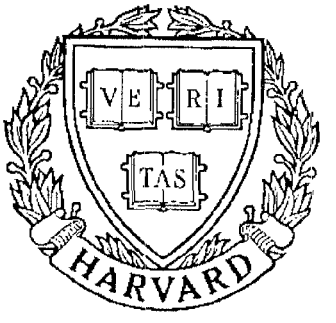
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Yield Optimization for Integrated Circuit

by H. Tian and L. Milor

TR 92-103

Yield Optimization for Integrated Circuit

Hao Tian and Linda Milor

ABSTRACT

As integrated circuits become increasingly complex, geometries smaller and smaller, it has become more difficult to achieve acceptable manufacturing yield. Four approaches to yield optimization are surveyed and compared. These include simplicial approximation, statistical modeling, the yield gradient approach, and minimax optimization.

September 23, 1992

1. Introduction

As integrated circuits become increasingly complex, geometries smaller and smaller, it is become more and more difficult to achieve acceptable manufacturing yield, even if the nominal design fulfills all design constraints. The manufacturing yield is composed of two factors: technological yield, and the parametric yield. The former is the result of catastrophic technological failure; the latter is the result of the sensitivity of circuit performance to IC device parameter variations, caused by unavoidable variation of the manufacturing conditions from device to device or from chip to chip.

There are currently many methods to optimize the parametric yield of a circuit. The algorithm used in these methods can be grouped in the following: Monte Carlo Method [1], linear approximation [2], quadratic approximation [3], simplicial approximation [4], statistic method [5] [6], and yield gradient [2]. Monte Carlo method, although crude and expensive, has been the technique primarily used for estimating the parametric yield in the past. However, continuing research in the field of the IC optimization has yielded significant gain in the efficiency of the optimization algorithm.

In this paper, I will briefly introduce the simplicial approximation algorithm, statistical algorithm, linear algorithm and yield gradient algorithm, and explain how they work.

2.1 The Simplicial Approximation approach

The simplicial approximation approach is a simple, efficient method for design centering that constructs an approximation to the feasible region. Specifically the simplicial approximation is based on approximating the boundary, ∂R, of the feasible region, R, which is an n-parameter design space by a polyhedron made up of n-dimensional simplices. Approximation is necessary since ∂R is generally known only in terms of nonlinear inequality constraints, that express acceptable circuit performance which depends upon the design parameters.

Below is the introduction of the algorithm presented in [4].

Assume the circuit under consideration obeys a set of differential-algebraic equations of the form

$$g(\xi, \dot{\xi}, x, t) = 0, \quad 0 \le t \le t_f \qquad (1)$$

where ξ is a vector of circuit performances (i.e., node voltages, branch voltages, and branch currents) and x is an n-vector of statistically varying design parameters (e.g. the lengths and widths of the transistors) with a joint probability density function (jpdf) $f_x(x, u_x, \Sigma_x)$, where u_x is the mean value of x and Σ_x is the covariance matrix.

Let X denote the n-dimensional parameter space so that $x \in X$. Thus the circuit is defined by a nominal set of parameter values $x^0 \in X$. The circuit with parameter values x is acceptable if the solution $\xi(t)$ of Eq.(1) obeys a given vector of n_c constraints,

$$\Phi_i(x) = \int_0^{t_f} \Phi_i(\xi, \dot{\xi}, x, t)\, dt \le 0, \quad i=1,2,..,n_c \qquad (2)$$

The set of the constraints defines the region of acceptability R.

$$R = \{ x \mid \phi_i(x) \le 0, \text{ for all } i \in [1,2,..,n_c] \text{ and } x^l \le x \le x^u \} \qquad (3)$$

Where x^l and x^u are the lower and upper limit of x. Note R is closed and is bounded if the limits x^l and x^u are finite. In what follows, we assume that R is convex. The boundary of R defined by ∂R ,

$$\partial R = \{x \mid \phi(x) \le 0,\ x^l \le x \le x^u, \text{and } \phi_i = 0, \text{or } x_i = x_i^l, \text{or } x_i = x_i^u, \text{for some } i\} \qquad (4)$$

The yield associated with some nominal design point, $u_x = X_0$ is

$$Y = prob(x \in R) = \iint_R \cdots \int f_x(x, X_0, \Sigma_x)\, dx \qquad (5)$$

The design centering can now be stated as that of choosing the nominal value X_0 of the designable parameter, so that the yield is maximum for a given distribution $f_x(X, X_0, \Sigma_x)$.

The simplicial approximation method is based on approximating the boundary ∂R of the feasible region R by a polyhedron, i.e. by the union of those portions of a set of n-dimensional hyperplanes which lie inside ∂R or on it. The procedure begins by determine any $m \ge n+1$ points, $x_1, x_2, \ldots, x_m$ on the boundary ∂R, usually m is taken either n+1, or 2n. (One way to find 2n points is by first finding a feasible set of designable parameters inside R and then performing one dimensional line searches in the positive and negative coordinate directions.) The initial feasible point can be obtained either by designer insight or through the use of an optimization program. Given a set of m points on R, the convex hull is then constructed by a set of m_H inequalities

$$\eta_k^T x \le b_k, \quad k=1,2,\ldots,m_H \qquad (6)$$

where η is a unit"outward pointing" vector normal to the kth hyperplane defined by,

$$\eta_k^T x = b_k, \quad k = 1,2,\ldots m_H \qquad (7)$$

and b_k is a measure of the distance of the kth hyperplane from the origin.

Given the first approximation to ∂R we can find a first estimate of the design center by determining the center of the largest hypersphere which can be inscribed inside the polyhedron of m_H hyperplanes described by Eq.(7) . The center of the largest hypersphere can be found easily using a linear programming approach. First recognize that the distance

from a point x^* inside the polytope to the kth hyperplane is

$$d_k = \eta_k^T x^* - b_k \qquad (8)$$

Therefore the center and radius of the largest hypersphere can be found by determining the maximum value of r and the point x^* for which

$$\eta_k^T x^* + r \le b_k, \quad k=1,2,\ldots,m_H \qquad (9)$$

The second step in the design centering procedure is to improve the simplicial approximation to ∂R and then update the design center. This step is accomplished by determining which of the m_H "faces" of the polyhedron that are tangent to the largest inscribed hypersphere is the "largest". The largest tangential face (which is defined as the face in which the largest (n-1) dimensional hypersphere may be inscribed) is of particular interest since it is the one which is most likely to be the poorest approximation to ∂R .

Suppose we want to find the largest hypersphere inscribed in the jth face of the polyhedron. The center of this hypersphere, denoted by x_j^*, must be on the jth hyperplane, so that

$$\eta_j^T x^* = b_j \qquad (10)$$

Now let $\eta_j^\perp$ denote a unit vector perpendicular to η_j, thus $\eta_j^\perp$ lies in the jth hyperplane and the surface of a hypersphere of radius r_j centered at x_j^* which lies in the jth hyperplane is described by

$$x_j^* + r_j \eta_j^\perp$$

The largest such hypersphere is the one with as larger an r_j as possible subject to the constraint that all points on the surface of this hypersphere lie within the approximate polyhedron, i.e., they satisfy the constraints

$$\eta_k^T (x_j^* + r_j \eta_j^\perp) \le b_k, \quad for k=1,2,\ldots m_H, k \neq j \qquad (11)$$

which can be shown to be equivalent to the constraints

$$\eta_j^T x_j^* + r_j \sin\phi_{jk} \le b_k \qquad (12)$$

where ϕ_{jk} is the angle between η_k and η_j. Thus the linear program to be solved is to determine x^* and maximize r subject to Eq.(10) and Eq. (12).

The proposed design centering procedure, illustrated for two dimensional is as follows.

Step 1 Determine a set of m $\ge$ n+1 points on the boundary, ∂R.

Step 2 Find the convex hull of these points and use this polyhedron as the initial approximation to ∂R. Set k = 0.

Step 3 Inscribe the largest hypersphere in the approximating polyhedron and take as the first estimate of the design center.

Step 4 Find the "midpoint" of the largest face in the polyhedron which is tangent to the inscribed hypersphere.

Step 5 Find a new boundary point on ∂R by searching along the outward normal of the largest face found in the step (4) extending from the "midpoint" of the face.

Step 6 Inflate the polyhedron by forming the convex hull of all previous points plus the new points generated in step (5).

Step 7 Find the center x_c^k and radius r_I^k of the largest hypersphere inscribed in the new polyhedron found in step (6). Set k = k+1, Go to step (5).

Step (5) in the above procedure is the key since it involves a line search along a given direction. Each step in this search involve first solving the circuit equation Eq. (1) and then evaluating the constraints Eq. (2) to see which, if any, are violated. The search ends when points x_B is located for which

$$\phi_j(x_B) = 0, \quad \phi_i(x_B) \leq 0, \quad i \neq j \qquad (13)$$

The above procedure will rapidly generate a sequence of points whose convex hull forms an interior polyhedron, ∂R_I, which approximate to ∂R. Step (7), the process of inscribing the largest hypersphere in ∂R_I can be used as the mean of monitoring the convergence of $\partial R_I \rightarrow \partial R$. if R is convex, as per our assumption, the sequence of the centers $\{x_c^k\}$ will converge to the center of the largest hypersphere that can be inscribed in ∂R. And the associated radii, $\{r_I^k\}$, will converge to the radius of the largest inscribed hypersphere. The sequence is considered to have converged when

$$|r_I^{k+1} - r_I^k| \leq \varepsilon_R r_R^k + \varepsilon_A \qquad (14)$$

where ε_R and ε_A are given relative and absolute convergence parameters.

Note that each interior approximation ∂R_I^k may be associated with a yield

$$Y_I^k = \iint_{R_I^k} \cdots \int f_x(x, u_x, \Sigma_x)\, dx \qquad (15)$$

where R_I^k is the volume in n-space bounded by ∂R_I^k, i.e., the kth-interior approximation to R.

$$Y_I^{k-1} < Y_I^k < Y \qquad (16)$$

where Y is computed over the true feasible region R. That is, the interior simplicial approximation gives the lower bound on the yield.

The basic premise about the computational effort required to use the above procedure is that the procedure is to be used in the situations where the number of statistically varying design parameters is small compared to the number of algebraic-differential equations that need to be solved in order to determine the circuit behavior. For these cases, the dimensionality of the linear program to be solved in the above procedure is relatively small when compared to the dimensionality of the circuit simulation problem. Thus the time spent in generating and inflating the convex hull is small when compare to the time involved in performing the transient simulation of large nonlinear circuit.

2.2 The Statistical Method

A major cost in statistical analysis occurs in repeated system simulation as the system parameters are varied. To reduce the cost, the system performance can be approximated by the regression model [5],[6],[7]. This model is then used to predict the performance variation and estimate the parametric yield. The papers [6], [7] have used linear regression for performance modeling and also have provided guidelines for deriving and validating the regression models. In [6]'s implementation, an average mean-squared error criterion is used to select an optimal set of circuit simulations. The simulation performances are then fitted to the regression models, and statistical F-tests are used to validate the adequacy of the fitted models. The derived regression models are then used to generate the performance distribution, from which the worst-case performance and parametric yield can be estimated. In [5]'s implementation a set of polynomial regression equations are constructed according to the required circuit performance. These simple polynomial equations then replace the circuit simulation in the Monte Carlo algorithm for computing the parametric yield.

Below is the brief introduction of the statistical design presented in [5].

Given a set of data, let $A \subset R^n$ be the region of acceptable performance, for which the circuit satisfies all specifications. Then we defined the function $z(x)$ as

$$z(x) = \begin{cases} 1 & x \in A \\ 0 & otherwise \end{cases} \qquad (17)$$

The parameter yield is then

$$p(x) = \int_{R^n} z(x)\, f(x)\, dx \qquad (18)$$

where $f(x)$ is the probability density function of parameters.
The integral can be approximated by the Monte Carlo analysis,

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} z(x_i) \qquad (19)$$

where N parameters, x_i, are generated according to $f(x)$, and P is approximated by $\hat{p}$. The accuracy of the approximation $\hat{p}$ can be estimated by its variance,

$$V(\hat{P}) = \frac{1}{N-1}\hat{P}(1-\hat{P}) \qquad (20)$$

Now the computational problem is to find the $z(x)$. One way of finding the $z(x)$ is to evaluate all the specifications at each x_i to determine whether or not x_i is in A. This method require a large number of circuit simulation and therefore it can be very expensive.

Another way is to use the regression model. In most cases, the unknown performance function can be approximated by polynomial equations

$$y = X\beta + \varepsilon \qquad (21)$$

where X contains the terms of polynomial, (e.g. x_1, x_2, x_3^2 ...), β is a vector of coefficients, y is the performance, and ε is an error.It is assumed that $E(\varepsilon)=0$ and $Cov(\varepsilon)=\sigma^2 I$, where I is the identity matrix. The coefficients of the equations are determined by minimizing the residual sum of the squares.

The appropriate form of Y is unknown and an initial assumption on its terms should be made. Initially only the first order terms are candidates, but as the terms are added to the models, the candidate terms include all the polynomial terms that are one order higher than those in the equations. The problem with this approach is that the number of candidate terms increases rapidly with the dimension. In [5]'s implementation, terms are chosen with the largest correlation with the residual and added to the equations one at a time. The statistic R^2_{press} is used to decide when to stop adding terms. (R^2_{press} is intuitively the fraction of variation in the data that is explained in the equation.) The equation that is chosen as the best for the data is the one where R^2_{press} is maximum.

To compute the variance due to the inaccuracy of the regression equation, the residuals, $\varepsilon=y-X\beta$, must be normally distributed (we say the residuals are normally distributed if they lie within the confidence bounds of skew and kurtosis). The mean and standard deviation are sufficient statistics to characterize a normal distribution. Therefore based on these statistics, the distribution of other statistics can be derived. To check if the residuals appear to be normally distributed, skew and kurtosis is used. For a finite set of points, the distribution of skew and kurtosis are derived and 90% confidence bounded are computed. Usually the residuals do not initially lie in the confidence bound. Weighted regression is then used to make the residuals normally distributed. For weighted regression, the same polynomial model is postulated as usual,

except $Cov(\varepsilon) = \sigma^2 V$, where V is symmetric matrix.

In the application of computing parametric yield, it is necessary to determine the boundary of acceptable region A . By choosing an appropriate choice of the symmetric matrix V, which can be used to weight more heavily points that are close to the boundary of A, the regression equations can be made to be more accurate close to the boundary of A. The V that has be used is a diagonal matrix with diagonal elements

$$V_{ii} = 1.0 + \frac{c|y(x_i) - s|}{|s|} \qquad (22)$$

where $c > 0$ is an arbitrary constant, it can be chosen so that the residuals appear to be normally distributed. s is the value of the performance specification, $y(x_i)$ is the observation at x_i, and || denotes the absolute value.

Once the regression equation is determined, parametric yield can be computed, and it is possible to compute its variance. But since the regression equations have a random error, ε, it is not possible to determine if $x \in A$ exactly; $z(x)$ is also a random variable. It is only possible to estimate the expectation of P:

$$E(P) = \int_{R^n} E(z(x))\, f(x)\, dx \qquad (23)$$

which can be approximated using the Monte Carlo analysis

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} E(z(x_i)) \qquad (24)$$

Note that once the regression equations are determined, N is not dependent on the number of circuit simulation, and computing $\hat{p}$ is computational inexpensive.

At each x_i, in order to evaluate $z(x_i)$, first evaluate $y(x_i) = X\beta$ and its variance for each specification. Second, given that residuals are normally distributed, compute the probability, $P_j(x_i)$, that a specification j is satisfied. Then if a circuit at arbitrary x_i, needs to satisfy m specifications, compute the probability that the circuit satisfy all m specifications:

$$P(x_i) = \prod_{j=1}^{m} P_j(x_i) \qquad (25)$$

Therefore

$$E(z(x_i)) = P(x_i) \qquad (26)$$

The variance due to Monte Carlo estimate is

$$\frac{1}{N-1} \hat{p}(1-\hat{p}) \qquad (27)$$

and the variance due to the inaccuracy in regression equations is

$$\int_{R^a} V(z(x_i))\, f(x)\, dx = \frac{1}{N}\sum_{i=1}^{N} V(z(x_i)) \qquad (28)$$

where

$$V(z(x_i)) = P(x_i)\,(1-P(x_i)) \qquad (29)$$

Therefore

$$V(\hat{P}) = \frac{1}{N-1}\hat{P}(1-\hat{P}) + \frac{1}{N}\sum_{i=1}^{N} V(z(x_i)) \qquad (30)$$

The actual data generation has two stages: in the first, rough approximations are made of all functions over their entire range, and in the second stage, the more critical functions are approximated more accurately near the boundary A.

The algorithm is as follow:

```
INPUT V (the desired variance)
For each specification {
      While R²press < 0.9 {
            Generate parameters using Latin Hypercube
            experimental design and simulate
            Fit the regression equation
      }
}
Compute parametric yield and its variance using Eq.(24) & Eq.(30)
While variance is too high {
      Determine specification with the largest contribution to variance
      For each of these specifications {
            Add more parameter near the boundary A and simulate
            Fit the regression equation
      }
      Compute the parametric yield and its variance
}
OUTPUT parametric yield
```

From the experimental results of the above algorithm, the speedup over Monte Carlo method decreases as the number of parameters increases. This is because the possible number of the terms in the polynomial equations increase rapidly with the dimension of the problem. This indicates that polynomial regression is most effective with the low dimension problems. On the other hand, the large number of specifications favors the statistical

model since each of the specifications is handled separately.

2.3 The Yield Gradient Approach

This approach is based on the quasi-Newton technique and employs the yield gradient for optimizing the circuit yield. The discussion for optimization below is based on [2].

In order to determine the yield, one must have a statistical model for the underlying random variations in the process. The model used here assumes that the performance variations in a MOSFET digital circuit are mostly dependent upon only four statistical variables which are independent. These four variables are all MOSFET related and are: length reduction L_r, width reduction W_r, oxide capacitance C_{ox}, and flatband voltage V_{fb}. The important issue of this model is that it utilizes only a few independent statistical variations which cause most of the performance variation. Because the number of variances is small, a deterministic approach to statistical analyses may now be taken. Each performance or constraint function is approximated linearly with respect to these four variables. Five base points vectors are chosen in the statistical variables and five circuit simulations are performed. From these results all the linear approximations to the constraint functions are computed. A set of hyperplane equations

$$a_j^T x + c_j \le b_j, \quad j = 1, \ldots, N_h \qquad (31)$$

results from these linear approximations when combined with the specification bounds for the performances. Here x is the vector of four statistical variables and a_j and c_j are the coefficients that are computed, and the specification bounds are given by the b_j terms. These equations describe a polytope region in $\mathbf{R}^4$ space which is approximation of the yield body (acceptability region), denoted by $\mathbf{M}$. This region is simply the set of all x which satisfy Eq.(31). The yield approximation can now be easily computed from

$$Y = \int_{\mathbf{M}} p(x)\, dx \qquad (32)$$

where $p(x)$ is the probability density function. Assume that no two hyperplane equations are identical, and also assume that no hyperplane is parallel to $e_1 = (1,0,0,0)^T$; that is $a_j^T e_1 \neq 0, \forall j$. If this is not true, one can always rotate the coordinate system such that the condition will be true for the first new unit coordinate.

As the design parameters w are varied this change is manifested in the yield via the constraints boundaries of $\mathbf{M}$ moving. It is necessary to formulate the yield integral Eq.(32) in terms of these constraint boundaries in order to derive the gradient, and thus

the integral can be expressed as

$$Y = \int_3 \int_{f_l(\hat{x},w)}^{f_u(\hat{x},w)} p(x)\, dx_1 d\hat{x} \qquad (33)$$

where $\hat{x}$ is $(x_2, x_3, x_4)^T$. The bounds on the inner integration are given by the general lower and upper bounding functions f_l and f_u. Taking any point x in $\mathbf{M}$ and, moving in the direction of the first coordinate, one will find an upper intersection point and a lower intersection point because $\mathbf{M}$ is bounded, and e_1 is not parallel to any hyperplane. The bounding functions are continuous, $\forall \hat{x} \in \mathbf{R}^3$, and are section-wise linear in $\hat{x}$. Let $\mathbf{M}^P$ be the projection of $\mathbf{M}$ onto $\mathbf{R}^3$, defined by dropping the first coordinate; then let

$$f_u(\hat{x},w) = f_l(\hat{x},w), \quad \forall \hat{x} \notin \mathbf{M}^P \qquad (34)$$

and for $\hat{x} \in \mathbf{M}$, one has the following definitions:

$$f_u(\hat{x},w) = \begin{cases} u_1(\hat{x},w), & \hat{x} \in A_1 \\ \vdots & \\ u_{N_u}(\hat{x},w), & \hat{x} \in A_{N_u} \end{cases} \qquad (35)$$

$$f_l(\hat{x},w) = \begin{cases} l_1(\hat{x},w), & \hat{x} \in B_1 \\ \vdots & \\ l_{N_l}(\hat{x},w), & \hat{x} \in B_{N_l} \end{cases} \qquad (36)$$

Here the A_i and B_i entities are closed, bounded polytope in $\mathbf{R}^3$, and are actually the faces of $\mathbf{M}$ projected onto $\mathbf{M}^P$. Any intersection between the polytope in Eq.(35) or Eq.(36) is another polytope in $\mathbf{R}^2, \mathbf{R}^1$, or $\mathbf{R}^0$. Each u_i or l_i corresponds to a particular hyperplane and has the general form

$$u_i(\hat{x},w) = \frac{a_{2j}x_2 + a_{3j}x_3 + a_{4j}x_4 + c_j - b_j}{a_{1j}} \qquad (37)$$

where the coefficients a_{ij} and c_j are functions of w.

Now differentiate Eq.(33) to obtain

$$\nabla_w Y = \int_3 \nabla_w \int_{f_l(\hat{x},w)}^{f_u(\hat{x},w)} p(x)\, dx_1 d\hat{x} \qquad (38)$$

which reduce to

$$\nabla_w Y = \int_3 \{\nabla_w f_u(\hat{x},w)\, p(f_u(\hat{x},w), \hat{x}) - \nabla_w f_l(\hat{x},w)\, p(f_l(\hat{x},w), \hat{x})\} d\hat{x} \qquad (39)$$

Substituting in the expressions for f_u and f_l one obtains

$$\nabla_w Y = \sum_{i=1}^{N_u} \int_{A_j} \nabla_w u_i(\hat{x},w)\, p(u_i(\hat{x},w), \hat{x})\, d\hat{x} - \sum_{i=1}^{N_l} \int_{B_i} \nabla_w l_i(\hat{x},w)\, p(l_i(\hat{x},w), \hat{x})\, d\hat{x} \qquad (40)$$

For brevity the gradient equations Eq.(40) can also be expressed as

$$\nabla_w Y = \sum_{i=1}^{N_f} \int_{C_j} \nabla_w f_i(\hat{x}, w)\, p(f_i(\hat{x}, w), \hat{x})\, s_i d\hat{x} \qquad (41)$$

that is, f_i is generic for u_i or l_i, likewise for C_j, and s_i is either 1 or -1 depending on the orientation of the face.

It is from the above expression that the yield gradient is computed. In order to carry this out, the gradient of the bounding functions u_i and l_i must be obtained; but from Eq.(37) it is seen that one needs the gradient of the hyperplane coefficients. As mentioned earlier, these coefficients are obtained by fitting the performance from a set of circuit simulations to a linear model for each specification. This results in a set of linear equations

$$H\begin{pmatrix} a_j \\ C_j \end{pmatrix} = p_j \qquad (42)$$

which are solved for the coefficients of the jth hyperplane. Here the matrix H is determined by the value of the statistical variables used in the simulations, and p_j is the vector of simulated performances. Differentiating this linear equation results in

$$H\begin{pmatrix} \partial a_j/\partial w \\ \partial c_j/\partial w \end{pmatrix} = \frac{\partial p_j}{\partial w} \qquad (43)$$

which can be solved to obtain the coefficient gradient. The performance sensitivities, $\partial p_j/\partial w$, can be obtained from the circuit simulation transient sensitivities.

To compute the yield gradient in Eq.(41), the integration domain need to be determined. A construction algorithm to determine the bounding polytope faces of the yield body is discussed in [2].

In order to optimize the circuit yield from the yield gradient, the quasi-Newton procedure have been used. The yield optimization problem is formulated as a constrained maximization problem with bounds on the parameters. To avoid the high computational cost and obtain the gradient whenever the yield is computed, [2] uses a quasi-Nowton procedure which employs the BFGS update formula to approximate the Hessian matrix and bypass the line search. From [8], [9], it is known that this technique can perform equally well without the line search. The solution of the linear equation formed by the Hessian approximation and the gradient provides a search direction, but only one step along this direction is ever taken. A simple algorithm is discussed in [2] to determine the step taken along this direction.

2.4 The Linear Approach - Minimax Optimization

The concept of this algorithm is to expand the yield body by pushing the boundary

hyperplanes away from the center of the statistical distribution. That is, away from the mean. Note that, the statistical distribution is fixed and changing the designed parameter only cause the hyperplanes to move. Intuitively, one can see how such a procedure should increase the yield from Fig.1, where two zero mean statistical variables are used. This formulation is not directly a solution to the yield optimization problem, but it works exceptionally well in practice. Its advantages include its ability to start from very poor initial solutions, the fact that it does not require the computation of the yield gradient, and its usage of circuit sensitivity information.

The discussion below is based on [2].

The statistical analysis is the same as the gradient approach. Four independent statistical variables L_r, W_r, C_{ox}, V_{fb}, are used. Five base point vectors are chosen from the statistical variables and five circuit simulations are performed. From these results, all the linear approximations to the constraint functions are computed. A set of hyperplane equations

$$h_j^T y + v_j \le b_j \qquad (44)$$

results from the linear approximations when combined with the specification bounds for the performance. Here y is the vector of four statistical variables, h_j, v_j are coefficients that are computed, and the specification bounds are given by the b_j terms. The distance from the origin to the hyperplane is defined as

$$d_j = -z_j = \frac{b_j - v_j}{\|h_j\|_2} \qquad (45)$$

Note that the distance is negative when the origin is on the negative side of the hyperplane.

The mathematical formulation to achieve the idea of pushing the boundary faces outward now can be understood from examination of Fig.1. The distance from the origin to the hyperplane is shown and labeled. The concept is to maximize the minimum distance as this will push the boundary away from the origin and usually increased the yield. This is similar to the simplicial approximation approach, but is also quite different in that the probability density function is completely fixed, and the boundaries move as the design point changes.

The distance from the origin need not always be positive, the definition of the distance allows the problem where the polytope is empty or far from the origin to be solved as well. The procedure will pull those hyperplanes with the negative distance up to the origin, then push them on past and away from the origin, create a large polytope around the origin,

Mathematically the problem to be solved can be stated as

$$\min_{w} \max_{j}\{z_j\} \qquad (46)$$

To solve this, the approach discussd in [17] and [18] was used. In this basic iterative approach, the distance functions are linearized, and problem cast as a linear programming problem, an LP. On each iteration k the following LP is solved:

$$\begin{aligned} &\min_{\Delta w_k} r^k \\ &s.t. \\ &\quad z_j^k + \nabla w^k \le r^k, \quad j = 1, \ldots, N_h \\ &\quad -\lambda^k \le \Delta w^k \le \lambda^k \\ &\quad B_l \le w^k + \Delta w^k \le B_u \end{aligned} \qquad (47)$$

Here B_l and B_u are parameter bounds given by the designer and their effect is to keep the next value of w within the bounds. The term λ^k is the current step length bound set by the algorithm and constrains how far any one iteration may move w. The top constraint equation is the linearized negative distance, and the gradient of the negative function $z(w)$ can be related back to the hyperplane coefficient sensitivities as follows:

$$\nabla_w z_j = \frac{\nabla_w v_j}{\|h_j\|_2} + \frac{b_j - v_j}{\|h_j\|_2^3} h_j^T \frac{\partial h_j}{\partial w} \qquad (48)$$

As discussed in the yield gradient approach, these coefficient sensitivities can be written in terms of circuit performance sensitivities; hence, they are computed easily from the transient sensitivities from circuit simulation.

The iteration control algorithm for this method adjusts λ and either accepts or rejects the new iterate Δw^k . Let us define the minimum distance and the linearly predicted increase as:

$$\Delta r_a = \max_j\{z_j^k\} - \max_j\{z_j^{k+1}\}$$
$$\Delta r_p = \max_j\{z_j^k\} - r^k$$

From the ratio of these terms one can determine β^k for this iteration from Fig.2 with the exception that if $\beta^k > 1.0$ and $\beta^{k-1} < 1.0$, then β^k is reduce to 1.0. Then the step length can be updated

$$\lambda^{k+1} = \beta^k \|\Delta w^k\|_\infty \qquad (49)$$

The current iteration is accepted, $w^{k+1} = w^k + \Delta w^k$, provide that $\Delta r_a / \Delta r_p > 10^3$; otherwise $w^{k+1} = w^k$.

From the experimental results, it is shown that the minimax technique has a speed advantage in computational cost per iteration over the quasi-Newton method to achieve a comparable result. The linear approximations are in close agreement with actual circuit simulation data.

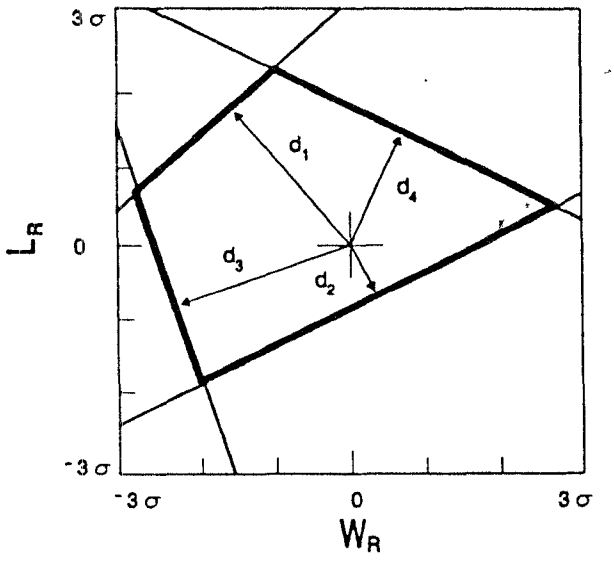


Fig. 1. Hypothetical yield body in terms of two statistical variables and depicting the distances of the boundary faces away from the center.

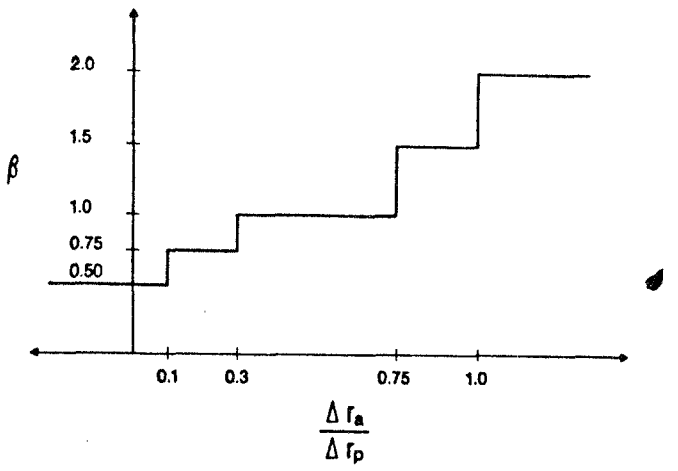


Fig. 2. Steplength adjustment factor β for minimax method versus ratio of actual improvement to predicted improvement.

Conclusions

In this paper, I have briefly introduced four optimization algorithms which compute the parametric yield of the integrated circuits. They are the simplicial approximation algorithm, statistical algorithm, linear algorithm and yield gradient algorithm.

The simplicial approximation algorithm assumes that the design parameters are independent sources of variations (which is only true for printed circuit boards), and the number of the design parameters is small compared to the number of algebraic-differential equations that need to be solved in order to determine the circuit behavior. The simplicial approximation also assumes that feasible region is a convex hull. These assumptions indicate that the algorithm is efficient for the digital circuits with low dimensionality. Since as the dimension of the feasible region grows, it would be very hard to keep the feasible region a convex hull, and the time spent in generating and inflating the convex hull will become large, also the computational effort to find a new boundary point will become very expensive. The implementation of the simplicial approximation algorithm is based on a straightforward interfacing of code for the algorithm with standard programs for (small scale) LP problems and for circuit simulations. More specifically, the method of "interior" simplicial approximation requires no gradient evaluations. Thus, any readily available conventional network simulator, e.g. SPICE, or ASTAP-11, can be employed to provide the user with immediate access to design centering via interior simplicial approximation.

The statistical methods employ regression models to approximate the system performances and then replace the circuit simulations in the Monte Carlo algorithm for

computing the parametric yield. From the experimental results of the statistical algorithm introduced in this paper, the speed up over Monte Carlo methods decreases as the number of parameters increases. This is because the number of possible terms in polynomial regression equations increases rapidly with the dimension of the problem. This indicates that polynomial regression is most effective on problems of low dimensionality digital circuits. On the other hand, a larger number of specifications favors statistical modeling because each of the specifications can be handled separately. Also, since the statistical modeling technique requires fewer simulations, it is potentially better for circuits that require a long time for simulation. And also, for problem with the same dimension, polynomial regression is likely to attain a higher speedup for larger circuits.

The yield gradient method and linear method (geometric approach - minimax optimization) utilize the fact that there are only a few independent statistical variables in MOS fabrication which account for most of the variation in a local area and consider these statistical variables separate from the circuit design parameters (as in the statistical methods). These two methods are particularly efficient for the MOS digital circuits. The gradient approach which is based on quasi-Newton techniques employs the gradient of the yield. The linear method uses a minimax technique where the concept is to push the boundaries of the yield body outward to increase the volume of the body; this involves solving an LP problem at each iteration. For the computational cost, since the minimax method does not require gradient computations, the minimax technique has a speed advantage in computational cost per iteration over the gradient method. Since no crucial assumptions were made in the implementation of the gradient method, the gradient based algorithm is more stable than the minimax method. An obvious extension would be to use both methods, perhaps starting with the minimax method and then switching to the gradient method. Since both methods require five circuit simulations at each optimization iteration, it is still costly to perform the yield optimization.

In conclusion, none of the methods discussed above works well if there are a large number of statistical variables, except the Monte Carlo Method. Therefore, analog circuits which have a large number of statistical variables require the use of the Monte Carlo method. However, digital circuits which have significantly fewer statistical variables are more suited to the methods previously discussed. Statistical methods, yield gradient and minimax, all use the better models of process variations and are more efficient than simplicial approximation. The main difference between the two is that the statistical method aims at more accuracy, where yield gradient and minimax are more concerned with efficiency.

References

[1] J.M. Hammersley and D.C. Handscomb, *Monte Carlo Methods.* Methuen and Co.Ltd., London, 1964.

[2] D.E. Hocever P.F. Cox and P. Yang, "Parametric yield optimization for MOS circuit blocks". IEEE *Transactions on Circuits and Systems,* CAS-29(2): 88-96, Feb, 1982.

[3] Hocevcu, Lisntner, Trick, "An extrapolated yield approximation technique for use in yield maximization", IEEE *Transactions on Computer-Aided design,* Vol. CAD-3, pp. 279-287, Oct, 1984.

[4] S.W. Director and G.D. Hachtel, "The Simplicial Approximation Approach to design Centering", IEEE *Transactions on circuits and systems,* Vol. CAS-24, No.7, pp. 363-372, July, 1977.

[5] L. Milor and A.S. Vincentelli, "Computing Parametric Yield Accurately and Efficiently".

[6] T.K. Yu, S.M. Kang, I.N. Hajj, T.N. Trick, "Statistical performance modeling and parametric yield estimation of MOS VLSI", IEEE *Transactions on Computer-Aided Design,* Vol. CAD-6, pp. 1013-1022, Nov, 1987

[7] T.K. Yu, S.M. Kang, J. Sacks. and W.J. Welch, "Parametric yield optimization of MOS integrated circuit by statistical modeling of circuit performances", *Technical Report No. 27* Dept. of Statistical, Univ. of Illinois, Champaign, July 1989.

[8] P.E. Gill, W. Murray, M.H.Wright, *Piratical Optimization.* London, UK: Academic, 1981.

[9] R. Fletcher, *Piratical Method of Optimization, Vol.1, Unconstrained Optimization.* Chichester: Wiley, 1980.

[10] K. Madsen, H. S-Jacobsen, J. Voldby "Automated minimax design of networks", IEEE *Trans. on Circuits Syst.,* Vol. CAS-22, pp. 791-796, Oct. 1975

[11] K. Madsen, "An algorithm for minimax solution of over determined systems of nonlinear equations", *J. Inst. Maths. Appl.,* Vol. 16, pp. 321-328, 1975