. Novel Algorithms for Fast Statistical Analysis of Scaled Circuits, Springer.
[21] R. Rutenbar et al., 2002. Computer-aided Design of Analog Integrated Circuits and Systems, John Wiley & Sons, Inc. New York.
[22] Y. Ho et al., 2007. Ordinal optimization. Soft optimization for hard problems. Springer.
[23] K. Price et al., 2005. Differential Evolution. A Practical Approach to Global Optimization. Springer, Berlin, Heidelberg, New York.
[24] H. Niederreiter, 1992. "Random Number Generation and

Quasi-Monte Carlo Methods” Philadelphia: SIAM.

[25] C.H. Chen, et al., 2000. “Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization”, *Discrete Event Dynamic Systems: Theory and Applications*, pp. 251-270.

[26] C. Chen et al., 2000. Computing Efforts Allocation for Ordinal Optimization and Discrete Event Simulation. *IEEE Trans. on Automatic Control*, Vol. 45, No. 5, pp. 960-964.

[27] P. Moscato, 1989. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Technical Report 158–79*, Caltech Concurrent Computation Program, California Institute of Technology.

[28] J. Nocedal. 1980. “Updating Quasi-newton Matrices with Limited Storage”, *Mathematics of Computation*, vol 35, pp. 773-782.

[29] S. Russell, et al., 2003. Artificial Intelligence: A Modern Approach (2nd edition), Prentice Hall, pp. 111-114.

[30] J. Lampinen et al, 2000. “On Stagnation of the Differential Evolution Algorithm”, *Proc. of 6th International Mendel Conf. on Soft Computing*, pp. 76-83.

[31] K. Deb, 2000. “An Efficient Constraint Handling Method for Genetic Algorithm”, *Computer Methods in Applied Mechanics and Engineering*, pp. 311-338.

[32] B. Liu et al., 2009. “A Memetic Approach to the Automatic Design of High Performance Analog Integrated Circuits”, *ACM Trans. on Design Automation of Electronics Systems*, 14(3), Article 42.

[33] L. Zielinski et al., 2006. “Yield Enhancement by Means of Evolutionary Computation Techniques”, *ISCAS*, pp. 4631-4634.

[34] J. Swidzinski et al., 1999. “A Novel Approach to Efficient Yield Estimation for Microwave Integrated Circuits”, *42nd Midwest Symposium on Circuits and Systems*, pp. 367-370.

[35] J. Swidzinski et al., 2000. “Nonlinear Statistical Modeling and Yield Estimation Technique for Use in Monte Carlo Simulations”, *IEEE Trans. on Microwave Theory and Techniques*, pp. 2316-2324.

[36] PD Wasserman, 1988. Neural Computing: Theory and Practice. New York: Van Nostrand Reinhold.

[37] K. Levenberg, 1944. “A Method for the Solution of Certain Problems in Least Squares”, *Quart. Appl. Math.* , pp. 164-168.

[38] D. Marquardt, 1963. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”, *SIAM J. Appl. Math.* , pp. 431-441.

Bo Liu was born in Beijing, P. R. China, on September 23, 1984. He received the B.S. degree in electronic engineering from Tsinghua University, P. R. China, in 2008. Since 2008, he is a Ph. D. candidate and is working as a research assistant at the MICAS laboratories of the Katholieke Universiteit Leuven, Belgium, under the supervision of Prof. Dr. Georges Gielen. His research interests lie in design automation methodologies of analog and RF integrated circuits, evolutionary computation, machine learning and fuzzy logic. He has authored or coauthored more than 20 papers in international journals and conference proceedings. He is a reviewer in artificial intelligence and analog design automation fields, such as IEEE Transactions on Evolutionary Computation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Information Sciences (Elsevier) and Integration, the VLSI Journal (Elsevier). He is also a book reviewer of Elsevier and Bentham Science Publishers.

Francisco. V. Fernández got the Physics-Electronics degree from the University of Seville, Spain, in 1988 and his Ph. D. degree in 1992. In 1993, he worked as a postdoctoral research fellow at Katholieke Universiteit Leuven (Belgium). From 1995 to 2009, he was an Associate Professor at the Dept. of Electronics and Electromagnetism of University of Seville, where he was promoted to full professor in 2009. He is also a researcher at IMSE-CNM (CSIC and University of Seville). His research interests lie in the design and design methodologies of analog and mixed-signal circuits. Dr. Fernández has authored or edited three books and has co-authored more than 100 papers in international journals and conferences. Dr. Fernández is currently the Editor-in-Chief of Integration, the VLSI Journal (Elsevier). He regularly serves at the Program Committee of several international conferences. He has also participated as researcher or main researcher in several National and European R&D projects.

Georges G.E. Gielen received his M.Sc. and Ph.D. degrees in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1986 and 1990, respectively. In 1990, he was appointed as a postdoctoral research assistant and visiting lecturer at the department of Electrical Engineering and Computer Science of the University of California, Berkeley. From 1991 to 1993, he was a postdoctoral research assistant of the Belgian National Fund of Scientific Research at the ESAT laboratory of the Katholieke Universiteit Leuven. In 1993, he was appointed assistant professor at the Katholieke Universiteit Leuven, where he was promoted to full professor in 2000.

His research interests are in the design of analog and mixed-signal integrated circuits, and especially in analog and mixed-signal CAD tools and design automation (modeling, simulation and symbolic analysis, analog synthesis, analog layout generation, analog and mixed-signal testing). He is coordinator or partner of several (industrial) research projects in this area. He has authored or coauthored two books and more than 300 papers in edited books, international journals and conference proceedings. He regularly is a member of the Program Committees of international conferences (DAC, ICCAD, ISCAS, DATE, CICC, etc.), and served as General Chair of the DATE conference in 2006 and of the ICCAD conference in 2007. He serves regularly as member of editorial boards of international journals (IEEE Transactions on Circuits and Systems, Springer international journal on Analog Integrated Circuits and Signal Processing, Elsevier Integration). He received the 1995 Best Paper Award in the John Wiley international journal on Circuit Theory and Applications, and was the 1997 Laureate of the Belgian Royal Academy on Sciences, Literature and Arts in the discipline of Engineering. He received the 2000 Alcatel Award from the Belgian National Fund of Scientific Research for his innovative research in telecommunications, and won the DATE 2004 Best Paper Award. He is a Fellow of the IEEE, served as elected member of the Board of Governors of the IEEE Circuits And Systems (CAS) society and as chairman of the IEEE Benelux CAS chapter. He served as the President of the IEEE Circuits And Systems (CAS) Society in 2005.