Ordinary Kriging Metamodel-Assisted Ant Colony Algorithm for Fast Analog Design Optimization

Oghenekarho Okobiah\textsuperscript{1}, Saraju P. Mohanty\textsuperscript{2}, Elias Kougianos\textsuperscript{3}

NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu)\textsuperscript{1,2,3}
Computer Science and Engineering, University of North Texas, USA\textsuperscript{1,2}
Engineering Technology, University of North Texas, USA\textsuperscript{3}

Email-ID: oghenekarhookobiah@my.unt.edu\textsuperscript{1}, saraju.mohanty@unt.edu\textsuperscript{2}, elias.kougianos@unt.edu\textsuperscript{3}.

Abstract—This paper explores an ordinary Kriging based metamodeling technique that allows designers to create a model of a circuit with very good accuracy, while greatly reducing the time required for simulations. Regression and interpolation based methods have been researched extensively and are a commonly used technique for creating metamodels. However, they do not take into account the effect of correlation between design and process parameters, which are critical in the nanoscale regime. Kriging provides an improved metamodeling technique which takes into effect correlation effects during the metamodel generation phase. The ordinary Kriging metamodels are subjected to an Ant Colony Optimization (ACO) algorithm that enables fast optimization of the circuit. This design methodology is evaluated on a sense amplifier circuit as a case study. The results show that the Kriging based metamodels are very accurate and the ACO based algorithm optimizes the sense amplifier precharge time with power consumption as a design constraint in an average time of 3.7 minutes (optimization on the metamodel), compared to 72 hours (optimization on the SPICE netlist).

Keywords-Nano-CMOS, Sense Amplifier, Robust Design, Metamodeling, Kriging Methods

I. INTRODUCTION

Analog simulations use very accurate models and have the ability to accurately estimate performance measures. However, with the scaling of designs in the deep nanometer region and the increase in the level of complexity, exhaustive design space exploration through computer simulation has become more daunting and most often impractical. In nano-CMOS designs, the effects of process variation are increasingly becoming more dominant. These factors make design optimization very difficult and time consuming. Metamodeling has been one researched and applied solution to reduce the time burden of computer simulation while keeping the accuracy to an acceptable level.

Metamodels, by definition, are an approximate description of the performance response of design models. Essentially, metamodels are models of a simulation model (hence the term meta) \cite{1}, \cite{2}. The use of metamodels abstracts the time complexity of analog simulations while capturing in detail the behavior of the design. This gives the designer quick access to a sufficiently accurate design space exploration tool.

Commonly used metamodeling techniques include Response Surface Modeling (RSM), linear and low-order polynomial regression techniques \cite{3}, \cite{4}, \cite{1}, \cite{1}, and neural networks \cite{5}, \cite{6}, \cite{7}. Linear and low-order polynomial regression techniques generally provide a better fit for local neighborhoods but are less accurate for global design spaces \cite{2}, \cite{8}. Due to the oscillatory characteristic of polynomial fits, designs with rapidly changing data points are not well fitted which is the case with nano-CMOS designs \cite{9}. In generating the metamodels, regression techniques assume the errors due to variation across the design space are random and thus equally approximate the error over points on the fitting surface. For many design processes, this error is not random but is correlated with other process and design parameters. In nano-CMOS designs, the correlation can vary significantly across the global design space and even locally, hence making correlation effects a significant factor in accurate metamodeling. Therefore, there is need to implement a metamodel which captures these correlation effects to improve its accuracy.

This paper proposes a Kriging based metamodeling technique that generates accurate metamodels with the error due to correlation taken into consideration. Kriging techniques were originally introduced in the early 1950’s in geostatistical analysis and have been applied to many other fields \cite{10}, \cite{11}, \cite{12} and recently in VLSI \cite{13}, \cite{14}. Kriging based techniques generate interpolating functions for each estimated point using the correlation effect between design points in the local space. Each point response is estimated with a unique set of weights. One major improvement of Kriging over conventional regression is that the estimated response at sample points is the same as the actual response at these points (although it may differ in-between). The generated Kriging metamodel is then subjected to an Ant Colony Optimization (ACO) based algorithm for fast optimization. \cite{15}, \cite{16}, \cite{17}. Originally used for discrete optimization problems, there has been recent research to adapt ACO for continuous functions \cite{16}, \cite{18}, \cite{17}, \cite{19}. The novel contributions of this paper are the application of Kriging to generate metamodels that account for correlation effects in nano-CMOS designs, and the use of ACO-based algorithms for fast optimization.

The rest of this paper is organized as follows. In section II, relevant prior research is summarized. Section III discusses

\textsuperscript{2}This research is supported in part by NSF awards CNS-0854182 and DUE-0942629 and SRC award P10883.
the background of Kriging methods. The proposed algorithm is presented in Section IV. In section V experimental results are presented. Conclusions and directions for future research are presented in Section VI.

II. RELATED PRIOR RESEARCH

The research and application of metamodeling as a design methodology has been proposed in the literature before. A popular metamodeling technique is low-order polynomial regression [12], [3], [4], [1]. In [3], an analysis of different metamodels generated with a number of sampling techniques is presented. While low-order polynomial regression techniques are suitable for global design spaces [9], [2], [5], [20]. Regression techniques average the errors in calculating weights over the design space. This creates an oscillating effect for fast changing data, especially when the autocorrelation error between design points varies significantly. Geometric programming is presented in [20]. This solves convex problems deduced from circuit design equations expressed in polynomial forms. The approximations made in deducing the circuit equations reduce the accuracy, even though they are accurate for local neighborhoods and present poor fit for global design spaces [8], [2], [3], [20]. Regression metamodels generated with a number of sampling techniques [17]. This paper implements an ACO based algorithm.

III. ORDINARY KRIGING METHOD FOR METAMODELING

The fundamental idea behind Kriging methods is that the predicted outputs are weighted averages of sampled data. The weights are unique to each predicted point and are a function of the distance between the point to be predicted and observed points. The weights are chosen so that the prediction variance is minimized [23], [10].

The general expression of a Kriging model is of the following form [22]:

The variogram is used to derive the Kriging weights, \( \lambda_j \). The autocorrelation of the design points is characterized by the covariance function [24]. The weights are chosen so that the Kriging variance is minimized. There are different variations of the Kriging model which include simple, ordinary and universal Kriging. The proposed metamodel in this paper is based on the ordinary Kriging technique which assumes a mean that is constant in the local domain of a predicted point.

In ordinary Kriging techniques, the weights are chosen to minimize the variance under the unbiasedness constraint 

\[ E(z(\hat{x}) - z(x)) = 0. \]

\[ \sum_{j=1}^{n} \lambda_j = 1 \]

Hence the weights are given by the following expression:

\[
\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_n \\
\mu
\end{pmatrix} = \Gamma^{-1}
\begin{pmatrix}
\gamma(e_1, e_0) \\
\vdots \\
\gamma(e_n, e_0) \\
1
\end{pmatrix},
\]

where \( \mu \) is a Lagrange multiplier. \( \Gamma \) is the covariance matrix of the observed points and for ordinary Kriging is given as:

\[
\Gamma =
\begin{pmatrix}
\gamma(e_1, e_1) & \cdots & \gamma(e_1, e_n) & 1 \\
\vdots & \ddots & \vdots & \vdots \\
\gamma(e_n, e_1) & \cdots & \gamma(e_n, e_n) & 1 \\
1 & 1 & 1 & 0
\end{pmatrix}
\]

where \( \gamma(e_1, e_2) = E((Z(e_1) - Z(e_2))^2) \).

Estimation of the correlation between sampled points and a predicted point is done with the semivariogram model. Based on the nature of the observed data points, the empirical model could be fit to either spherical, linear, Gaussian or exponential theoretical models. The smoothness of the predicted points is affected by the theoretical model used. A steeper model reduces the smoothness because it places more weight on closer neighbors. The most common model used is the spherical and is expressed by the following expression:

\[ \gamma(h) = C_0 + C \left( \frac{3h}{2a} - \frac{1}{2} \left( \frac{h}{a} \right)^3 \right) \quad \text{for } 0 < h \leq a, \]

where \( C_0 \) and \( C \) are fitting coefficients and \( a \) is a shape factor.

IV. THE PROPOSED DESIGN FLOW USING KRIGING METAMODELS AND ANT COLONY ALGORITHM

Algorithm 1 shows the complete flow of the proposed design that incorporates ordinary Kriging and the ACO algorithm. A sense amplifier used in conventional DRAMs is used to demonstrate the proposed design optimization methodology.

Ant Colony Optimization (ACO) algorithms are inspired by the foraging behavior of ant species and are metaheuristic random searching algorithms. The majority of ACO algorithms have been applied to discrete combinatorial optimization problems such as routing, scheduling, and timetabling [15], [25]. A classic example in demonstrating the ACO algorithm is the Traveling Salesman Problem (TSP). The basic stages of ACO based algorithms are metaheuristic which means they can be easily modified for application on a wide range of optimization problems [25]. Recently, there has been more
research to extend the application of the ACO algorithms to continuous function problems [18], [16], [17], [19].

The basic characteristics of ACO algorithms include the incremental construction of solutions and the use of pheromone updates to guide point explorations. The basic stages for the ACO metaheuristic are as follows:

1) Initialize variables and set conditions
2) Construct ant nodes
3) Perform local search (optional)
4) Update pheromones

After the initialization step, the algorithm iterates between steps 2-4 until the condition for termination is met.

ACO algorithms for continuous problems differ from discrete optimizations in the selection of ant nodes. In [18], [25], [16], [17], [26], [27] several modifications to ACO for continuous function optimizations have been presented. One major way of adapting the ACO for continuous functions is dividing the solution space into different intervals and having each ant or node search an interval for an optimal solution. The nodes can also directly search the continuous function.

The proposed algorithm in this paper is most closely related to an optimal solution.

The proposed algorithm is shown in Algorithm 2. The details of implementation are included in steps 6 and 7. For each iteration, a new set of ant solutions (feasible design points) is generated. The solutions are updated with an evaporating pheromone. The best solution is however updated with more pheromone thus increasing the probability of the path (node) being traversed by more search ants. The iterations are continued until the termination condition is met which, in this case, is a maximum number of allowed iterations. The speed of convergence of the ACO algorithms can be controlled by the rate of pheromone update. A series of runs show that for this circuit, an average of 100 iterations converge the algorithm to an optimal solution.

The optimization of the Kriging metamodels is done with the proposed ACO base algorithm. For this algorithm (3), a metamodel generated with 500 sample data points is used. The optimization goal is to minimize the precharge time $T_{PC}$ without violating the power constraint. The parameters are initialized for random design points of the NMOS length and width, $L_n$ and $W_n$. The algorithm updates the pheromones by evaporation but remembers the best solution, thereby increasing the probability of more search ants in that direction.

### Algorithm 1 The Proposed Design Flow.

1: Create baseline design.
2: Identify Figures of Merit (FoMs) (verify functionality).
3: Create physical layout.
4: Perform DRC/LVS and RLCK extraction.
5: Identify design parameters and parameterize netlist.
6: Metamodel Generation.
7: Perform Latin HyperCube Sampling to generate design points for metamodel.
8: Generate Krigeing metamodels using mGStat tool.
9: Optimization.
10: while (Optimization objective not met ) do
11:  Perform ACO based algorithm.
12: end while
13: return Optimized Design.

### Algorithm 2 ACO Based Heuristic Algorithm Setup.

1: Initialize number of ants (solutionset)
2: Initialize iteration counter: $counter \leftarrow 0$
3: Start with initial baseline solution ($\hat{SA}_i$) with Ordinary Kriging.
4: Generate metamodel functions for each FoM of ($\hat{SA}_i$) $\hat{T}_{PC}$
5: Consider the objective of interest $T_{PC}$
6: Generate random ant nodes $AL, W_i$, where $i = 1, 2, \ldots, N_{ant}$.
7: Assign initial pheromone, $\tau_i$
8: $counter \leftarrow max_{Iteration}$
9: while ($counter > 0$ ) do
10:  Generate ant solutions $T_{PC}$
11:  Rank solutions ($\hat{SA}_i$) in set from best to worst.
12:  Update pheromone, increase pheromone for best solution and evaporate pheromone for all others
13:  $result \leftarrow T_{PC}$
14:  Generate new ant nodes $AL, W_i$
15:  $counter \leftarrow counter - 1$
16: end while
17: return $result$

### Algorithm 3 ACO Heuristic Algorithm for Sense Amplifier.

1: Initialize parameters.
2: Set termination conditions.
3: Generate random node ants.
4: Perform pheromone update.
5: while (Termination condition not met) do
6:  Generate node ants with pheromone probability.
7:  Update pheromone.
8: end while
9: return $result$. 

$$\tau_i = (1 - \rho) \cdot \tau_i + \rho \cdot \Delta \tau_i,$$

where $\rho$ is the rate of evaporation, and the pheromone $\tau_i$ in the $i$-th iteration is updated only for the best solutions. The criteria and number of best solutions can be decided independently for each optimization problem.

The proposed algorithm is shown in Algorithm 2. The details of implementation are included in steps 6 and 7. For each iteration, a new set of ant solutions (feasible design points) is generated. The solutions are updated with an evaporating pheromone. The best solution is however updated with more pheromone thus increasing the probability of the path (node) being traversed by more search ants. The iterations are continued until the termination condition is met which, in this case, is a maximum number of allowed iterations. The speed of convergence of the ACO algorithms can be controlled by the rate of pheromone update. A series of runs show that for this circuit, an average of 100 iterations converge the algorithm to an optimal solution.
surfaces which are not shown in this paper for brevity. For ordinary Kriging, the sample data points were generated using the Latin Hypercube Sampling (LHS) technique. Three sets of ordinary Kriging metamodels were generated with the mGstat toolbox [29] using 100, 200 and 500 sampling points. For each set, a metamodel was generated for each FoM. Figure 3 shows the surface plots generated through the ordinary Kriging technique.

A statistical summary of the accuracy of the metamodels is shown in Table II. The metamodel predictions are compared to the exhaustive surfaces via the Mean Square Error (MSE), Root Mean Square Error (RMSE), the correlation coefficient $R^2$ and the standard deviation (STD). MSE and RMSE are defined by the following expressions:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2,$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2},$$

where the summation runs over the $N$ points used to generate the metamodels, and $Y_i$ and $\hat{Y}_i$ are the actual and predicted responses, respectively at point $i$.

Table II shows that the metamodels created are quite accurate with very low RMSE values and average $R^2$ values in the range of 0.98–0.99 with the $R^2$ values for $P_{SA}$ a little lower than the other FoMs. As expected, the metamodels generated from 500 LHS sampling points are generally more accurate than the ones generated from 200 or 100 sampling points. The time required for the metamodel generation is 3.69 minutes on average time taken for design optimization using the sense amplifier as the case study circuit is 3.9 minutes. The bulk of the time is consumed in the metamodel generation as the

![Fig. 1. Sense amplifier circuit with sizing for 45nm CMOS.](image1)

![Fig. 2. Sense amplifier functional simulation.](image2)
ACO algorithm converges in an average time of 1.36 sec. The process from design space exploration to optimization is reduced by a factor of approximately $10^3 \times$. The final (optimized) layout is shown in Fig. 4.

### VI. Conclusions and Future Research

This paper presented a new design methodology that combines Kriging techniques for metamodel generation and ACO based algorithms. Kriging provides an improved method for metamodeling that takes into account the correlation effects between points in the design and process space for nano-CMOS designs. It also improves the accuracy of the metamodels. It has very low RMSE when compared to exhaustive surfaces but it takes longer time to generate than conventional methods. ACO based algorithms have also been applied to the metamodels for fast optimizations. The precharge time $T_{PC}$ is improved by 65.77% within the power constraints. This methodology has been demonstrated using two design parameters. In future research, this methodology will be extended to more parameters and multi-objective optimization goals.
TABLE II
STATISTICAL ANALYSIS OF THE KRIGING PREDICTED RESPONSES

<table>
<thead>
<tr>
<th>FoMs</th>
<th>Samples</th>
<th>Ordinary Kriging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>Precharge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>2.20 × 10⁻¹⁰</td>
<td>5.23 × 10⁻¹⁰</td>
</tr>
<tr>
<td>RMSE</td>
<td>4.69 × 10⁻¹⁰</td>
<td>2.29 × 10⁻¹⁰</td>
</tr>
<tr>
<td>R²</td>
<td>0.9650</td>
<td>0.9917</td>
</tr>
<tr>
<td>STD</td>
<td>4.32 × 10⁻¹⁰</td>
<td>2.03 × 10⁻¹⁰</td>
</tr>
<tr>
<td>Sense Delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>4.22 × 10⁻¹⁰</td>
<td>1.16 × 10⁻⁹</td>
</tr>
<tr>
<td>RMSE</td>
<td>2.05 × 10⁻¹⁰</td>
<td>1.08 × 10⁻⁹</td>
</tr>
<tr>
<td>R²</td>
<td>0.9529</td>
<td>0.9871</td>
</tr>
<tr>
<td>STD</td>
<td>1.89 × 10⁻¹⁰</td>
<td>3.99 × 10⁻¹⁰</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>1.84 × 10⁻⁹</td>
<td>1.08 × 10⁻⁹</td>
</tr>
<tr>
<td>RMSE</td>
<td>3.14 × 10⁻⁹</td>
<td>3.29 × 10⁻⁹</td>
</tr>
<tr>
<td>R²</td>
<td>0.8384</td>
<td>0.8525</td>
</tr>
<tr>
<td>STD</td>
<td>1.19 × 10⁻⁹</td>
<td>9.47 × 10⁻¹⁰</td>
</tr>
<tr>
<td>Sense Margin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>1.12 × 10⁻⁹</td>
<td>3.41 × 10⁻⁹</td>
</tr>
<tr>
<td>RMSE</td>
<td>3.35 × 10⁻⁹</td>
<td>1.85 × 10⁻⁹</td>
</tr>
<tr>
<td>R²</td>
<td>0.9804</td>
<td>0.9940</td>
</tr>
<tr>
<td>STD</td>
<td>2.98 × 10⁻⁹</td>
<td>1.62 × 10⁻⁹</td>
</tr>
</tbody>
</table>

REFERENCES


