Statistical Timing for Parametric Yield Prediction of Digital Integrated Circuits

J. A. G. Jess[†], K. Kalafala[‡], S. R. Naidu[†], R. H. J. M. Otten[†], C. Visweswariah[*]
[†]Eindhoven University of Technology, Eindhoven, The Netherlands
[‡]IBM Microelectronics Division, East Fishkill, NY, USA
[*]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA

ABSTRACT

Uncertainty in circuit performance due to manufacturing and environmental variations is increasing with each new generation of technology. It is therefore important to predict the performance of a chip as a probabilistic quantity. This paper proposes three novel algorithms for *statistical timing analysis* and parametric yield prediction of digital integrated circuits. The methods have been implemented in the context of the EinsTimer static timing analyzer. Numerical results are presented to study the strengths and weaknesses of these complementary approaches. Across-the-chip variability continues to be accommodated by EinsTimer's "Linear Combination of Delay (LCD)" mode. Timing analysis results in the face of statistical temperature and V_{dd} variations are presented on an industrial ASIC part on which a bounded timing methodology leads to surprisingly wrong results.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated circuits—*Design aids*

General Terms

Algorithms, verification

Keywords

Statistical timing, yield prediction

1. INTRODUCTION

Yield loss is broadly categorized into *catastrophic yield loss* (due to contamination and dust particles, for example) and *parametric* or *circuit-limited yield loss* which impacts the spread of performance of functional parts. This paper presents three algorithms for statistical timing analysis and parametric yield prediction of digital integrated circuits due to both manufacturing and environmental variations.

With each new generation of technology, variability in chip performance is increasing. The increased variability renders existing timing analysis methodology unnecessarily pessimistic and unrealistic. The traditional "bounded" static timing approach further breaks down in the case of multiple voltage islands. The International Technology Road-map for Semiconductors (ITRS) [1] has identified a clear need for statistical timing analysis.

The algorithms in this paper pay special attention to correlations. Capturing and taking into account inherent correlations are absolutely key to obtaining a correct result. Correlations occur because different paths may share one or more gates, and because all gate delays depend on some global parameters such as junction depth or temperature. All methods in this paper fully take into account both classes of correlations and are equipped to handle *deterministic* across-the-chip variations. While the present work does not directly handle statistical intra-chip variations, the last section of the paper describes how these could be accommodated in the future.

2. PREVIOUS WORK

Due to space limitations, it is impossible to do justice to the wealth of literature on parametric statistical timing analysis and yield prediction. The methods are broadly classified into *performance-space* methods that manipulate timing variables such as arrival times and slacks as statistical quantities, and *parameter-space* methods that perform manipulations in the space of the sources of variation. In performance-space, we are conceptually interested in integrating the joint probability density function (JPDF) of the delays of all paths over a cube of side equal to the required delay and of dimensionality equal to the number of paths. In other words, it amounts to the integration of a complicated JPDF over a simple integration region in high-dimensional space. In parameter-space, on the other hand, we are interested in integrating the JPDF of the sources of parametric variation over a complex *feasible region* in relatively low-dimensional space.

Monte Carlo and modified Monte Carlo methods have often been used as in [2] where yield is estimated by means of a surface integral of the feasible region. In the context of digital circuits, [3] considers the probability of each path meeting its timing requirement, but ignores correlations between paths. An extremely efficient discrete probability approach in performance-space was proposed in [4], but path reconvergence is handled with difficulty and global correlations are ignored. A good source of information about statistical design is [5]. A recent performance-space probabilistic framework was proposed in [6] but has a restricted domain of application.

Unfortunately, most existing methods take into account one or other type of correlation mentioned above, but not both. They also often neglect the dependence of slew and downstream load capacitance on the sources of variation. These drawbacks are addressed in this paper.

3. MODELING

All three methods presented in this paper assume that the delay and slew of each arc of the timing graph are linear functions of the sources of variation, similar to the assumptions in [7, 3, 8], for example. However, the nominal delays and slews and the sensitivity coefficients can be location-dependent to accommodate deterministic intra-chip variability. The actual statistical timing analysis consists of two phases. In the first phase, a representative set of paths is gathered by the timing analysis program after a nominal timing analysis. The sensitivity coefficients of each "complete" path (including the launching and capturing clock paths if any) are computed and accumulated by path tracing procedures. In the second phase, the statistical timing engine predicts the distribution of the minimum of all the path slacks. All methods work off of a common timing graph and path-tracing procedure.

The *slack* (difference between required arrival time and arrival time for primary outputs or timing slack for latch timing tests) of each of P paths is modeled as

$$s_i = s_i^{nom} + \sum_{j=1}^{n} A_{ij}\delta z_j \qquad (1)$$

where s_i is the slack of the i^{th} path (a statistical quantity), s_i^{nom} is the nominal slack, n is the number of global sources of variation, A_{ij} is a $P \times n$ matrix of path sensitivities and δz_j is the variation of the j^{th} global parameter from its nominal value. Delay, slew and loading effects are taken into account in the coefficients of A in our implementation.

For a required slack η, we can write the feasible region

$$F = \{\delta z | s_i^{nom} + \sum_{j=1}^{n} A_{ij}\delta z_j \geq \eta, i = 1, 2, \cdots, n\}. \qquad (2)$$

Each of the above constraints represents a *hyperplane* in n-dimensional parameter-space, on one side of which the circuit has sufficient slack and on the other side of which it is a failing circuit. The intersection of all the "good" *half-spaces* forms a *convex polytope* and is defined as the *feasible region*. The goal of the parameter-space methods is to integrate the JPDF of the sources of variation in the feasible region. This procedure is repeated for a range of η values to produce the entire slack vs. yield curve.

4. PARALLELEPIPED METHOD

This section describes the first of the three novel statistical timing algorithms, called the parallelepiped method, which is a parameter-space method. The basic idea is to recursively divide the feasible region into the largest possible fully feasible parallelepipeds, and integrate the JPDF of the underlying sources of variation over these parallelepipeds instead of the original feasible region. The approach does not require delays to have linear models (but the bounds described below cannot be guaranteed with nonlinear models), and allows for arbitrary distributions of the sources of variation.

4.1 The algorithm

The basic reference on the parallelepiped approach is the second algorithm of Cohen and Hickey [9]. The method rests on the fact that if all vertices of an n-parallelepiped lie in any convex feasible region, then *all* points in the interior of the parallelepiped are feasible. With the above observation, the region of integration in the parameter space is recursively subdivided into progressively smaller parallelepipeds until we find parallelepipeds all of whose vertices are feasible. Then we simply sum up the weighted volume of the

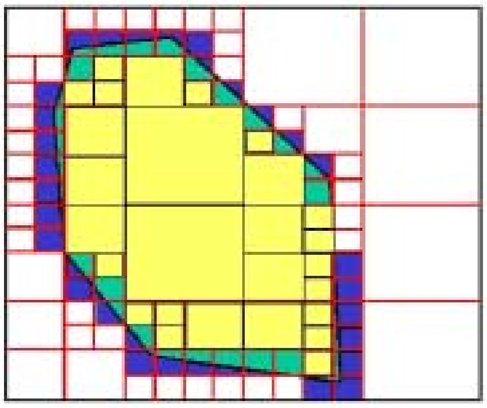

Figure 1: Illustration of the parallelepiped method in 2 dimensions to a recursion depth of 4.

feasible parallelepipeds to obtain a lower bound on the desired yield as shown in the pseudo-code below for a single given performance requirement.

```
procedure Vol(ll, recursionDepth){
  if(recursionDepth < maxDepth){
    if(all vertices of parallelepiped are feasible)
        add integral of region to yield;
    else{
      subdivide region into smaller parallelepipeds;
      for(each new parallelepiped p)
          Vol(lowerLeft(p), recursionDepth+1);
    }
  }
}
Vol(lowerLeft(boundingBox), 0);
```

The algorithm begins by choosing a `boundingBox` that is known to contain the feasible region. For statistical timing, the obvious choice is the $\pm 4\sigma$ or $\pm 3\sigma$ box in n-dimensions. In the algorithm, `lowerLeft` represents a function that returns the vertex of the parallelepiped that has the lowest coordinate in each dimension. Fig. 1 graphically illustrates the method in two dimensions. The yellow (in color copies of this paper) or grey (in black-and-white copies) regions contribute to the final yield computation, and obviously provide a lower bound on the required probability integral. Note that descent to the lowest level of recursion is confined to the boundaries of the feasible region.

Since at worst we visit every leaf node of a q-ary tree where $q = 2^n$, and at each vertex we check feasibility of each path constraint, we end up with a worst-case complexity of $O(Pn2^{(n\times \texttt{maxDepth})})$. Pn is the complexity of checking the feasibility of one vertex. In fact, if a static timer is employed, the feasibility of a vertex can be established more efficiently. In any case, the method is exponential in the product of the recursion depth and the dimensionality of the manufacturing space. However, several tricks can be applied to speed up this algorithm in practice.

1. If a particular path is infeasible at all vertices, the recursion can stop at once. No matter how deep the recursion, that particular path will not become feasible, so there is no good yield to be had.

2. If a particular path is feasible at all vertices, that path can be skipped as the recursion proceeds. This trick is implemented

by simply maintaining a list of "skippable" paths that grows as the depth of the recursion increases.

3. The number of recursion levels can be drastically reduced by modifying the basic algorithm to additionally produce an upper bound and a best estimate answer. The (strict) lower bound is still the weighted volume of the yellow/grey region of Fig. 1. At the lowest level of recursion, if at least one vertex is feasible and at least one infeasible, the upper bound gets the entire weighted volume of the parallelepiped (represented by the blue/black region in the Figure). Although not a strict upper bound, in practice this estimate always exceeds the exact yield. The "best estimate" result gets yield credit proportional to the fraction of vertices that is feasible. With this mechanism, we have found that three to four levels of recursion are always sufficient for accurate results. The law of large numbers helps, since each parallelepiped at the lowest level of recursion contributes a signed error.

4. The parallelepiped method can handle any statistical distribution of the underlying sources of variation, provided the JPDF can be integrated over the volume of a parallelepiped. If one or more sources of variation form a multi-variate normal distribution, that part of the integral can be expressed as the product of differences of error functions in that subspace. The manufacturing space is first rotated and scaled so as to obtain circular symmetry. Then the required error functions are precomputed and stored in a single array of size $2^n + 1$ to avoid repeated calls to the system erf function.

The following tricks will further improve efficiency, but have not yet been implemented.

1. Once a decision is made to recurse, only the *internal* vertices of the sub-parallelepipeds need to be visited, since feasibility at the vertices of the parent parallelepiped has already been ascertained.

2. Since the bulk of the weighted volume is near the center of the JPDF, an *adaptive grid* scheme could be considered which uses a finer grid near the origin of the Δz space, and a progressively coarser grid towards the boundary of the bounding box.

3. Recursion can be carried out by subdividing the parallelepiped in one dimension at a time, and if a path is infeasible at all vertices, for example, subdivision in the other dimensions is obviated.

4.2 Modified algorithm

The above algorithm has been adapted to compute the entire yield-vs.-slack curve at once instead of one point at a time. As each parallelepiped is processed, the ramifications on yield for *all* performances are simultaneously recorded before proceeding to the next parallelepiped or next level of recursion. All CPU time results in this paper use this modified method.

The basic idea is briefly explained here. Let s_{min}^{parent} be the smallest slack at any of the vertices of the *parent* parallelepiped. Then for all slack below s_{min}^{parent}, the entire parent parallelepiped is in the feasible region, and appropriate yield credit is given. As we recurse, we are only interested in slacks greater than s_{min}^{parent} within this volume. For each sub-parallelepiped at the present level of recursion, yield credit corresponding to the smaller parallelepiped is granted for all slack from s_{min}^{parent} to s_{min} (lowest slack amongst the vertices of the sub-parallelepiped). The upper bound and best guess yields are similarly kept updated as the recursion proceeds.

Thus the entire yield curve is produced by a single recursive loop. The modified algorithm computes identical results compared to repeated invocation of the basic algorithm presented earlier. Modified versions of all of the tricks mentioned in conjunction with the basic algorithm continue to be applicable, and have been implemented in our prototype.

5. ELLIPSOID METHOD

This section describes the second of the three novel statistical timing algorithms, called the ellipsoid method, which is a parameter-space method. The basic idea is to compute the maximum volume n-dimensional ellipsoid that is entirely within the feasible region (a similar idea was explored in [10]), and integrate the JPDF of the sources of variation in the simpler ellipsoidal approximation rather than the original feasible region. The integral provides a lower bound on the yield. This method relies on a linear delay model, but allows arbitrary distributions of the underlying sources of variation.

5.1 Ellipse computation

There has been tremendous recent progress in solving the maximum volume ellipsoid problem. It is shown in [11] that ellipsoidal volume maximization subject to linear constraints is a special case of the MAXDET problem in which the determinant of a matrix is maximized subject to linear matrix inequalities (LMIs). We start with a set of linear inequality constraints that define a feasible region (2), and which can be expressed in matrix form as

$$F = \{\Delta z \mid R_i^T \Delta z \le t_i, i = 1, 2, \cdots P\}, \tag{3}$$

where R_i^T is the i^{th} row of $-A$ and $t_i = s_i^{nom} - \eta$. An arbitrary ellipsoid is expressed as a collection of points

$$E = \{By + d \mid \|y\| \le 1\}, \tag{4}$$

where B is a symmetric, positive-definite matrix that linearly transforms all points in the unit sphere, and d is the center of the ellipsoid. To find the largest ellipse in the feasible region, it is sufficient to maximize the determinant of the transformation matrix B since the volume of the ellipse is the volume of the unit sphere times the determinant of B. The requirement that $E \subset F$ means that

$$R_i^T(By + d) \le t_i \text{ for all } \|y\| \le 1, i = 1, 2, \cdots, P. \tag{5}$$

This in turn means that

$$sup_{\|y\| \le 1}(R_i^T By + R_i^T d) \le t_i, i = 1, 2, \cdots, P \tag{6}$$

or

$$\|BR_i\| + R_i^T d \le t_i, i = 1, 2, \cdots, P. \tag{7}$$

To find the ellipsoid of largest volume inside the feasible region F, we solve the convex optimization problem

$$\begin{array}{ll} \text{maximize} & \log\det B \\ \text{subject to} & B = B^T > 0 \\ \text{subject to} & \|BR_i\| + R_i^T d \le t_i, i = 1, 2, \cdots, P. \end{array} \tag{8}$$

Zhang and Gao [12] recently proposed an efficient, structure-exploiting primal-dual optimization algorithm to solve (8), and have made available public-domain MATLAB code. By manipulating the optimality conditions and taking advantage of the properties of transformation matrices of ellipsoids, this method solves for fewer variables, can handle problems of larger dimensionality and is vastly more efficient than the original implementation of [11]. In addition to the linear slack constraints, bounding box constraints (e.g., the $\pm 4\sigma$ box) are applied. An example of the ellipse computed by the program in 2 dimensions is shown in Fig. 2.

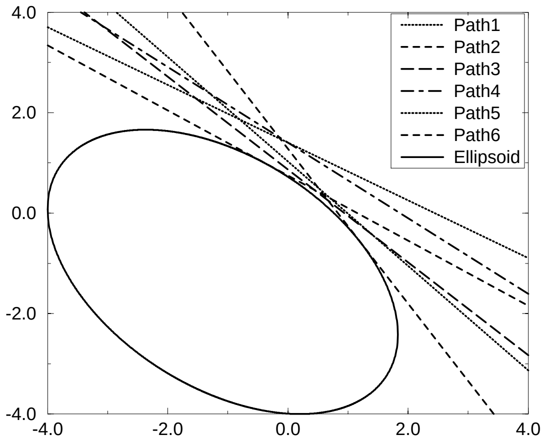

Figure 2: Largest ellipse in 2 dimensions bounded by hyperplane constraints and the 4σ box.

5.2 Integration

There remains the problem of integrating the JPDF of the sources of variation over the resulting ellipsoidal approximation of the feasible region. The yield is represented as

$$\int\int\cdots\int_{R^*} JPDF(z_1, z_2, \ldots, z_n)\, dz_n dz_{n-1}\ldots dz_1. \quad (9)$$

where R^* is the ellipsoidal region. Instead of integrating over the ellipsoid we perform a change of variables and integrate over the unit sphere. Also,

$$\begin{aligned}\delta z_1 \delta z_2 \ldots \delta z_n &= \begin{vmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \cdots & \frac{\partial z_1}{\partial y_n} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \cdots & \frac{\partial z_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \frac{\partial z_n}{\partial y_2} & \cdots & \frac{\partial z_n}{\partial y_n} \end{vmatrix} \delta y_1 \delta y_2 \ldots \delta y_n \\ &= |B| \delta y_1 \delta y_2 \ldots \delta y_n, \end{aligned} \quad (10)$$

where we make use of the fact that the Jacobian matrix is simply the transformation matrix B which is computed by the convex optimization problem discussed previously. Therefore the yield Y can be written as

$$\begin{aligned} Y &= \int\int\cdots\int_{R^*} JPDF(z)dz \\ &= \int\int\cdots\int_{R} JPDF(By+d)|B|dy. \end{aligned} \quad (11)$$

where R is the unit sphere.

The integral in (11) is computed by a modification of the stochastic integration method of Genz and Monahan [13]. The integration is split up into the product of a *radial* and *spherical* part. The spherical part is accomplished by applying Mysovskikh's rules [14, 15] to a series of spherical surfaces (or infinitesimal annulus shells), and the radial part is computed by Gaussian quadrature. Spherical integration is performed by randomizing deterministic integration rules to obtain a higher degree of accuracy than conventional Monte-Carlo integration. Integration is performed by applying a degree-5 rule with $(n+1)(n+2)$ points, giving the method polynomial complexity. The basic set of points is randomly rotated to get a new set of points. The quality of the radial integration can be improved by increasing the number of spherical surfaces, whereas the quality of the spherical integration can be improved by increasing the number of points sampled on each spherical surface.

6. BINDING PROBABILITY METHOD

This section describes the last of the three novel statistical timing algorithms, called the binding probability method, which is a performance-space method. The basic idea of this method is to compute the probability distribution of the minimum slack of the first two nominally most critical paths. Then a recursion is set up to find the distribution of the minimum of this distribution and the third most critical path, and so on. The difficult part, of course, is to keep the correlations alive as the recursion proceeds. This method rests on both the linear delay model as well as requiring the underlying parameter distributions to be Gaussian.

6.1 Computing PDF and binding probability

The slack of every path is a linear combination of the Gaussian sources of variation, and hence is Gaussian. The algorithm begins by taking the two nominally most critical paths and computing their 2×2 covariance matrix

$$\Phi = \begin{bmatrix} (\frac{\partial s_1}{\partial z})^T \\ (\frac{\partial s_2}{\partial z})^T \end{bmatrix} [V] \begin{bmatrix} \frac{\partial s_1}{\partial z} \frac{\partial s_2}{\partial z} \end{bmatrix} = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} [V]\, [A_1 A_2] \quad (12)$$

where s_1 is the slack of the first path, s_2 is the slack of the second path, A_i^T is the i^{th} row of A, z are the sources of variation and V is the $n \times n$ covariance matrix of the sources of variation. Comparing to

$$\Phi = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{bmatrix} \quad (13)$$

the variances σ_1, σ_2 and the correlation coefficient ρ can easily be computed. Next, the distribution of $\min(s_1, s_2)$ is computed as follows.

$$\begin{aligned} p_1 &= p(s_1 = \eta) &&= \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma_1} e^{-\frac{1}{2}(\frac{\eta-\mu_1}{\sigma_1})^2} d\eta \\ p_2 &= p(s_2 = \eta) &&= \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma_2} e^{-\frac{1}{2}(\frac{\eta-\mu_2}{\sigma_2})^2} d\eta \\ p_3 &= p(s_1 \le \eta | p_2) &&= 0.5 + \frac{1}{2} erf\left(\frac{\left(\frac{\eta-\mu_1}{\sqrt{2}\sigma_1}\right) - \left(\frac{\eta-\mu_2}{\sqrt{2}\sigma_2}\right)\rho}{\sqrt{1-\rho^2}}\right) \\ p_4 &= p(s_2 \le \eta | p_1) &&= 0.5 + \frac{1}{2} erf\left(\frac{\left(\frac{\eta-\mu_2}{\sqrt{2}\sigma_2}\right) - \left(\frac{\eta-\mu_1}{\sqrt{2}\sigma_1}\right)\rho}{\sqrt{1-\rho^2}}\right). \end{aligned} \quad (14)$$

By sweeping through a range of η values (theoretically from $-\infty$ to ∞), the entire PDF of $\min(s_1, s_2) = p_1 p_4 + p_2 p_3$ is computed. Further, the probability that the first path dominates the second over all possible η values is called the *binding probability* of the first path, $b_1 = \int_{\eta=-\infty}^{\infty} p_1 p_4$. Likewise, the probability that the second path dominates is $b_2 = 1 - b_1 = \int_{\eta=-\infty}^{\infty} p_2 p_3$. As a practical matter, the algorithm sweeps η through a range that includes $\pm 3\sigma$ or $\pm 4\sigma$ of both the paths being processed.

6.2 The recursion

The next step is to create a *fictitious path* that captures the correlations of all the paths processed so far. The fictitious path consists of a *linear combination* of all the timing graph edges along path 1 with probability b_1 and all the gates along path 2 with probability b_2; in other words, a vector consisting of a linear combination of the slack sensitivities of the two paths is created, to be plugged back into (12) for the covariance computation at the next step of recursion. If one or the other path is always dominating (binding probability of unity), its sensitivities are preserved as is. The distribution obtained above is then approximated to be Gaussian. The algorithm proceeds by finding the probability distribution of the minimum of this fictitious path slack and the third most critical path, and so on, until the probability distribution changes by a sufficiently small tolerance.

At the end of this procedure, the binding probabilities accumulated along the way give us the probability that any given path is critical, and in fact the probability that any given branch of the timing graph is critical. Such diagnostics can be used to guide yield-aware optimization.

7. IMPLEMENTATION

All three methods presented in this paper have been implemented in C++ as a prototype component called EinsStat in the EinsTimer static timing analysis environment. The program runs under Nutshell with TCL scripting capabilities. In the case of the ellipsoid method, the maximum volume ellipsoid MATLAB code of Zhang and Gao [12] was converted via the MATLAB compiler to C++.

The "front-end" collects the worst paths, and computes their nominal slacks and sensitivities directly off the EinsTimer timing graph. Currently, a user-specified number of worst data paths are collected throughout the entire design (which could easily be extended to collect all paths within a user-specified slack window). For each path, a corresponding downstream (setup) test slack is computed – this implicitly takes into account the most critical clock path leading to the test. "Full paths" are traced from the clock source, through the launching latch, through the data path, to the capturing latch and back to the clock source. This way, common clock path correlations are fully captured.

Sensitivities with respect to global environmental conditions are determined by finite differences as our analytical delay models [16] are characterized as functions of voltage and temperature. In addition, the underlying delay calculations fully account for input slew and downstream pin capacitance dependencies on the sources of variation.

8. NUMERICAL RESULTS

Two flavors of numerical results will be presented in this section. The first is results obtained from statistical analysis of a real-world 200K gate ASIC design with environmental variations, while leaving manufacturing parameters at their "slow chip" setting. The second set of results is from running artificially generated problems with a large number of nominally equally critical paths and random sensitivities.

A 200K gate ASIC circuit was first analyzed with individual temperature and voltage variations, leading to a surprising result. The EinsTimer best-case result was the worst slack of all and the nominal result was the best slack! Further investigation revealed that this chip has a short primary-input-to-latch path, whose slack deteriorates rapidly with lower temperature and higher V_{dd} because the clock is disproportionately sped up. At higher temperatures and lower V_{dd} values, latch-to-latch paths with more traditional slack sensitivities dominate. This type of surprising result is easily unearthed with statistical analysis.

Fig. 3 shows statistical timing results on the 200K gate ASIC circuit with simultaneous temperature and voltage variations. The Y axis represents yield and the X axis represents slack. Superposed on the same plot are results obtained by running EinsTimer 1,200 times at a regular grid of sample temperature/V_{dd} pairs and converting the sample points to probabilities. All methods are pretty accurate, except the binding probability method that is unable to accurately capture the highly skewed slack distribution in this case. CPU times on an IBM Risc/System 6000 model 43P-S85 are shown in Table 1. Although results are shown here with only two sources of environmental variation, the anticipated applications of these

Table 1: CPU times on 200K gate ASIC

Method	15,000 paths	10,000 paths
Repeated EinsTimer runs	68 hours	
Monte Carlo 1M samples	855 s	60 s
Parallelepipeds	141 s	20 s
Binding probability	152 s	20 s
Ellipsoid	Out of memory	3.41 hours

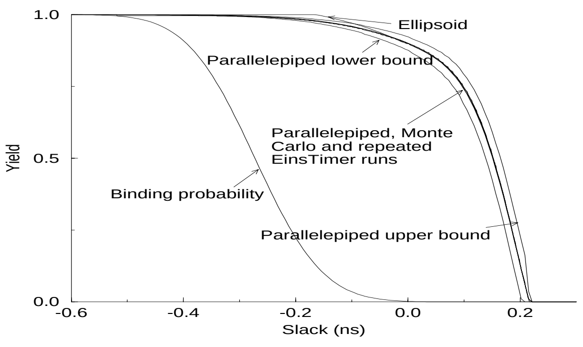

Figure 3: Statistical timing results on 200K gate ASIC.

methods are to solve the problem of timing circuits with multiple voltage islands and to take manufacturing variations into account.

The second set of results are from randomly generated problems with a large number of nominally equally critical paths. Fig. 4a shows the growth in CPU time as a function of the number of paths analyzed, with the number of sources of variation fixed at 4 and 100 points requested on the slack curve. The ellipsoid method has polynomial complexity in the number of paths, while the others are linear. Fig. 4b shows the growth in CPU time as a function of the number of points requested on the yield curve (number of variations fixed at 4, number of paths at 1,000). With the exception of the ellipsoid method, all the methods are insensitive to first order to the number of data points requested. Finally, Fig. 4c shows the growth of CPU time with the number of sources of variation (paths fixed at 1,000 and data points at 100). To first order, the Monte Carlo and binding probability methods are unaffected by the number of parameters, whereas the ellipsoid method has polynomial dependence and the parallelepiped method has exponential dependence which dominates the run time above 6 dimensions.

9. COMPARISONS

The parallelepiped method is very fast and accurate at low-dimensionality, but has exponential growth of CPU time with the number of sources of variation. The CPU time is independent to first order of the number of points requested on the yield curve, and linear with the number of paths selected. Hence it is best suited to accurate but low-dimensionality analysis.

The ellipsoid method, on the other hand, handles high-dimensionality extremely well and successfully handled problems with over 20 sources of variation. However, it has linear growth with the number of points requested on the slack curve, and polynomial growth with the number of paths. Thus it is most effective when the dimensionality is high and the number of paths can be filtered down to a manageable number.

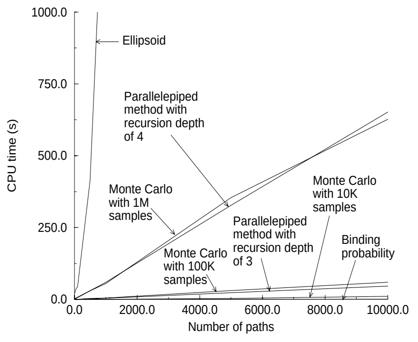

Figure 4a. CPU time vs. number of paths.

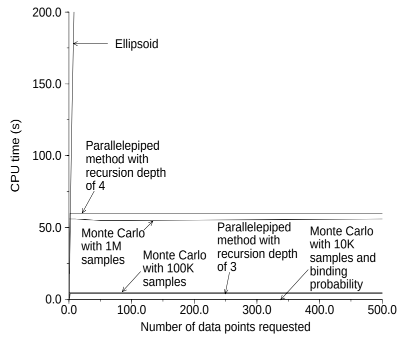

Figure 4b. CPU time vs. number of data points requested

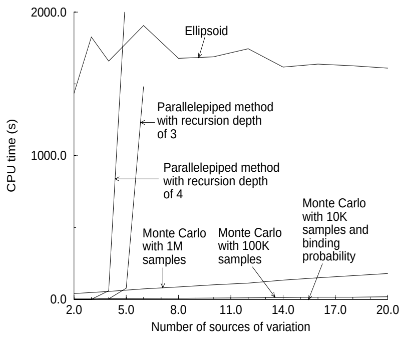

Figure 4c. CPU time vs. number of sources of variation.

The binding probability method is extremely fast, and consistently outperforms all the other methods. It is also the least accurate of the methods proposed, both because of the Gaussian approximation, and because of the loss of accuracy in propagating correlations. The complexity is linear in the product of the number of paths and the number of data points requested on the yield curve, but is relatively insensitive to the number of sources of variation.

The three methods therefore provide a complementary arsenal of techniques depending on the situation at hand.

10. CONCLUSIONS AND FUTURE WORK

This paper presented three algorithms for statistical timing analysis that pay a great deal of attention to the inherent correlation between the delays of gates and paths on a chip. Each method has strengths and weaknesses, and by implementing all three in a common infrastructure and with a common interface, the best features of each method can be exploited as the situation demands. Results of statistical timing analysis on a 200K gate ASIC were presented.

There are several avenues of future work. Several measures to improve efficiency were suggested in the body of this manuscript. Various diagnostics can be inferred from these methods, too. For example, in the ellipsoid method, the major and minor axes of the ellipse tell us the least and most important directions in which to nudge the circuit for improved parametric yield, and the most important manufacturing parameters on which to improve control if possible. The binding probability method gives us a rank-ordered set of gates, the improvement of whose delays will have the most impact on improving yield. The extreme efficiency of the binding probability method is motivating some new research into handling skewed distributions in this method. Extending the method to compute yield gradients will enable automated yield-aware optimization.

Two other extensions are intriguing. The first is to compute the so-called Löwner-John ellipsoid, which is the smallest ellipsoid that circumscribes the feasible region, so as to obtain an upper bound on the yield. The second is to obtain a simplicial decomposition of the feasible region (see [17] for an excellent survey) and then to integrate the JPDF of the sources of variation over the resulting simplices.

Finally, statistical intra-chip variation can be accommodated in a number of ways. One technique is to have a position-dependent random variable, upon which the delays of all gates depend. Another is to divide the chip into regions, with each region having a common set of random variables. The variables of nearby regions are tightly correlated, while those that are far apart are only loosely correlated.

11. REFERENCES

[1] "International technology roadmap for semiconductors 2001 edition," tech. rep., Semiconductor industry association, 2001. Available at `http://public.itrs.net/Files/2001ITRS/Home.htm`.

[2] P. Feldmann and S. W. Director, "Integrated circuit quality optimization using surface integrals," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 12, pp. 1868–1879, December 1993.

[3] A. Gattiker, S. Nassif, R. Dinakar, and C. Long, "Timing yield estimation from static timing analysis," *Proc. IEEE International Symposium on Quality Electronic Design*, pp. 437–442, 2001.

[4] J.-J. Liou, K.-T. Cheng, S. Kundu, and A. Krstic, "Fast statistical timing analysis by probabilistic event propagation," *Proc. 2001 Design Automation Conference*, pp. 661–666, June 2001.

[5] S. W. Director and W. Maly, eds., *Statistical approach to VLSI*, vol. 8 of *Advances in CAD for VLSI*. North-Holland, 1994.

[6] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," *Proc. 2002 Design Automation Conference*, pp. 556–561, June 2002.

[7] D. E. Hocevar, P. F. Cox, and P. Yang, "Parametric yield optimization for MOS circuit blocks," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 7, pp. 645–658, June 1988.

[8] K. Krishna and S. W. Director, "The linearized performance penalty (LPP) method for optimization of parametric yield and its reliability," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 14, pp. 1557–1568, December 1995.

[9] J. Cohen and T. Hickey, "Two algorithms for determining volumes of convex polyhedra," *Journal of the ACM*, vol. 26, pp. 401–414, July 1979.

[10] J. M. Wojciechowski and J. Vlach, "Ellipsoidal method for design centering and yield estimation," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 12, pp. 1570–1579, October 1993.

[11] L. Vandenberghe, S. Boyd, and S.-P. Wu, "Determinant maximization with linear matrix inequality constraints," tech. rep., Information Systems Laboratory, Electrical Engineering Department, Stanford University, April 1996. Available from `http://www.stanford.edu/~boyd/maxdet.html`.

[12] Y. Zhang and L. Gao, "On numerical solution of the maximum volume ellipsoid problem," Tech. Rep. CAAM technical report TR01-15, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, August 2001.

[13] A. Genz and J. Monahan, "A stochastic algorithm for high-dimensional multiple integrals over unbounded regions with gaussian weight," *Journal of Computational Applied Mathematics*, no. 112, pp. 71–81, 1999.

[14] I. P. Mysovskikh, "The approximation of multiple integrals by using interpolatory cubature formulae," in *Qualitative approximation* (R. A. DeVore and K. Scherer, eds.), (New York), pp. 217–243, AP, 1980.

[15] I. P. Mysovskikh, *Interpolatory cubature formulas*. Moscow-Leningrad: Izd 'Nauka', 1981. Russian text.

[16] "IEEE standard for integrated circuit (IC) delay and power calculation system," no. IEEE Standard 1481-1999, pp. 1–390, 1999. available at `http://fp.ieeexplore.ieee.org/iel5/6837/18380/00846710.pdf?isNumber=18380&prod=standards&arnumber=00846710`.

[17] B. Büeler and A. Enge and K. Fukuda, "Exact volume computation for polytopes: a practical study," in *Polytopes - combinatorics and computation* (G. Kalai and e. G. M. Ziegler, eds.), vol. DMV-Seminars vol. 29, Birkhäuser Verlag, Basel, Germany, 2000. Available from `http://www.lix.polytechnique.fr/Labo/Andreas.Enge/Volumen_en.html`.