# Macromodeling of Analog Circuits for Hierarchical Circuit Design

Jianfeng Shao and Ramesh Harjani
Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455, USA

**Abstract– Hierarchy plays a significant role in the design of digital and analog circuits. At each level of the hierarchy it becomes essential to evaluate if a sub-block design is feasible and if so which design style is the best candidate for the particular problem. This paper proposes a general methodology for evaluating the feasibility and the performance of sub-blocks at all levels of the hierarchy. A modified simplicial approximation technique is used to generate the feasibility macromodel and a layered volume-slicing methodology with radial basis functions is used to generate the performance macromodel. However, due to lack of space, only details of the performance macromodeling techniques are included. Macromodels are developed and verified for analog blocks at three different levels of hierarchy (current mirror, opamp, and A/D converter).**

## 1 Introduction

As feature sizes shrink even further, an increasing percentage of IC's will have analog circuit designs in them, stressing the need for analog design automation. Hierarchy has played a significant role in the design of digital and more recently in analog circuits [1]. In digital design, extremely large designs are routinely designed by breaking up the task into smaller and smaller sub-tasks, i.e., divide and conquer. Hierarchy helps to hide the lower-level details and also helps to focus attention on more tractable sub-tasks. Even though it may be informal and less well accepted, hierarchy does exist in analog circuit design practice. For example, an analog to digital (A/D) converter is not designed at the transistor level right from the start.

An example of hierarchy for an A/D converter is shown in Figure 1. There are a number of different kinds of A/D converters, e.g, flash, successive approximation, sigma-delta, etc, Each of these is called a design style [1]. The hierarchy for a first-order sigma-delta is shown in this figure. The converter has as subcomponents, an integrator, a comparator, a 1-bit D/A and a digital low pass filter. There are different design styles, or ways of implementing, each one of these components. For example, one of many possible choices for the integrator is shown. One of the subcomponents within the integrator is an opamp. Once again, there are many different design styles for the opamp, e.g., simple one-stage, Miller-compensated two-stage, folded-cascode, etc [1].

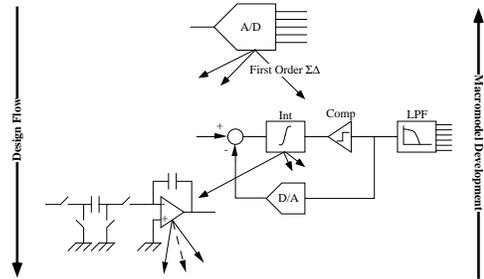Design proceeds down such a hierarchy. At each level



Figure 1: A/D converter hierarchical decomposition

of the hierarchy we are presented with a number of candidate design styles for each functional block. Each of these design styles provides the same functionality but provides different performance tradeoffs. As part of the design we need to select the best candidate for the job at each level of the hierarchy. Two decisions need to be made during this selection process. First, we need to evaluate which design styles are *feasible*, i.e, can be designed to meet input specifications. Second, we need to evaluate which design style provides the best *performance*. A second aspect of design, i.e., the translation of specifications from one level of the hierarchy to the next lower level, has been described elsewhere [1, 13].

One possible solution to the selection problem is to exhaustively try out all the options and then select the best candidate among them. However, the design time increases exponentially, as the number of levels in the hierarchy and/or the branching factor increase. Unfortunately, performance of analog circuits are strongly tied to the bottom level transistor behavior. Therefore, without an appropriate macromodel, the performance or feasibility of a design at a level cannot be evaluated without traveling to the very bottom of the hierarchy. Typical branching factors in a real design situation can be fairly large at all except the lowest level of the hierarchy. For example, there are probably twenty different design styles for operational amplifiers, i.e., a branching factor of twenty. As a solution to this problem we propose a numerical macromodeling solution that generates macromodels *a priori*. To accurately predict performance and feasibility, these macromodels are built bottom-up throughout the hierarchy.
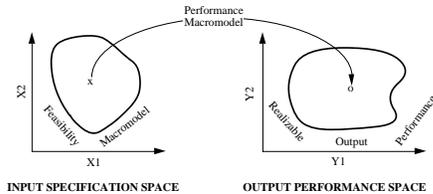
Figure 2: Feasibility/performance relationship

To ensure that the methodology is general, i.e, can be applied at all levels of the hierarchy and can be applied to any circuit block, it is essential that the methodology be abstract and not depend on the implementation details of any of the functional blocks. To this effect it is essential that the methodology use a set of general basis functions to perform macromodeling and general techniques for experimental design. Two kinds of macromodels are necessary. One model is necessary to check if a design is feasible, *the feasibility macromodel*, and the second model is necessary to predict the behavior, *the performance macromodel*. The relationship between the feasibility macromodel and performance macromodel is shown in Figure 2. Performance macromodels are mappings from the feasible input specification space to the realizable performance space. At each level of the hierarchy each design style has a single feasibility macromodel and a set of performance macromodels, one for each performance metric.

The primary advantage of developing hierarchical macromodels is that the macromodels need only be built once. They can then be used during design time to accommodate a complete top-down design. Therefore, the performance or feasibility of a block at any level can be determined by evaluating the macromodels at that level rather than having to travel to the very bottom of the hierarchy. Only when we need to complete the circuit design do we travel down the hierarchy. However, at each level of the hierarchy, since we have feasibility macromodels, we know which design style can be designed to meet specifications and which cannot. Likewise, since we have performance macromodels for each of these feasible design styles, we can choose the best candidate.

OASYS [1] is an example of analog synthesis system that uses similar abstractions to efficiently design analog circuits. We shall use the OASYS system to evaluate our macromodeling approach. However, the methodology being described is general and can be used with any other hierarchical design system. The modeling technique has been fine tuned to handle analog circuits; however, the approach is general and is applicable to both analog and digital circuits. This feature makes it particularly suitable for mixed-signal designs.

This paper proposes a general methodology for macromodeling using radial basis functions. Since a large number of simulations are required and increases

exponentially in higher dimensions, systematic design plans are used to reduce the number of experiments. In our methodology, the significance of input variables is evaluated using experimental runs. Next variable screening and variable grouping is performed based on the significance of each input on individual models. To further minimize the cost of simulations we develop an adaptive volume slicing technique during regression analysis to dynamically perform experimental runs.

The paper is organized as follows. Section 2 reviews the previous work in this area. Section 3 presents the macromodeling methodology that has been developed. In section 4 we provide examples with analog circuit blocks to illustrate the viability of our approach. Finally, in section 5 we provide some conclusions.

## 2  Review of Previous Work

In an attempt to evaluate past research it is instructive to provide a more formal definition for the macromodel. In general, the relationship between the $i$th response and input variables and its macromodel can be represented by

$$y_i = f_i(x_1, x_2, \ldots, x_n) \qquad (1)$$
$$\hat{y}_i = \hat{f}_i(x_1, x_2, \ldots, x_n) \qquad (2)$$

where $x_i, (0 \leq i \leq n)$, is the $i$th input variable, $y_i$ is the $i$th response, $\hat{y}_i$ is the approximation of the $i$th response, $f_i$ is the unknown relationship between $y_i$ and $\{x_1, x_2, \ldots, x_n\}$, $\hat{f}_i$ is the macromodel of $f_i$, and $n$ is the number of input variables in the $i$th macromodel. As we shall see later, the number of variables in $y_i$ and $\hat{y}_i$ do not have to be the same.

A number of macromodeling techniques have been developed for design centering of integrated circuits [2, 3, 4]. Design centering deals with choosing a nominal design which maximizes the fabrication yield. The design approach to improve yield seeks to inscribe the largest norm body, usually a hyperellipsoid, in the boundary by moving the center of the norm body, called the nominal point [2, 3]. In [2, 3, 4] an approximation of the feasibility region is generated using simplicial approximation. We use a variation of this technique to generate a macromodel for the feasibility region [5].

In [6, 7, 8] macromodels are built by performing regression analysis on empirical polynomials with the aid of well designed experiments. The approach in [9] employs a quasi-physical model form for a MOS process using a predetermined set of variables and is thus not sufficiently general for our macromodeling requirements. In the MULREG program [10], a multiple-layered regression scheme is used with a large number of variables. In this approach, the polynomial regression model is synthesized layer by layer using a number of low order polynomials. The data points are generated randomly which, unfortunately, leads to extremely large and non-optimal experimental runs. The approach in [4] builds

a second-order polynomial macromodel using regression analysis with a fractional factorial experiment plan. However, as with other polynomial based approaches, the number of experimental runs required increases exponentially with the number of input variables. Additionally, factorial design plans require that the maximum complexity of the polynomial be known before experiments can be designed. However, to provide a general solution, we cannot make any *a priori* assumptions about the complexity of the response surface.

In the next section, we develop our solution that uses radial basis functions in combination with a dynamic experimental design strategy to provide general macromodels efficiently.

## 3 Macromodeling Approach

As mentioned earlier, macromodeling is desirable when the computation cost of generating data points is unacceptably high. As we have seen before, the complexity of the design space can grow rapidly. Both of these stress the need for efficient macromodeling techniques. Macromodel construction requires a number of simulations to gather the data points. Therefore, the number of experimental runs can be used as a metric of the cost of macromodeling. However, if too few experimental runs are performed, the accuracy of the resultant macromodel is sacrificed. Hence, there is a direct tradeoff between the cost of generating the model versus the accuracy of the model. Experiments can be generated statically or dynamically. Static experiments are either designed using factorial design techniques [11] or done manually using previous knowledge of the target space. On the other hand, dynamic techniques use no previous knowledge of the target space, but adapt to provide the best tradeoff between cost and accuracy.

In the following paragraphs, we present the four primary aspects of our macromodeling approach. These are:

**Feasibility region definition:** The circuit design problem is defined by specifying input variables and output responses. The specifications include the domain of the input variables and constraints on the output responses. This in effect defines the feasibility region for the design tool. To find the boundary points of the feasibility surface we use one of two search algorithms. Once an adequate number of data points are gathered, a macromodel can be built for the feasibility region. The two methods that are used to perform the search are radial binary search and vertical binary search [5]. Radial binary search requires that the feasibility surface be convex in all dimensions while the vertical binary search requires that the surface be convex in only one dimension. Figure 3 shows the feasibility region for a two-stage opamp computed by using the vertical binary search procedure. Interested readers are referred to [5] for further details.
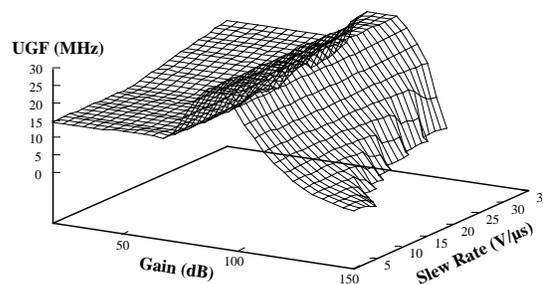


Figure 3: 2-Stage opamp feasibility region

**Experiment design:** Experiment design techniques are employed to build an appropriate experiment plan. The use of an appropriate experiment plan results in substantial savings in the number of experimental runs. In our approach, we use both static and dynamic experimental design techniques. We use a static factorial design technique to measure variable significance and we use a dynamic technique called *volume slicing* to generate the data points for regression. During volume slicing, we dynamically increase the number of experimental runs where the response surface is more complex and decrease the number of experiments where the response surface is smooth.

**Variable screening and grouping:** Through systematic experimental runs the significance of each variable, i.e., its effect on the output response, is estimated. The variables below a certain threshold level are neglected. Selected variables are further grouped into layers with the more significant variables in the upper layers. This classification reflects the varied influences of different variables on the output response and has a significant impact on the complexity of the modeling technique for high dimensions.

**Regression analysis:** Having once performed the necessary experimental runs and gathered the data points, we now use these data points to calculate the coefficients of the macromodel. Additionally, the accuracy of the resultant macromodel also needs to be verified. To dynamically collect the data points, we perform the regression analysis at two levels. A local regression analysis procedure is called recursively for a local area until a certain accuracy is obtained. After all the data points have been obtained, a global regression analysis is then performed to obtain an approximation for the entire surface. The adaptive volume slicing technique was developed in conjunction with local regression analysis to dynamically generate the necessary experiments.

### 3.1 Static experiment design

Experiment design is a sampling strategy. Properly designed experiments can result in substantial savings in the number of experimental runs. Since one of our

primary concerns is to minimize the number of experimental runs, systematic experiment design techniques are employed to achieve this goal. The resultant experiment plan is used for variable screening and variable grouping. So, as not to waste experimental runs we reuse the results of these experiments for regression analysis as well. To maintain consistency, we use a two-level fractional factorial plan to design experiments for variable screening and for variable grouping.

In a two-level fractional factorial plan, each variable takes two values. In our application, they are selected to be within the feasible input domain. Assuming that each variable has been normalized such that its low and high values are $-1$ and $+1$ respectively, then a full two-level factorial plan with $n$ variables requires $2^n$ experimental runs for all possible combinations [11]. The number of experimental runs can be reduced substantially by using a $2_{III}^{n-p}$ fractional factorial plan. Interested readers are referred to [4, 11] for more details.

## 3.2 Variable Screening and Grouping

As discussed earlier, not all the input variables have the same effect on the output response. Therefore, it is possible to identify a subset of the input variables which are more significant than the others. These significant variables are then used in macromodel construction while the other variables are discarded from consideration. Even among the selected significant variables, the degree of influence on the response is different. To further reduce the complexity of the regression analysis, we group the significant input variables into layers.

A slightly modified $2_{III}^{n-p}$ fractional factorial plan is used for variable screening. For each input variable $x_i$ we define the following quantities:

- Main effect, $v_i = \frac{1}{2}\{v_{i+} - v_{i-}\} = \frac{1}{n_r}\sum_{k=1}^{n_r} s_{ik} \cdot y_k$
- High deviation, $d_i^h$, from the nominal (i.e., the center point) response, $d_i^h = \frac{2}{n_r}\sum_{k \in K_i} y_k - y_c$
- Low deviation, $d_i^l$, from the nominal response, $d_i^l = \frac{2}{n_r}\sum_{k \in K_i'} y_k - y_c$

where $n_r$ is the number of experimental runs, $y_k$ is the response value of the $k$th run, $s_{ik}$ is the sign of $x_k$ on the $k$th run, $K_i$ is the set of run indices when $x_i$ is $+1$, $K_i'$ is the set of run indices when $x_i$ is $-1$, and $y_c$ is the response value of the center point.

Using these quantities, a statistical significance test is performed to determine if the corresponding input variable is significant. Further we can define

$$\delta_i^v = \frac{|v_i|}{\hat{\sigma}_v}, \qquad \delta_i^h = \frac{|v_i^h|}{\hat{\sigma}_h}, \qquad \delta_i^l = \frac{|v_i^l|}{\hat{\sigma}_l} \qquad (3)$$

where $\hat{\sigma}_v, \hat{\sigma}_h$, and $\hat{\sigma}_l$ are the estimates of the standard derivation of $v_i, d_i^h$, and $d_i^l$, respectively.

A variable is considered to be significant if $|\delta_i| > t_{\alpha/2, n-1}$ is true for at least two of the three variable,

$v_i, d_i^h$, and $d_i^l$. Here $\alpha$ is the desired level of significance and $t_{\alpha/2, n-1}$ is obtained from the $t$-distribution table. Variables that are considered insignificant are neglected from further discussion. Moreover, we rank the remaining variables by their significance in terms of $\delta_i^v, \delta_i^h$, and $\delta_i^l$. We group them into layers according to their ranks. The more significant a variable is, the higher is the group that it is placed in.

## 3.3 Regression Analysis

Having identified the set of significant input variables and grouped them into layers, we now proceed to construct the macromodel for the response in terms of these variables. In general, a macromodel of a response $y$ is given as described in eq. (2), and the actual response $y$ can be written as, $y = \hat{y} + \epsilon$ where $\epsilon$ is the random error due to the effects of the insignificant variables we neglected earlier.

The form of the function $f(.)$, in eq. (2), can be selected from some known function classes. For example, in most previous macromodeling efforts, the polynomial function is used. However, models for a large number of variables requires excessive computation work and the resulting model is too complicated to be used. Additionally, polynomial function forms require some *a priori* knowledge of the system in order to limit the order of the polynomial. Therefore, the polynomial function form isn't well suited for a general macromodeling methodology. We employ radial basis functions (RBFs) as the model form in our approach. The linear-in-the-variable structure of RBFs provides the generality we require and the dimensionality of the problem space has little effect on the complexity of the resulting macromodel. Furthermore, no *a priori* knowledge of the problem domain is required for complete model development.

Rather than sampling randomly, we have developed an adaptive volume slicing technique to perform the experimental runs and gather data points. We perform additional experimental runs only in areas where more details are required. To this end, we dynamically adapt the spacing to minimize the total number of experimental runs. However, before discussing volume slicing we present some details about radial basis functions.

### 3.3.1 Radial Basis Functions

Radial basis functions have been used extensively to approximate multi-dimensional spaces [12]. The form of the radial basis function for an $n$-variable input space with a scalar output response $y$ is given by,

$$y = \lambda_0 + \sum_i^{n_r} \lambda_i \phi(\| \vec{x} - \vec{x}_i \|) \qquad (4)$$

where $\phi(.)$ is a function from $R^n$ to $R$, $\| . \|$ denotes the Euclidean norm, $\vec{x} \in R^n$, $\lambda_i$, $(0 \leq i \leq n_r)$, are the weights or the parameters, $\vec{x}_i \in R^n$, $(1 \leq i \leq n_r)$,

are the RBF centers and $n_r$ is the number of the centers. The function form $\phi(.)$ is selected before hand. The choice of the function form doesn't affect the average performance, but particular forms are better suited for different conditions. The centers $\vec{x}_i$ are points in $n$-dimensional space where experimental runs are performed. The centers could potentially be distributed uniformly within the input domain. However, substantial savings in terms of experimental runs can be garnered by selecting appropriate centers. These centers can be selected using *a priori* knowledge of the design space or by using the knowledge garnered from previous experimental runs. The second of these two approaches, dynamic experiment design, was selected because of it generality. We call this experiment design technique adaptive volume slicing and use it to select the RBF centers dynamically.

We note, from eq. (4), that the dimension $n$ of the input space has little influence on the function, because the Euclidean norm $\parallel . \parallel$ is a scalar. This gives the RBF an advantage over other model forms where the number of terms in the function depends upon the dimension of the input space.

Typical choices for $\phi(.)$ are the thin-plate-spline function, $\phi(r) = r^2 \log r$, and the gaussian function, $\phi(r) = \exp(-r^2/\beta^2)$,. The approximation capabilities of the RBF approximation is directly related into its localization properties. The localization property implies that the contribution of $\vec{x}$ in the input domain, $D$, which are far away from the center of the function $\phi_i$, is much less than those in the vicinity of the center. For a set of given centers, the global function of these radial basis functions corresponding to the contribution from each center is formed into eq. (4). It is easy to see that $\phi(r) \rightarrow 0$, as $r \rightarrow \infty$. For these choices, the RBF approximation has good localization properties. A judicious choice of the RBF centers, or experiments, is extremely important and leads to our adaptive volume slicing strategy.

### 3.3.2 Adaptive Volume Slicing

The allocation of the RBF centers has a direct impact on the performance of the RBF approximation. Additionally, we also wish to minimize the number of experimental runs. Both these constraints require a efficient method to determine when and where to place the RBF centers. We call our method of locating the RBF centers as adaptive volume slicing. In adaptive volume slicing we choose the intersection points of surfaces in the input domain as the RBF centers, and slice the volume only when the accuracy of the approximation is not sufficient. To illustrate this methodology, we present an example in two dimensions.

The adaptive volume slicing method is compatible with the experiment design techniques used for variable screening and grouping. Therefore, we are able to reuse the experimental runs generated for variable screening
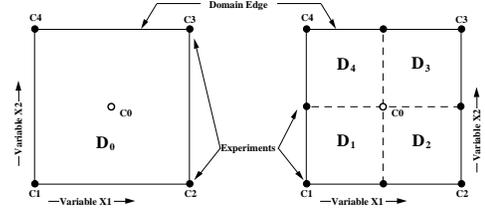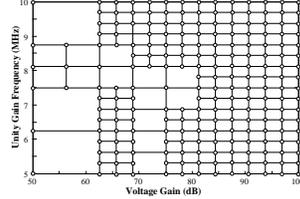


Figure 4: Before and after volume slicing
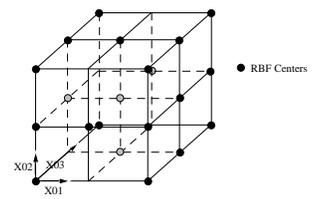


Figure 5: RBF center distribution



Figure 6: Volume slicing in 3-d

and grouping. To maintain generality we adopt a regular structure for the data points, i.e, the method is recursive. The left-side of Figure 4 shows a unit of the input domain $D_0$, with input variables $x_1$ and $x_2$ and the output response $y$. To start with, the intersection points $c_i$, $1 \leq i \leq 4$ (the four nodes of the square), are chosen as the RBF centers. From eq. (4), we have

$$y = \sum_{j=1}^{4} \lambda_j \phi_j (\parallel \vec{x} - \vec{c}_j \parallel) \qquad (5)$$

which is the approximation of the response of the input domain $D_0$. The RBF coefficients $\lambda_j$ are solved using the values at $c_i$'s. To check if the accuracy of this approximation is good enough, we perform an experimental run at the center point $c_0$ and generate the response value $y_c$. From eq. (5), we have the estimated value of the response at $c_0$, $y_e$. If the criterion in eq. (6) is satisfied, then the RBF model of the four centers is a good approximation for the domain $D_0$. Otherwise, we slice the square such that it results in the figure on the right-side of Figure 4. For each subarea $D_i$, $1 \leq i \leq 4$, we repeat the above procedure for $D_0$. The recursive procedure is terminated when the criterion given in eq. (6) is satisfied for each subarea.

$$\left| \frac{y_c - y_e}{y_c} \right| \leq \epsilon \qquad (6)$$

The parameter $\epsilon$, in eq. (6) can be varied for the desired level of accuracy. As a second example, we use our volume slicing method to generate RBF centers for an opamp circuit design. The distribution of the RBF centers are shown in Figure 5. In this figure, each circle represents an RBF center, and the lines themselves represent domain edges. We slice input domain units
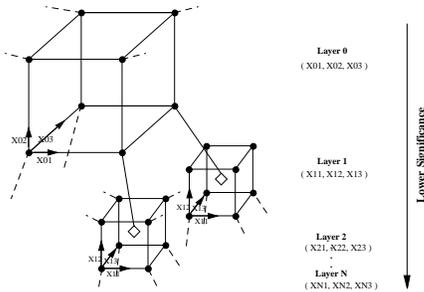
Figure 7: The layered structure for input variables

and add new centers (i.e., experimental runs) only when it is necessary to further explore the details of the response surface. By doing so we not only increase the performance of the RBF approximation but we also minimize the number of necessary experimental runs. Since these improvements are made without any earlier knowledge about the shape of the response surface our methodology proves to be extremely general. The simple volume slicing technique mentioned here can easily be extended to 3-dimensions. However, for higher dimensions ($n > 3$) the basic methodology has to be modified due to the complexity of the data structures that are generated. This modification and generalization of the volume slicing approach to $n$-dimensions is described next.

### 3.3.3   Layered Volume Slicing

To extend the volume slicing method to 3-dimensions, a cubic unit is used instead of a square unit as shown in Figure 6. When slicing a cube, we add 19 additional center points, instead of 5 in the 2-dimensional case. The number of RBF centers that have to be added dynamically grows exponentially for higher dimensions. To reduce this problem, we use the significance of input variables. We group the variables into layers, as discussed in subsection 3.2. Each layer has at most three variables. The most significant variables are assigned the highest layers. The layering for a single unit cube is shown in Figure 7.

The volume slicing method for each layer is the same as that described in subsection 3.3.2. Let the top layer be $L_0$, then the procedure for layered volume slicing is as follows:

1. *Call the procedure for layer $L_0$.*
2. *For layer $L_i$, perform the volume slicing procedure in the input domain of $x_{i1}, x_{i2}$ and $x_{i3}$: obtain the value of the response at each node, derive the local RBF model for the local domain, run an extra simulation at the center of the domain to check if eq. (6) is satisfied. In the case that eq. (6) is not satisfied, the local domain is sliced as described in subsection 3.3.2.*
   *For each RBF center, to obtain the value of the response, set the values of $x_{i1}, x_{i2}$ and $x_{i3}$, and call*

*the volume slicing procedure for layer $L_{i+1}$.*
*Return the average response value of the center point.*

We make the approximation that the response value of a RBF center in layer $L_i$ is equal to the average of the response values of layer $L_{i+1}$ with the input variables in layer $L_i$ set to the values at that center. Since we have grouped the most significant variables on the top layer, the response values of RBF centers in a lower layer have less influence. So, the resulting error from making this approximation is small. It is important to note that this approximation only affects the experiment design phase. It has no effect on the final regression. For the final regression all the input variables are included at the same level. We have found that this variable grouping strategy works well in practice. We show an example of this in section 4.

### 3.3.4   RBFs vs. Polynomials

We now compare the effects of using RBFs instead of polynomials as the function form in the macromodel. The number of RBF centers is determined by the size of our data points. Increasing the dimension of input space does not directly affect the number of regressors (RBF centers). On the other hand, for polynomials of $n$ variables, the number of regressors is determined by $m = \sum_{i=0}^{l} m_i$ where $l$ is the polynomial degree and

$$m_0 = 1, \; m_i = m_{i-1}(n_r + i - 1)/i, \qquad 1 \leq i \leq l$$

It is obvious that the number of regressors $m$ increases exponentially as $l$ increases. Therefore, in practice, $l$ must be restricted. This is the reason that, in most polynomial approximations, only the second order polynomial is considered. Although second order polynomial approximations work well for some cases, it usually requires some knowledge of the response surface. Moreover, second order polynomials do not provide a general solution and cannot be used for higher order surfaces. The RBF approach provides a more general solution and can be used for any dimension response surface.

## 4   Macromodeling Results

In this section, we applying our macromodeling methodology on a few circuit examples. We illustrate the versatility of the approach by apply the modeling technique to circuits at different levels of the hierarchy. Macromodeling results for a CMOS current mirror, with particular emphasis on the feasibility region, have appeared in [5]. The macromodels that were developed were fairly accurate. The maximum error was less than 3 percent. For our first example here we apply the modeling technique to a two-stage Miller-compensated CMOS opamp and a one-stage OTA [1]. For our second example, we apply our modeling technique to a first-order sigma-delta A/D converter. As mentioned

| Variable | Definition | Range |
|---|---|---|
| Gain | voltage gain | 40—100 dB |
| UGF | bandwidth | 0.2—30 MHz |
| Slew | slew rate | 0.5—30 V/$\mu$s |
| $C_{ld}$ | load | 0.1—50 pF |
| Power | supply current | 1.0—5.0 mA |
| $V_o^{max}$ | max. $V_{out}$ | 0.5—2.25 V |
| $V_o^{min}$ | min. $V_{out}$ | −2.25—−0.5 V |
| Phase | phase margin | $30^o$—$75^o$ |

Table 1: The input variables for the opamp

| Variable | Definition | $v_i$ | $d_i^l$ | $d_i^h$ |
|---|---|---|---|---|
| Gain | $x_1$ | 29.157 | 30.560 | 27.754 |
| UGF | $x_2$ | 12.270 | 10.867 | 13.672 |
| Slew | $x_3$ | 14.312 | 12.910 | 15.715 |
| $C_{ld}$ | $x_1 \times x_2$ | 4.825 | 6.227 | 3.422 |
| Power | $x_1 \times x_3$ | 1.785 | 0.382 | 3.187 |
| Phase | $x_2 \times x_3$ | 8.377 | 9.779 | 6.974 |
| $V_o^{max}$ | $x_1 \times x_2 \times x_3$ | 10.920 | 12.322 | 9.517 |

Table 2: Two-stage opamp variable significance

earlier, we treat each of these designs as block boxes, only the interfaces are visible.

## Example I. Opamp Macromodels

The opamp uses the current mirror and other functional blocks in its design. We investigate two design styles: a two-stage opamp and a one-stage OTA. The subset of input variables considered for our example are listed in Table 1. We have selected the active area as the response to be monitored.

For our experiments we set $V_o^{max}$ and $V_o^{min}$ to have the same magnitude but opposite sign. We, therefore, only have seven input variables. Using a $2_{III}^{7-4}$ design plan, we obtain the measurements of $v_i$, $d_i^h$, and $d_i^l$ as shown in Table 2. We use the significance criterion

| Gain dB | UGF MHz | Slew V/$\mu$s | Area$_e$ $\mu M^2$ | Area$_o$ $\mu M^2$ | $\epsilon$ |
|---|---|---|---|---|---|
| 50.866 | 5.086 | 6.069 | 11700 | 11390 | 0.0257 |
| 50.866 | 8.086 | 6.069 | 13100 | 12660 | 0.0330 |
| 65.866 | 8.086 | 6.069 | 13700 | 14890 | 0.0872 |
| 50.866 | 6.586 | 7.269 | 12300 | 12320 | 0.0017 |
| 65.866 | 6.586 | 7.269 | 12900 | 11980 | 0.0706 |
| 50.866 | 5.086 | 8.469 | 11600 | 11610 | 0.0010 |
| 65.866 | 5.086 | 8.469 | 12100 | 12070 | 0.0019 |
| 50.866 | 8.086 | 8.469 | 13100 | 12550 | 0.0413 |
| 65.866 | 8.086 | 8.469 | 13700 | 14890 | 0.0872 |

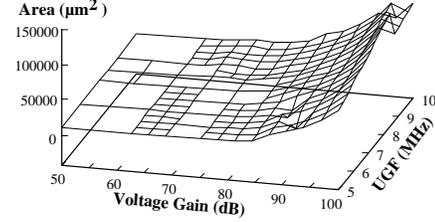Table 3: 2-stage opamp macromodeling results



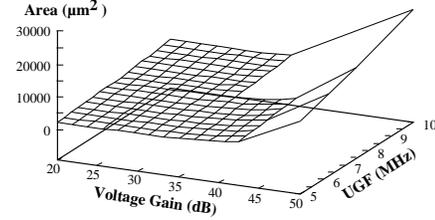Figure 8: The 2-stage opamp performance surface



Figure 9: The OTA performance surface

developed in section 3.2 to discard and group the input variables. The variable *power* is discarded, and the variables, *gain, ugf*, and *slew* are grouped into Layer 0, the variables, $C_{ld}$, *phase*, and $V_o^{max}$ are grouped into Layer 1. Using the layered volume slicing procedure we obtain the response surface for the selected input domain. Figure 8 shows the response surface where all variables except the *gain* and the *ugf* are fixed. The variables are not fixed in the model. However, they were fixed for the figure because it is not possible to show data for dimensions greater than three. Table 3 shows the results of the macromodel for the opamp. For the results shown in this table the $C_{ld}$, $V_o^{max}$, and phase are fixed. The errors here, though slightly larger than the current mirror, are still small.

The performance surface for the one-stage OTA is modeled in a similar manner and is shown in Figure 9.

## Example II. $\Sigma\Delta$ Converter Macromodel

As shown in Figure 1, the sigma-delta converter includes an integrator, a comparator, and a digital LPF. The feasibility macromodel of the sigma-delta converter can be built through the design hierarchy. Given a set of specifications for the sigma-delta converter; resolution $n$ and bandwidth, $f_0$, the design is feasible if and only if the specifications translated for each sub-block lie in their feasible design regions. Macromodels for the integrator, the comparator, and the LPF can be built by using a similar procedure. Using the feasibility macromodels for the two-stage opamp and the one-stage OTA and the comparator, etc., we obtain the feasibility curve in 2-d for the sigma-delta converter in Figure 10. The performance surface has also been built using our methodology, but is not included here for lack of space.
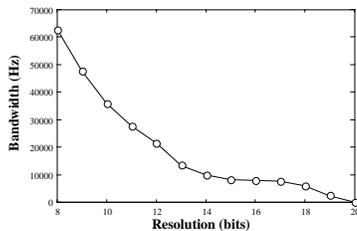
Figure 10: Feasibility region for a sigma-delta converter

## Methodology Performance

The rationale for developing hierarchical macromodels was to substantially reduce the design time. The savings in the design time for a simple design may not be significant because of the limited number of design styles at each level included in the OASYS system, i.e., small branching factor. Recall that there are over twenty design styles for an opamp while only two of them have been implemented in OASYS. However, even for the limited branching factor, the savings can be substantial when running multiple experiments as may be necessary for design space exploration [1]. Additionally, the rationale for developing a systematic methodology to select experimental runs was to reduce the effort necessary to develop the macromodel. The savings in design time and in the number of experiments is illustrated by the following examples.

**A.** To develop the performance macromodel for the OTA, we performed 196 experiments in 524.34 seconds CPU time(2.675 seconds/data point). Having once built the macromodel, we were able to perform 667 experiments in 20.46 seconds CPU time (0.031 seconds/data point). This is a savings of 98.84%.

**B.** In macromodeling, traditional experimental design techniques use no knowledge of the statistical distribution of previous regressors and thus uniformly distributes the regressors. To obtain the same accuracy for the two-stage opamp performance macromodel, the most optimum traditional method would generate 289 data points, while the volume slicing method only required 204 data points (see Figures 5. This is a savings of approximately 30%. However, care should be taken when interpreting this number. Since no *a priori* knowledge can be assumed, a designer would normally either overestimate the number of experimental runs, i.e., our savings would be substantially more. Or she/he would underestimate the number of experimental runs, i.e, a less accurate model.

**C.** Design space exploration without a macromodel is generated by performing a design run for each point on the design surface. However, with the macromodel, the design surface is already approximated by the macromodel. Therefore, design space exploration only involves evaluating the macromodel a number of times.

As illustrated by (**A**), the savings can be substantial. However, more importantly, the substantial reduction in real time implies that design space exploration becomes practical even for systems that take substantial design time [13].

## 5 Conclusions

In this paper, we have presented a general macromodeling approach for hierarchical circuit design. The validity of our approach was tested by generating macromodels at different hierarchical levels. Fractional factorial experiment experiment design techniques were used to measure the significance of input variables. Variable screening and grouping techniques were employed to select and organize the input variables based upon their influence on the output response. An adaptive volume slicing technique was used during regression analysis to dynamically distribute regressors such that the number of experimental runs is minimized. The RBF approximation is well suited to our methodology because of its locality and linear-in-parameter structure. We have found that our methodology works well for analog circuit blocks. Though not explicitly tested, our approach should be directly applicable to digital circuit blocks as well as no knowledge of the underlying circuit is assumed.

## References

[1] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS a framework for analog circuit syntheis," *IEEE Tran. CAD*, Dec. 1989.

[2] S. W. Director and G. D. Hachtel, "The simplicial approach to design centering," *IEEE Tran. CAS*, Jul. 1977.

[3] R. K. Brayton, G. D. Hachtel, and A. S. Vincentelli, "A survey of optimization techniques for integrated-circuit design," *Proceedings of IEEE*, Oct. 1981.

[4] K. K. Low, *A Methodology for Statistical Integrated Circuit Design*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1989.

[5] J. Shao and R. Harjani, "Feasibility region modeling of analog circuits for hierarchical circuit design," in *IEEE MWSCS*, 1994.

[6] Y. Aoki, H. Masuda, S. Shimada, and S. Sato, "A new design centering methodology for vlsi device development," *IEEE Tran. CAD*, May 1987.

[7] A. R. Alvarez, et. al., "Application of statistical design and response surface methods to computer-aided VLSI device design," *IEEE Tran. CAD*, Feb. 1988.

[8] T. Yu, S. Kang, I. Hajj, and T. Trick, "Statistical performance modeling and parametric yeild estimation of MOS VLSI," *IEEE Tran. CAD*, Nov. 1987.

[9] P. Cox, et. al., "Statistical modeling for efficient parametric yield estimation of MOS VLSI circuits," *IEEE Tran. ED*, Feb 1985.

[10] C. Shyamsundar, "Mulreg – user's manual," technical report, Carnegie Mellon University, 1986.

[11] G. Box, W. Hunter, and J. Hunter, *Statistics for Experimenters: an Introduction to Design Data Analysis and Model Building*. John Wiley, 1978.

[12] M. Powell, "Radial basis functions for multivariable integration: A review," in *IMA Conference on Algorithm and Approximations and Data*, RMCS, 1985.

[13] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "ASTRX/OBLX: Tools for rapid synthesis of high-performance analog circuits," in *ACM/IEEE DAC*, 1994.