A Linear-Time Approach for Static Timing Analysis Covering All Process Corners

Sari Onaissi, *Student Member, IEEE*, and Farid N. Najm, *Fellow, IEEE*

***Abstract*—Manufacturing process variations lead to circuit timing variability and a corresponding timing yield loss. Traditional corner analysis consists of checking all process corners (combinations of process parameter extremes) to make sure that circuit timing constraints are met at all corners, typically by running static timing analysis (STA) at every corner. This approach is becoming too expensive due to the increase in the number of corners with modern processes. As an alternative, we propose a linear-time approach for STA which covers all process corners in a single pass. Our technique assumes a linear dependence of delays and slews on process parameters and provides estimates of the worst case circuit delay and slew. It exhibits high accuracy in practice, and if the circuit has m gates and n relevant process parameters, the complexity of the algorithm is $\mathcal{O}(mn)$.**

***Index Terms*—Hyperplanes, multicorner, process variations, static timing analysis (STA).**

I. Introduction

THE CONTINUOUS scaling of very large-scale integration (VLSI) technology has led to an increase in the impact that manufacturing process variations can have on circuit delays. These process variations can include die-to-die and within-die process variations, and more generally, they include supply voltage and temperature variations.

One traditional approach to timing verification, at least for application-specific integrated circuits (ASICs), is to make sure that a circuit passes its timing requirements at every process corner, using static timing analysis (STA). We will refer to this as traditional corner analysis. A loose definition of a "process corner" is that it is a vector of extreme values of all process parameters under consideration. However, such techniques, which involve performing STA over all corners, can be time consuming as the number of corners can be exponential in the number of process parameters under study. Moreover, such methods usually do not allow for the incorporation of within-die variations into the timing analysis of a circuit.

With the increase in the number of interesting process variables in modern processes, the increased cost of traditional corner analysis has become a concern. One alternative approach has been statistical STA (SSTA) [2], [5]–[7], [9], [13]. In SSTA, process parameters are considered to be random variables (RVs), and they lead to other RVs that model cell delays and signal arrival times. However, SSTA has certain problems of its own. For one thing, it depends on knowledge of correlations among within-die features, which are not easily available. Also, it is not necessarily very cheap, particularly when one needs to use principal component analysis to resolve the within-die correlation issue. Furthermore, certain parameters that affect timing are not statistical in nature and cannot be handled using SSTA. For example, supply voltage and temperature variations are not necessarily statistical; rather, they are uncertain. Assuming that an uncertain variable has a statistical distribution (of any kind) when it actually does not can lead to wrong conclusions. SSTA techniques typically compute bounds on delays (slews) by looking at worst case points in delay (slew) distributions. However, to the best of our knowledge, all these methods use parameter distributions, or at the very least, they use the fact that parameters have distributions with certain properties (e.g., apply the central limit theorem) in order to compute these bounds. Therefore, when some of the underlying parameters are uncertain and not random, the application of such methods becomes problematic. In this paper, we consider all parameters to be uncertain rather than statistical. Obviously, any statistical parameter (with known finite-span distribution) can be modeled by an uncertain parameter (with a certain range), but not the other way around. Therefore, this paper can be used to handle statistical parameters as well, for example, if their distributions are uncertain or if their correlations are unknown, so that one is prepared to assume them to be independent.

We propose a novel technique for all-corner analysis in a single shot, with an approach that looks very much like a single run of STA. This is achieved by using linear models of delay and slew (in terms of underlying parameters) and propagating sensitivities through the circuit. The computational complexity of the algorithm will be seen to be $\mathcal{O}(mn)$, where m is the number of gates or cells in the circuit and n is the number of process parameters under consideration. Compare this with the cost of traditional corner analysis, which is $\mathcal{O}(m2^n)$. Our approach is not ideal, it does incur some error in the estimation of the worst case delay, for example, but this error is negligible for the circuits that we have tested.

The rest of this paper is organized as follows. An overview is given in Section II, which conveys the salient features of our technique including the delay and slew model and the scope of this paper. In Section III, we present our "max" operation for finding the maximum of a set of affine linear functions in process parameters. A description is then given in Section IV

Manuscript received June 1, 2007; revised September 28, 2007 and January 11, 2008. This work was supported in part by Intel Corporation and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). This paper was recommended by Associate Editor D. Sylvester.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: sari@eecg.utoronto.ca).

Digital Object Identifier 10.1109/TCAD.2008.923635

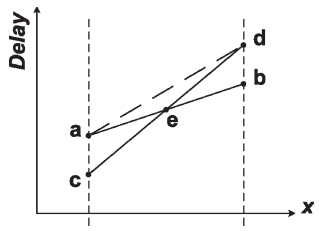

Fig. 1. Simple 1-D case.

of the propagation technique through a single logic gate, cell, or stage. Section V is a brief description of circuit-level propagation, and Section VI describes implementation issues. Finally, Section VII presents empirical validation results, and concluding remarks are given in Section VIII.

II. Overview

The general idea of our approach is very similar to traditional STA, except that, instead of using specific values of arrival times, delays, and slews, we represent them as affine linear functions of the underlying process parameters.

A. Linearity

Linearity is not too strong an assumption, as one may easily verify by circuit simulation on a modern process, and it has been widely adopted recently in the context of SSTA (e.g., in [13]). Propagating linear functions of delay in the context of STA means that "add" and "max" operations have to be performed on these linear functions at every node of the timing graph. Moreover, state-of-the-art slew propagation techniques also require the use of a "max" operation when finding the slew of a signal at the output of a logic cell, although this "max" operation is used in a somewhat less straightforward way than in the case of delays. Thus, propagating linear functions in a timing graph would seem problematic at face value because, while the summation of two linear functions is also a linear function, this is not true when one considers the "max" operation. For instance, in the simplest case when, for example, delay depends linearly on a single parameter, such as in Fig. 1, the max of two intersecting straight line segments **ab** and **cd** is a broken line **aed**. In this paper, instead of the true maximum of two planes, which is not a plane, we will use a new plane which is an upper bound on the two planes at all points. Thus, in Fig. 1, we would use the dashed line **ad** in place of the true maximum **aed**. We will only be concerned with the accuracy (tightness of this upper bound) at the process corners and not at any nominal midrange points. The trick is to do this efficiently and with good accuracy. It looks easy in the 1-D case, but it is not so simple in general.

As a final comment on the linearity question, we should mention that, even if the linear dependence of gate delay (or slew) on process parameters is not strictly valid, one can still apply the proposed technique by first constructing a linear expression which is an upper bound on whatever nonlinear surface that one may have for describing the true dependence of delay (or slew) on these parameters, and then use that linear expression in place of the true delay (or slew) in our algorithm.

B. Delay and Slew Model

All delays (and consequently all arrival times) and slews in the logic circuit will be captured as affine linear functions of normalized process parameters, whose values range between -1 and $+1$. We will refer to these functions as delay and slew hyperplanes; if there are n process parameters under consideration, these functions represent planes in $(n+1)$-dimensional space. Normalizing a process parameter is a trivial operation, which can be illustrated with a simple example. Suppose that the delay of a logic gate depends linearly on one process parameter, for example, V_t, according to $D = \alpha_0 + s\Delta V_t$, where $-0.05\ \text{V} \leq \Delta V_t \leq 0.05\ \text{V}$ and s has units of seconds per volt. This process parameter can be normalized, i.e., made to vary between -1 and $+1$, by simply multiplying s by 0.05 V, leading to a new sensitivity coefficient $\alpha = (0.05\ \text{V})s$ whose units are seconds. If a unitless variable X, which varies between -1 and $+1$, is used to represent the normalized threshold voltage, then the gate delay can now be written as $D = \alpha_0 + \alpha X$.

In general, having normalized all process parameters, a delay or slew hyperplane is captured by a linear expression as follows:

$$H = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_n X_n \tag{1}$$

where X_i are the normalized process parameters, α_0 is the nominal value of delay or slew, and α_i are the sensitivities of H to the different normalized process parameter variations. A corner C is defined as a set of values of all the normalized process parameters, where each of the parameters takes a value of either -1 or $+1$. Therefore, $C = (X_1, X_2, \ldots, X_i, \ldots, X_n)$, where $X_i = \pm 1$ for $1 \leq i \leq n$. If the total number of process parameters under consideration is n, then the total number of corners is 2^n.

The value of a hyperplane at a corner is the figure obtained by substituting values of the coordinates of the corner in the equation of the hyperplane. Thus, if, for a certain hyperplane, the value at corner C is D_C, then the point (C, D_C) belongs to this plane in $(n+1)$-dimensional space. This terminology is used to refer to both delays and slews, depending on the hyperplane being considered. We also use the notation $H(C)$ to refer to the value of hyperplane H at corner C. Finally, we will use the terminology "height of a point" in a hyperplane to refer to the value (delay or slew) of that point. This will help provide some intuition for the various operations that we will perform on hyperplanes.

C. Scope of This Paper

In traditional corner analysis, one is typically concerned with global die-to-die variations, not with within-die variations. The true reason for this is the exponential complexity of traditional corner analysis. It becomes too expensive to enumerate combinations of within-die variations. However, due to the linear-time

complexity of our approach, as will be seen later, it is actually possible to apply it to within-die variations as well. Indeed, the only requirement on our variables X_i in (1) is that their various combinations be meaningful process corners that one cares about. Some of them may be physical, some may be voltage or temperature, some may be global, some may be local, etc. In this paper, we will simply refer to the X_i as process parameters and to their combinations as process corners, without regard to exactly what type of parameters they are.

Due to space limitations, and for clarity of the presentation, the description of the technique in this paper will be somewhat limited. For one thing, we will focus on combinational circuits, which are the crux of the problem, and we will only discuss the problem of estimating the largest circuit delay and slew. In other words, we focus on setup constraints. However, the work is applicable as is to hold constraints, and we will show a couple of charts at the end that illustrate our results on estimation of the minimum circuit delay and slew (required to check for hold time violations). Furthermore, if one includes within-die variables, then the technique becomes useful for checking the margins that one needs to leave for clock skew and other mismatch related effects. With-in die variations would be handled by including a separate (independent) variable for each gate/region. This may result in longer delay expressions. Whether it is pessimistic or not depends on whether the assumption of independence among these variables, made by the user, is realistic or not. This is not discussed in detail here, and more work is required to fully apply our approach in that context. This remains a possible future application of this paper.

III. Max Operation

In traditional STA, "add" and "max" operations are used to find the signal arrival time and slew at the output of a node in the timing graph, given the arrival times and slews of its input signals. As mentioned earlier, our method is quite similar to traditional STA, except that, instead of propagating delays and slews as exact values, they are propagated as hyperplanes. This gives rise to the need for an efficient method for finding the maximum of a given set of hyperplanes (note that adding hyperplanes is a very simple operation). For a given set of hyperplanes, if we consider the largest value at every corner (over all the input hyperplanes evaluated at that corner), then the resulting set of 2^n points obviously need not lie on a single hyperplane. For example, in Fig. 2, the four highest points at the corners are $(-1, -1, 10)$, $(-1, +1, 10)$, $(+1, -1, 16)$, and $(+1, +1, 14)$, and they are not coplanar. Yet, in order to maintain computational efficiency, we will insist on modeling all delays and slews with hyperplanes. Thus, we seek to find a hyperplane that acts as a ceiling to the given hyperplanes at all corners, never underestimating the maximum value at any corner but possibly overestimating it at some, as we never want to underestimate the maximum delay or slew at any process corner. We are interested in a maximum hyperplane that has minimal overestimation. The problem of finding an optimal such hyperplane, which minimizes, for example, the average overestimation error at all corners, can be formulated as a linear program (LP). However, such an LP would be of exponential

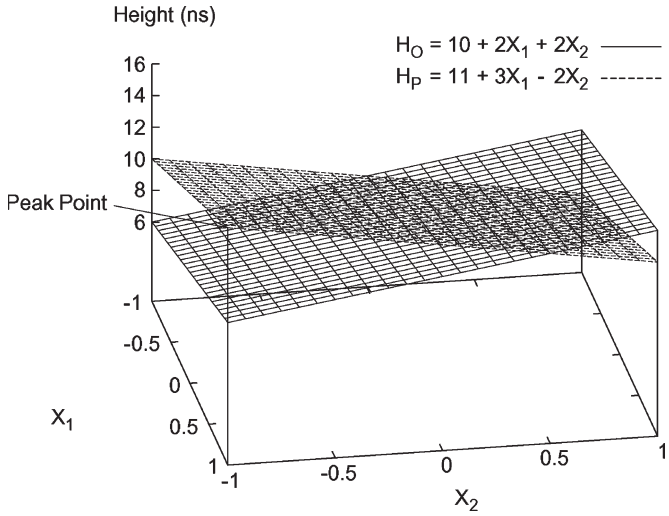

Fig. 2. Peak point of two hyperplanes.

complexity, which is not acceptable. Instead, in this paper, we propose a linear-time algorithm for finding a "good" maximum hyperplane that is a ceiling on all the given hyperplanes without being too pessimistic. Let the following k hyperplanes be our given set of hyperplanes for which we want to find a maximum hyperplane:

$$\begin{aligned} H_1 &= \alpha_0^{(1)} + \alpha_1^{(1)} X_1 + \alpha_2^{(1)} X_2 + \cdots + \alpha_n^{(1)} X_n \\ H_2 &= \alpha_0^{(2)} + \alpha_1^{(2)} X_1 + \alpha_2^{(2)} X_2 + \cdots + \alpha_n^{(2)} X_n \\ &\vdots \\ H_k &= \alpha_0^{(k)} + \alpha_1^{(k)} X_1 + \alpha_2^{(k)} X_2 + \cdots + \alpha_n^{(k)} X_n. \end{aligned} \tag{2}$$

We refer to these hyperplanes as the input hyperplanes, and we shall refer to our objective hyperplane (their maximum) as the output hyperplane H_F.

Let P be the largest value over all corners of these input hyperplanes, i.e.,

$$P = \max_{i=1}^{k} \left[\max_{j=1}^{2^n} (H_i(C_j)) \right]. \tag{3}$$

We refer to P as the peak value of the input hyperplanes. Let the peak value occur at corner $C_p = (X_1^*, X_2^*, \ldots, X_n^*)$, on hyperplane H_p, where H_p is one of the k input hyperplanes, so that

$$H_p(C_p) = P. \tag{4}$$

We will call H_p the peak plane, C_p the peak corner, and the point (C_p, P) the peak point of the k input hyperplanes.

Recall that the output hyperplane H_F acts as a ceiling to the k input hyperplanes at all corners but, at the same time, attempts to minimize the overestimation incurred at some corners. In this respect, we specify certain criteria that the output hyperplane should satisfy and will then describe our algorithm. These criteria are meant to reduce the overestimation error and to

make sure that the output hyperplane never underestimates the maximum value at any corner.

1) Output Hyperplane Criteria: We require the output hyperplane H_F to satisfy the following criteria.

1) For every corner C_j, for $1 \leq j \leq 2^n$, we require that

$$H_F(C_j) \geq \max_{i=1}^{k} (H_i(C_j)) \tag{5}$$

so that the output hyperplane should never underestimate the maximum value at any corner.

2) For every corner C_j, for $1 \leq j \leq 2^n$, we require that

$$H_F(C_j) \leq P \tag{6}$$

where P is the peak value defined previously. The purpose of this criterion is to limit the overestimation incurred by the output hyperplane.

3) Given that the peak point defined earlier is (C_p, P), we also require

$$H_F(C_p) = P \tag{7}$$

so that the output hyperplane does not incur overestimation at the peak corner.

In order to find an output hyperplane that meets these criteria, our approach consists of the following four major tasks to be performed: 1) finding the peak point over all input hyperplanes; 2) changing the origin; 3) raising the input hyperplanes; and 4) covering the raised hyperplanes with the output hyperplane. In what follows, we explain each of these operations and explain the procedures that we use to achieve these tasks. We also prove that these procedures indeed achieve the required tasks and that the combination of these four tasks results in a hyperplane that satisfies the criteria specified before.

A. Finding the Peak Point

Recall that the peak point (C_p, P) is such that P is the highest value achieved in the k input hyperplanes at all the corners, and $C_p = (X_1^*, X_2^*, \ldots, X_n^*)$ is the corner at which this value occurs. This point belongs to the peak plane H_p. An example of a peak point is the point $(+1, -1, 16)$ in Fig. 2.

In order to find the peak point, the highest value of every hyperplane and its corresponding corner are first found for each of the k input hyperplanes. For a given plane H_i, its highest value p_i can be found by using

$$p_i = \alpha_0^{(i)} + \sum_{j=1}^{n} \left| \alpha_j^{(i)} \right|. \tag{8}$$

The corner c_{pi} corresponding to this value can be easily found by setting $X_j = +1$ if $\alpha_j^{(i)} > 0$ and $X_j = -1$ if $\alpha_j^{(i)} < 0$. We then find

$$P = \max_{i=1}^{k} (p_i) \tag{9}$$

and the peak corner C_p is simply the corner corresponding to the highest among all the p_i values.

Finding the highest point of every hyperplane is of complexity $\mathcal{O}(n)$, and doing this for all k input-delay hyperplanes is $\mathcal{O}(kn)$. Finding the maximum of all these points is $\mathcal{O}(k)$, so that the overall complexity of finding the peak point is $\mathcal{O}(kn)$.

B. Changing the Origin

The next step is to change the origin of the system of coordinates in $(n+1)$-dimensional space such that the new origin is at the point $(C_p, 0)$. We also want to change the directions of some of the coordinate axes so that the new normalized process parameters in this system (Y_i) vary between zero and two. This transformation of the coordinate system is not absolutely required but is cheap and will make subsequent steps of the algorithm clearer and more understandable. Let us call the peak corner in the new system of coordinates C_p', where $C_p' = (0, 0, \ldots, 0)$; thus, the peak point in the modified system of coordinates becomes (C_p', P).

Let the transformed equations of the input hyperplanes, after modifying the system of coordinates, be as follows:

$$\begin{aligned} H_1' &= \beta_0^{(1)} + \beta_1^{(1)} Y_1 + \beta_2^{(1)} Y_2 + \cdots + \beta_n^{(1)} Y_n \\ H_2' &= \beta_0^{(2)} + \beta_1^{(2)} Y_1 + \beta_2^{(2)} Y_2 + \cdots + \beta_n^{(2)} Y_n \\ &\vdots \\ H_k' &= \beta_0^{(k)} + \beta_1^{(k)} Y_1 + \beta_2^{(k)} Y_2 + \cdots + \beta_n^{(k)} Y_n. \end{aligned} \tag{10}$$

Modifying the system of coordinates is a simple exercise in analytical geometry, and it can be shown that one can achieve it by replacing X_j with $-X_j^*(Y_j - 1)$ for $1 \leq j \leq n$, in each of the k input hyperplane equations. It is also easily shown that substituting X_j with $-X_j^*(Y_j - 1)$ in the equation of a hyperplane H_i in (2) results in

$$\beta_0^{(i)} = \alpha_0^{(i)} + \sum_{j=1}^{n} \alpha_j^{(i)} X_j^* \tag{11}$$

and, for $1 \leq j \leq n$, we have

$$\beta_j^{(i)} = -\alpha_j^{(i)} X_j^*. \tag{12}$$

These relations are used to find the expressions for the k input hyperplanes in the modified system of coordinates. For a single hyperplane, the complexity of this operation is $\mathcal{O}(n)$, and thus, for all the k hyperplanes, the complexity is $\mathcal{O}(kn)$.

1) Remarks: Without loss of generality, if the peak hyperplane is H_k', then it is easily shown that $\beta_0^{(k)} = P$ and that $\beta_j^{(k)} \leq 0$ for $1 \leq j \leq n$. To see this, recall that C_p' is the corner at the origin in the new coordinate system, i.e., $C_p' = (0, 0, \ldots, 0)$, so that the value of the constant term of the peak plane must be P. Moreover, if any $\beta_j^{(k)} > 0$, for some $1 \leq j \leq n$, then we can always find a point that is higher than (C_p', P) by setting the value of Y_j to two. This is a contradiction because no point has a delay that is higher than P among all of the input hyperplanes.

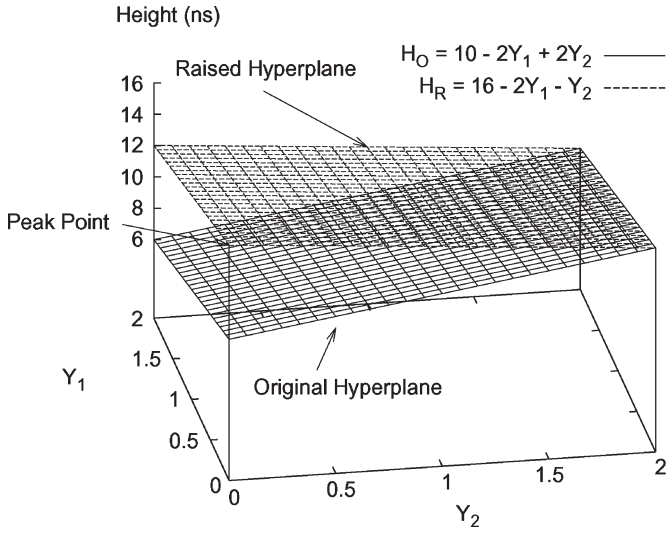

Fig. 3. Raising a hyperplane.

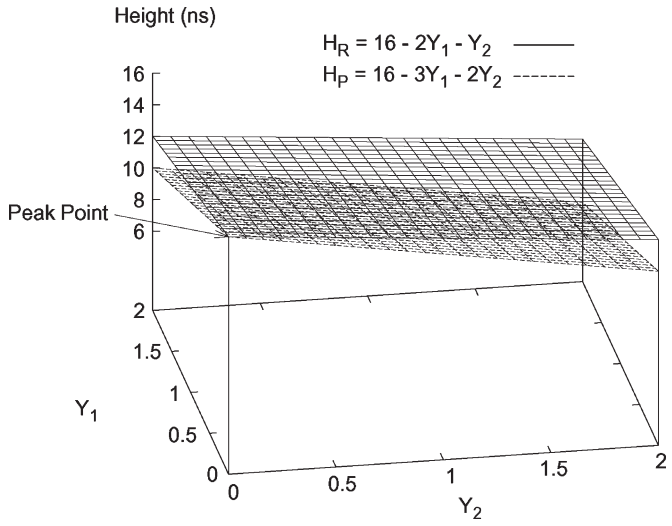

Fig. 4. Raised planes.

Likewise, if, for a hyperplane H_i' other than the peak plane, the highest point also corresponds to the peak corner C_p', then, by the same reasoning, each $\beta_j^{(i)} \leq 0$ for $1 \leq j \leq n$. Furthermore, in the case, when the highest point of a hyperplane H_i' corresponds to a corner other than C_p', then at least one $\beta_j^{(i)} \geq 0$. If this were not true, then the highest point in such a hyperplane would correspond to the peak corner C_p'.

C. Raising the Hyperplanes

We then perform an operation that we call "raising hyperplanes" on all of the input hyperplanes. Intuitively, the purpose of this step is to raise some corners of every hyperplane, by as little as possible, but by just enough to make it pass through the peak point at the peak corner. This will greatly facilitate the subsequent step of choosing an output hyperplane. As an example of this operation, one of the planes (H_O) of Fig. 2 is shown in both its original form and in its "raised" form in Fig. 3. Then, Fig. 4 shows both the peak plane H_P and the new raised plane H_R; both planes now pass through the peak point at the peak corner.

Raising a hyperplane is the most crucial and involved step in our algorithm. The three required criteria of Section III-A1 lead to three related criteria that our "raised" hyperplanes must meet. For a hyperplane H_i' in (10), let the corresponding "raised" hyperplane be H_i'', which is given by

$$H_i'' = \gamma_0^{(i)} + \gamma_1^{(i)} Y_1 + \gamma_2^{(i)} Y_2 + \cdots + \gamma_n^{(i)} Y_n. \tag{13}$$

1) Raised Hyperplane Criteria: For any hyperplane H_i' in (10), the criteria for its "raised" hyperplane H_i'' in (13) are as follows:

1) For every corner C_j', for $1 \leq j \leq 2^n$, we require that

$$H_i''\left(C_j'\right) \geq H_i'\left(C_j'\right) \tag{14}$$

so that a "raised" hyperplane never underestimates the value of the original hyperplane at any corner.

2) For every corner C_j', for $1 \leq j \leq 2^n$, we require that

$$H_i''\left(C_j'\right) \leq P \tag{15}$$

where P is the peak value of the k input hyperplanes. The purpose of this criterion is to limit the overestimation incurred by a raised hyperplane at corners.

3) Given that the peak point in the modified system of coordinates is (C_p', P), then we require that

$$H_i''\left(C_p'\right) = P \tag{16}$$

so that every "raised" hyperplane passes through the peak point.

It is easy to see that (16) leads to the requirement that the constant term in the equation of any "raised" hyperplane be P, since $C_p' = (0, 0, \ldots, 0)$, so that, for all $1 \leq i \leq k$, we have, as a first result

$$\gamma_0^{(i)} = P. \tag{17}$$

2) Procedure: Given the remarks in Section III-B1, we can classify our input hyperplanes into three classes. The first class contains only one hyperplane: the peak plane. The second class contains planes, other than the peak plane, whose highest points happen to be at the peak corner C_p'. The third class consists of those remaining hyperplanes whose highest points are at corners other than the peak corner. The steps taken to "raise" a hyperplane differ from one class to another, as considered in the following three cases. In each case, we will describe without proof the procedure applied in our algorithm. We then give a section in which the validity of all the steps is rigorously proven.

a) Case 1: This is the case when the hyperplane belongs to the first class, i.e., it is the peak plane. In this case, we select

$$\gamma_0^{(i)} = \beta_0^{(i)} = P \tag{18}$$

and, for $1 \leq j \leq n$

$$\gamma_j^{(i)} = \beta_j^{(i)}. \tag{19}$$

Thus, the hyperplane remains unchanged.

b) Case 2: This is the case when the hyperplane belongs to the second class, i.e., the highest point in this plane is achieved at the peak corner C'_p. In this case, and as shown in Section III-B1, $\beta_j^{(i)} \leq 0$, for $1 \leq j \leq n$. For this case, in order to "raise" hyperplane H'_i, our procedure is to only change the value of its constant term and of only one of its sensitivity coefficients $\beta_j^{(i)}$, where $1 \leq j \leq n$. There is some flexibility in the choice of exactly which coefficient to change. Best empirical results are obtained by choosing the largest coefficient, i.e., the least negative one. Assume, without loss of generality, that $\beta_1^{(i)} = \max_{j=1}^{n}(\beta_j^{(i)})$. Then, the "raised" hyperplane is obtained according to the following construction:

$$\gamma_0^{(i)} = P \tag{20}$$

$$\gamma_1^{(i)} = \frac{-P + \beta_0^{(i)} + 2\beta_1^{(i)}}{2} \tag{21}$$

and, for $2 \leq j \leq n$

$$\gamma_j^{(i)} = \beta_j^{(i)}. \tag{22}$$

Because the original plane $H'_i \leq P$ at all corners, then, at the corner $(2, 0, 0, \ldots, 0)$, $H'_i \leq P$ leads to $\beta_0^{(i)} + 2\beta_1^{(i)} \leq P$, and therefore, $\gamma_1^{(i)} \leq 0$, and so $\gamma_j^{(i)} \leq 0$, for $1 \leq j \leq n$.

c) Case 3: This is the case when the highest point of this plane corresponds to a corner other than the peak corner C'_p. In this case, and as we saw in Section III-B1, at least one $\beta_j^{(i)} \geq 0$, for $1 \leq j \leq n$. In this case, in order to "raise" a plane H'_i, we change the value of the constant term and of all the sensitivities with positive values in the expression for that hyperplane. Assume, without loss of generality, that $\beta_j^{(i)} \geq 0$, for $1 \leq j \leq \hat{n}$, where $\hat{n}$ is the number of positive sensitivities of H'_i. In this case, we formulate the "raised" hyperplane according to

$$\gamma_0^{(i)} = P \tag{23}$$

and, for $1 \leq j \leq \hat{n}$

$$\gamma_j^{(i)} = \frac{-P + \beta_0^{(i)} + \sum_{l=1}^{\hat{n}} 2\beta_l^{(i)}}{2\hat{n}} \tag{24}$$

and, for $\hat{n} + 1 \leq j \leq n$

$$\gamma_j^{(i)} = \beta_j^{(i)}. \tag{25}$$

As in the previous case, because $H'_i \leq P$ at all corners, then, by a judicious choice of a specific corner, we easily find that $-P + \beta_0^{(i)} + \sum_{j=1}^{\hat{n}} 2\beta_j^{(i)} \leq 0$. Thus, in this case as well, $\gamma_j^{(i)} \leq 0$, for $1 \leq j \leq n$.

3) Proof of Correctness: We will now prove that, for each of the three cases under consideration, the "raised" hyperplane meets the criteria specified in Section III-C1. Notice that, in all three cases, the "raised" hyperplane H''_i has a constant term $\gamma_0^{(i)} = P$, and $\gamma_j^{(i)} \leq 0$, for $1 \leq j \leq n$. Recall also that the peak corner is at the origin $C'_p = (0, 0, \ldots, 0)$; thus, for each of these cases

$$H''_i\left(C'_p\right) = \gamma_0^{(i)} = P. \tag{26}$$

Therefore, the "raised" hyperplane H''_i satisfies the third criterion of Section III-C1 for all the three cases.

Moreover, given that, for all the three cases, $\gamma_j^{(i)} \leq 0$, and because $0 \leq Y_j \leq 2$, for $1 \leq j \leq n$, then it is also straightforward to see, from (13), that, for all corners C_t, $1 \leq t \leq 2^n$

$$H''_i(C_t) \leq \gamma_0^{(i)} = P. \tag{27}$$

Thus, the "raised" hyperplane H''_i satisfies the second criterion of Section III-C1 for all the three cases in Section III-C2.

It remains to prove that, for all the three cases of Section III-C2, the "raised" hyperplane H''_i satisfies the first criterion of Section III-C1. Recall that this criterion is the requirement that the delay of the "raised" hyperplane H''_i, at any corner, be no less than the delay of the original hyperplane H'_i at the same corner.

We start with Case 1. In this case, the hyperplane H'_i is the peak plane, and it is not changed. Thus, the first criterion of Section III-C1 is trivially satisfied for this case.

We now consider Case 2. In this case, only one of the sensitivities of H'_i is changed to find H''_i, and we assumed, without loss of generality, that $\beta_1^{(i)}$ is the sensitivity term to be changed. Also, the constant term of the hyperplane, $\beta_0^{(i)}$, was changed by increasing its value to $\gamma_0^{(i)} = P$. Notice that the original value $\beta_0^{(i)} \leq P$ because this is not the peak plane. Thus, it is impossible for the "raised" plane to underestimate the delay at a corner C'_t which has $Y_1 = 0$. Therefore, it is enough to prove that H''_i does not underestimate the delay at any corner C'_t, where $Y_1 = 2$. Given such a corner $C'_t = (2, Y_2, \ldots, Y_n)$, we have

$$H''_i\left(C'_t\right) = P + 2\gamma_1^{(i)} + \gamma_2^{(i)}Y_2 + \cdots + \gamma_n^{(i)}Y_n \tag{28}$$

which, using (21) and (22), can be written as

$$H''_i\left(C'_t\right) = P + 2\left(\frac{-P + \beta_0^{(i)} + 2\beta_1^{(i)}}{2}\right) + \sum_{j=2}^{n} \beta_j^{(i)}Y_j \tag{29}$$

which easily reduces to

$$H''_i\left(C'_t\right) = \beta_0^{(i)} + 2\beta_1^{(i)} + \sum_{j=2}^{n} \beta_j^{(i)}Y_j = H'_i\left(C'_t\right). \tag{30}$$

Therefore, the first criterion of Section III-C1 is satisfied for every corner C'_t in this case.

We finally consider Case 3. In this case, all the positive sensitivities of H'_i are changed in order to arrive at H''_i, and we assumed, without loss of generality, that the only positive sensitivities are $\beta_j^{(i)} \geq 0$, for $1 \leq j \leq \hat{n}$. In addition, the constant term of the hyperplane expression $\beta_0^{(i)}$ is changed, and its value is increased to P. Thus, it is impossible for the "raised" hyperplane to underestimate the value at a corner C'_t which has $Y_j = 0$, for $1 \leq j \leq \hat{n}$. Therefore, it is enough to prove that H''_i

does not underestimate the value at any corner C'_t for which at least one $Y_j = 2$, for $1 \leq j \leq \hat{n}$.

It would suffice to prove that H''_i does not underestimate the value at corners C'_t, where all $Y_j = 2$, for $1 \leq j \leq \hat{n}$. This is because, with $\gamma_j^{(i)} \leq 0$ and $\beta_j^{(i)} \geq 0$, for $1 \leq j \leq \hat{n}$, if, for such a corner C'_t, $H''_i(C'_t) \geq H'_i(C'_t)$, then changing any Y_j, for $1 \leq j \leq \hat{n}$, from two to zero would only increase the value of $H''_i(C'_t)$ and decrease the value of $H'_i(C'_t)$, thus maintaining the inequality. Now, given such a corner C'_t, where $Y_j = 2$, for $1 \leq j \leq \hat{n}$, we have

$$H''_i\left(C'_t\right) = P + 2\sum_{j=1}^{\hat{n}} \gamma_j^{(i)} + \sum_{j=\hat{n}+1}^{n} \gamma_j^{(i)} Y_j \tag{31}$$

which, using (24) and (25), can be written as

$$H''_i\left(C'_t\right) = P + 2\sum_{j=1}^{\hat{n}} \left(\frac{-P + \beta_0^{(i)} + 2\sum_{l=1}^{\hat{n}} \beta_l^{(i)}}{2\hat{n}} \right) + \sum_{j=\hat{n}+1}^{n} \beta_j^{(i)} Y_j \tag{32}$$

which easily reduces to

$$H''_i\left(C'_t\right) = \beta_0^{(i)} + 2\sum_{j=1}^{\hat{n}} \beta_j^{(i)} + \sum_{j=\hat{n}+1}^{n} \beta_j^{(i)} Y_j = H'_i\left(C'_t\right). \tag{33}$$

Therefore, the first criterion in Section III-C1 is satisfied for any corner C'_t in this Case 3.

4) Complexity: "Raising" one hyperplane requires the examination of each of its sensitivities and might involve the modification of these sensitivities according to prespecified equations. Thus, "raising" one hyperplane is of complexity $\mathcal{O}(n)$, and performing this operation for all the k input hyperplanes is $\mathcal{O}(kn)$.

D. Covering the Raised Hyperplanes

We now have a set of raised hyperplanes shown in (34). These hyperplanes satisfy the three criteria for raised hyperplanes in Section III-C1, and our goal now is to find an output hyperplane that satisfies the criteria of Section III-A

$$\begin{aligned} H''_1 &= P + \gamma_1^{(1)} Y_1 + \gamma_2^{(1)} Y_2 + \cdots + \gamma_n^{(1)} Y_n \\ H''_2 &= P + \gamma_1^{(2)} Y_1 + \gamma_2^{(2)} Y_2 + \cdots + \gamma_n^{(2)} Y_n \\ &\vdots \\ H''_k &= P + \gamma_1^{(k)} Y_1 + \gamma_2^{(k)} Y_2 + \cdots + \gamma_n^{(k)} Y_n. \end{aligned} \tag{34}$$

Let the expression for the desired output hyperplane be

$$H'_F = \lambda_0 + \lambda_1 Y_1 + \lambda_2 Y_2 + \cdots + \lambda_n Y_n. \tag{35}$$

Our algorithm finds the output plane based on

$$\lambda_0 = P \tag{36}$$

and, for $1 \leq j \leq n$

$$\lambda_j = \max_{i=1}^{k} \left(\gamma_j^{(i)} \right). \tag{37}$$

After finding the equation of the output hyperplane in the modified system of coordinates, we change our system back to the original one. This can be easily done by reversing the transformations performed before. This yields the required equation of the output hyperplane.

1) Proof of Correctness: We will prove that the output hyperplane found earlier satisfies the criteria set in Section III-A. In our discussion, H'_F refers to the equation of the output hyperplane in the modified system of coordinates, whereas H_F refers to the equation of this hyperplane in the original system.

First of all, because $\lambda_0 = P$, it follows that $H'_F(C'_p) = P$, and equivalently, $H_F(C_p) = P$; thus, the third criterion of Section III-A is satisfied. Because $\gamma_j^{(i)} \leq 0$, $1 \leq j \leq n$, for all raised hyperplanes, then $\lambda_j \leq 0$, for $1 \leq j \leq n$. Given that $0 \leq Y_j \leq 2$, for $1 \leq j \leq n$, it is easy to see that, for any corner C'_t in the modified system of coordinates, $1 \leq t \leq 2^n$, $H'_F(C'_t) \leq P$, and equivalently, $H_F(C_t) \leq P$ for any corner C_t in the original system of coordinates. Thus, the output hyperplane satisfies the second criterion of Section III-A.

It remains to be proven that the output hyperplane satisfies the first criterion of Section III-A. Recall that this criterion is that the output hyperplane should not underestimate the maximum value at any corner. From the first criterion of Section III-C1, we know that, for any "raised" plane H''_i and at any corner C'_t in the modified system of coordinates, we have $H''_i(C'_t) \geq H'_i(C'_t)$, where H'_i is the hyperplane that was "raised" in order to create H''_i. From (36), (37), and the fact that $0 \leq Y_j \leq 2$, for $1 \leq j \leq n$, we can deduce that, for all corners (C'_t), $1 \leq t \leq 2^n$, in the modified system of coordinates

$$H'_F\left(C'_t\right) \geq \max_{i=1}^{k} \left(H''_i\left(C'_t\right) \right). \tag{38}$$

Thus, by using the first criterion for "raised hyperplanes" of Section III-C1, we can easily deduce that

$$H'_F(C'_t) \geq \max_{i=1}^{k} \left(H'_i\left(C'_t\right) \right) \tag{39}$$

and, equivalently, that, for all corners (C_t), $1 \leq t \leq 2^n$, in the original system of coordinates

$$H_F(C_t) \geq \max_{i=1}^{k} \left(H_i(C_t) \right). \tag{40}$$

Therefore, the output hyperplane also satisfies the first criterion of Section III-A.

2) Complexity: Finding each value of λ takes a time that is linear in k, and thus finding all n values is of complexity $\mathcal{O}(nk)$. An examination of the complexities of all the steps involved in finding the maximum of a set of k hyperplanes reveals that the computational complexity of our "max" operation is $\mathcal{O}(nk)$.

IV. Method at Logic Stage Level

We now present our method for delay and slew hyperplane propagation through a logic stage. A logic stage is defined as a logic cell and its output interconnect structure. In this section, we present a method, where, given the signal-arrival-time and slew hyperplanes at the inputs of a logic stage, the signal-arrival-time and slew hyperplanes at its outputs can be found. The output hyperplanes (delay and slew) become inputs for the analysis of downstream stages. In what follows, we first present our method for delay and slew propagation through the logic cell of the logic stage and then through its interconnect structure, and in both cases, a separate analysis is presented for each of delay and slew propagation. In our analysis, we assume that the logic cell and its output interconnect have already been characterized, so that the delay, slew, and variability introduced by the cell and the interconnect structure can be accounted for.

A. Propagation in Logic Cell

Our approach extends the traditional timing model of a logic cell to find hyperplane expressions for the signal arrival time and slew at the output of every timing arc of the cell, given the signal-arrival-time and slew hyperplanes at the input of the timing arc and the load-capacitance hyperplane at its output. Using our max operation for hyperplanes from Section III, we then find hyperplane expressions for the signal arrival time and slew at the output of the logic cell. As mentioned before, the signal-arrival-time and slew hyperplanes at the inputs of the logic stage are assumed to be known; thus, the delay and slew information at the input of the logic cell is available. Moreover, using the methods of [1] and [11], we find a hyperplane expression for the "effective" load capacitance of the cell.

Note that, for a given input, the delay introduced by the cell for a rising signal at the output differs from the delay in the case of a falling signal, and the same is true for slew. Therefore, in our analysis, we distinguish between the rising and falling timing arcs of the cell inputs, and as a result, each input can have two timing arcs. Whether one wants to take into account rising timing arcs or falling timing arcs, or both, in the computations of the output hyperplanes depends on the objective of the analysis. Suppose that our cell has u inputs. If one is interested in, for example, the output arrival time and slew for a rising output, then only the rising timing arcs are considered, and the number of timing arcs under consideration would be u. Now, if one simply wants the worst case output arrival time and slew regardless of signal direction, then both rising and falling timing arcs would be considered, leading to a total of $2u$ timing arcs. In any case, and in order to show a generic analysis, we will assume that the cell has k timing arcs, where k could either be u or $2u$.

1) Delay Propagation: The timing model for a logic cell provides a means (typically a table) to find the signal arrival time at the output of every timing arc, for a given input-signal slew, output-capacitive loading, and input-signal arrival time. The output arrival time of the logic cell is then typically computed as the maximum of output arrival times of those timing arcs

$$D_{\text{out}} = \max_{i=1}^{k}(D_{\text{arc},i}). \tag{41}$$

Focusing on a single timing arc, let its input slew be S_{in}, its input arrival time be D_{in}, and its output (effective) capacitance load be C_{out}. The output arrival time D_{arc} of this timing arc can be found as the sum of the input arrival time and the delay introduced by the logic cell for this timing arc. Let function $F(\cdot)$ represent the functional dependence of the delay introduced by the logic cell on its many variables; thus, D_{arc} can be expressed as follows:

$$D_{\text{arc}} = D_{\text{in}} + F(S_{\text{in}}, C_{\text{out}}, X_1, \ldots, X_n). \tag{42}$$

We assume a linear model for S_{in}, C_{out}, and D_{in} in terms of process variables

$$\begin{aligned} S_{\text{in}} &= S_{\text{in}}^{\text{nom}} + \sum_{j=1}^{n} \alpha_j X_j \\ C_{\text{out}} &= C_{\text{out}}^{\text{nom}} + \sum_{j=1}^{n} \beta_j X_j \\ D_{\text{in}} &= D_{\text{in}}^{\text{nom}} + \sum_{j=1}^{n} \delta_j X_j \end{aligned} \tag{43}$$

where $S_{\text{in}}^{\text{nom}}$, $C_{\text{out}}^{\text{nom}}$, and $D_{\text{in}}^{\text{nom}}$ represent the nominal values of input slew, output load, and input arrival time, respectively. We define the nominal point in the domain of the function $F(\cdot)$ to be the point $(S_{\text{in}}^{\text{nom}}, C_{\text{out}}^{\text{nom}}, 0, 0, \ldots, 0)$. Let $D_{g,\text{arc}}^{\text{nom}}$ be the value of the cell-introduced delay at the nominal point for this timing arc. Thus

$$D_{g,\text{arc}}^{\text{nom}} = F\left(S_{\text{in}}^{\text{nom}}, C_{\text{out}}^{\text{nom}}, 0, 0, \ldots, 0\right). \tag{44}$$

Using a first-order Taylor series expansion of $F(\cdot)$ around the $D_{g,\text{arc}}^{\text{nom}}$, we can approximate the expression in (42) as follows:

$$\begin{aligned} D_{\text{arc}} = D_{\text{in}}^{\text{nom}} &+ \sum_{j=1}^{n} \delta_j X_j + D_{g,\text{arc}}^{\text{nom}} + \left.\frac{\partial F}{\partial S_{\text{in}}}\right|_{\text{nom}} \\ &\times (S_{\text{in}} - S_{\text{in}}^{\text{nom}}) + \left.\frac{\partial F}{\partial C_{\text{out}}}\right|_{\text{nom}} \times (C_{\text{out}} - C_{\text{out}}^{\text{nom}}) \\ &+ \sum_{j=1}^{n} \left(\left.\frac{\partial F}{\partial X_j}\right|_{\text{nom}} \times X_j \right) \end{aligned} \tag{45}$$

where the partial derivatives are taken at the nominal point. Given (43), this can be further simplified, leading to

$$D_{\text{arc}} = D_{\text{arc}}^{\text{nom}} + \sum_{j=1}^{n} \omega_j X_j \tag{46}$$

where

$$D_{\text{arc}}^{\text{nom}} = D_{\text{in}}^{\text{nom}} + D_{g,\text{arc}}^{\text{nom}} \tag{47}$$

$$\omega_j = \delta_j + \left.\frac{\partial F}{\partial S_{\text{in}}}\right|_{\text{nom}} \times \alpha_j + \left.\frac{\partial F}{\partial C_{\text{out}}}\right|_{\text{nom}} \times \beta_j + \left.\frac{\partial F}{\partial X_j}\right|_{\text{nom}}. \tag{48}$$

The value of the partial derivative $(\partial F/\partial X_j)|_{\text{nom}}$ can be found by characterizing a logic cell at the nominal point to find its delay sensitivity to process parameter X_j. However, it is not practical to characterize a cell for every possible nominal input slew and effective output capacitance combination. In this paper, we propose the use of the method of "relative sensitivities," in which the sensitivity in a single input-slew and output-capacitance context can be used to approximate the sensitivity in any circuit context. This method is described in detail in Section VI-C. However, our hyperplane propagation method does not require the use of this approximation, and one could use tables to store values of $(\partial F/\partial X_j)|_{\text{nom}}$, as is usually done for delays and slews.

After the hyperplanes $D_{\text{arc},i}$ are found for $1 \leq i \leq k$, our method for finding the maximum of a set of hyperplanes is used to find D_{out} as seen in (41).

2) Slew Propagation: There are many methods for slew propagation in logic gates. One method, sometimes used in practice, involves finding the slews at the outputs of all timing arcs of the cell and propagating their maximum value. However, this method, sometimes referred to as worst slew propagation, results in an overly pessimistic analysis [14]. At the other end of the spectrum, some traditional approaches find the timing arc with the maximum signal arrival time at its output and propagate the slew at the output of this timing arc to the output of the cell. This leads to an overly optimistic analysis of downstream logic stages. In our case, we use an extension of the general method described in [8] and [14] independently. In this method, the slew at the output of every timing arc is first found, as in other methods. These slews are then adjusted or "tuned" (in order to control the pessimism of the analysis), as will be explained in detail shortly. This leads to a modified set of arc slews, $S_{\text{arc},i}$ for $1 \leq i \leq k$. The output slew of the logic cell is then computed by finding the maximum of these modified timing arc slews

$$S_{\text{out}} = \max_{i=1}^{k}(S_{\text{arc},i}). \tag{49}$$

For a single timing arc, the timing model of a logic cell provides a means to find the signal slew at output, given input slew and output capacitive loading. However, the method in [8] and [14] adds a "tuning" factor to this slew. Let $H(\cdot)$ be the function used to find the slew at the output of a specific timing arc. Let v be a given nonnegative parameter; the modified slew at the output of the timing arc, S_{arc}, is written as [8], [14]

$$S_{\text{arc}} = H(S_{\text{in}}, C_{\text{out}}, X_1, X_2, \ldots, X_n) + v(D_{\text{arc}} - D_{\text{out}}) \tag{50}$$

where S_{in} is the input-signal slew hyperplane, C_{out} is the output (effective) capacitance, D_{arc} is the arrival-time hyperplane at the output of the timing arc, and D_{out} is the arrival-time hyperplane at the output of the cell. D_{arc} and D_{out} are found, as shown in Section IV-A1. S_{in} and C_{out} are linear functions of process parameters, as shown in (43). The value of parameter v depends on the desired level of pessimism; however, the method to choose an optimal value for this parameter is beyond the scope of this paper and is covered thoroughly in [14]. In any case, and whatever the chosen value, this does not affect the applicability of our method. That being said, it is worth noting that, for testing purposes, we use $v = 2$. This is equivalent to propagating the latest time that a signal passes the 90% threshold (10% for falling signals) [14]. As we did in the case of delays, we now find a hyperplane expression for S_{arc}. Let $S_{\text{arc}}^{\text{nom}}$ be the value of the arc output slew at the nominal point $(S_{\text{in}}^{\text{nom}}, C_{\text{out}}^{\text{nom}}, 0, 0, \ldots, 0)$. Thus

$$S_{\text{arc}}^{\text{nom}} = H\left(S_{\text{in}}^{\text{nom}}, C_{\text{out}}^{\text{nom}}, 0, 0, \ldots, 0\right) + v\left(D_{\text{arc}}^{\text{nom}} - D_{\text{out}}^{\text{nom}}\right). \tag{51}$$

Recall from (46) that the sensitivity of D_{arc} to X_j is ω_j. Also, let σ_j be the sensitivity of D_{out} to process parameter X_j. Using a first-order Taylor series expansion of $H(\cdot)$ around the $S_{\text{arc}}^{\text{nom}}$, we can approximate the expression in (50) as follows:

$$\begin{aligned} S_{\text{arc}} = {} & S_{\text{arc}}^{\text{nom}} + \left.\frac{\partial H}{\partial S_{\text{in}}}\right|_{\text{nom}} \times (S_{\text{in}} - S_{\text{in}}^{\text{nom}}) \\ & + \left.\frac{\partial H}{\partial C_{\text{out}}}\right|_{\text{nom}} \times (C_{\text{out}} - C_{\text{out}}^{\text{nom}}) \\ & + \sum_{j=1}^{n}\left(\left.\frac{\partial H}{\partial X_j}\right|_{\text{nom}} \times X_j\right) + v\sum_{j=1}^{n}(\omega_j - \sigma_j)X_j. \end{aligned} \tag{52}$$

Thus, by using (43), (52) can be written as

$$S_{\text{arc}} = S_{\text{arc}}^{\text{nom}} + \sum_{j=1}^{n}\gamma_j X_j \tag{53}$$

where

$$\begin{aligned} \gamma_j = {} & \left.\frac{\partial H}{\partial S_{\text{in}}}\right|_{\text{nom}} \times \alpha_j + \left.\frac{\partial H}{\partial C_{\text{out}}}\right|_{\text{nom}} \times \beta_j \\ & + \left.\frac{\partial H}{\partial X_j}\right|_{\text{nom}} + v(\omega_j - \sigma_j). \end{aligned} \tag{54}$$

The value of $(\partial H/\partial X_j)|_{\text{nom}}$ can also be found by characterizing the gate at the nominal input-slew and output-capacitance context it is in. However, in this case, we also propose the use of the method of "relative sensitivities" to approximate its value.

After finding the hyperplanes $S_{\text{arc},i}$ for all of the timing arcs of the cell, we use our method for finding the maximum of a set of hyperplanes to find S_{out} as seen in (49).

3) Complexity: Delay and slew hyperplane propagation through the logic cell requires finding the maximum of the set of k input-delay hyperplanes and the maximum of the set of k modified input-slew hyperplanes. The total complexity of this operation is thus $\mathcal{O}(kn)$.

B. Delay and Slew Propagation in Interconnect

In order to be able to propagate delay and slew hyperplanes to subsequent logic stages, we must first account for the delay, slew, and variability introduced by the interconnect structure of the current logic stage. When performing timing analysis, interconnect structures are typically modeled as RC trees. Process parameter variations cause variability in the values of resistances and capacitances of an interconnect structure,

and in this section, we describe a method that can be used to account for the resulting variability in the delays and slews of signals traveling through interconnect. Let the logic stage (and hence the interconnect structure) have f fan-out nets. In what follows, we present a method used to find the delay and slew hyperplanes at each of these f nodes, given the arrival time and slew hyperplanes at the input of the interconnect RC tree.

1) Delay Propagation: The delay of a signal traversing an interconnect structure is affected by the slew rate of the signal at the input of that interconnect structure. In [4], the delay D_{node} required for a signal to travel from the input of an interconnect RC tree to a particular fan-out node, for a ramp input signal, is described as

$$D_{\text{node}} = (1-\alpha)E_{\text{step}} + \alpha D_{\text{step}} \tag{55}$$

where E_{step} is the Elmore delay metric of the step input response at that node and D_{step} is its exact step input response delay. D_{step} is usually approximated by methods such as the D2M metric, or can even be found by using HSPICE, and α is given by

$$\alpha = \left(\frac{2m_2 - m_1^2}{2m_2 - m_1^2 + S_{\text{in-tree}}^2}\right)^{\frac{5}{2}} \tag{56}$$

where $S_{\text{in-tree}}$ is the input-signal transition time (or slew), and m_1 and m_2 are the first- and second-order moments of the step input response at that node, respectively. In our case, $S_{\text{in-tree}}$ is the slew at the output of the logic cell, which we model to be linearly dependent on process parameter variations as shown in Section IV-A2. Moreover, E_{step}, D_{step}, m_1, and m_2 are also assumed to be linearly sensitive to process parameter variations. The step response moments can be written as follows:

$$m_1 = m_1^{\text{nom}} + \sum_{j=1}^{n} s_j^{(1)} X_j \tag{57}$$

$$m_2 = m_2^{\text{nom}} + \sum_{j=1}^{n} s_j^{(2)} X_j. \tag{58}$$

The sensitivities of these moments, of E_{step}, and of D_{step} to process parameter variations can be obtained by using the method described in [10]. It is obvious from (55) that D_{node} is not an affine linear function in process parameter variations. However, in this paper, we find such a linear approximation by applying a first-order Taylor series expansion of (55) around the nominal point as proposed in [10]. After finding a hyperplane expression for D_{node}, we add to it the signal arrival time at the input of the interconnect structure to get the signal-arrival-time hyperplane at that fan-out node. Results in [10] show that this approximation of D_{node} is accurate.

2) Slew Propagation: In order to find the slew of a signal at a node of an interconnect RC tree, moment analysis (for step input) is usually performed, and a 10%–90% metric S_{step} is found for every fan-out node of the RC interconnect structure. A method for computing such a metric is given in [4]. Also, the authors of [4] show that, for the more realistic case of a ramp input signal, a node signal slew rate S_{node} can be computed by using the following expression:

$$S_{\text{node}} = \sqrt{S_{\text{step}}^2 + S_{\text{in-tree}}^2} \tag{59}$$

where $S_{\text{in-tree}}$ is the slew of the input signal of the interconnect structure and S_{step} is defined earlier. In our case, we model $S_{\text{in-tree}}$ as a hyperplane, as mentioned in Section IV-B1, and S_{step} as

$$S_{\text{step}} = S_{\text{step}}^{\text{nom}} + \sum_{j=1}^{n} \lambda_j X_j \tag{60}$$

where λ_j, $1 \le j \le n$, are the sensitivities of the step slew metric to process parameter variations. These sensitivities are found by using the results of [10] and performing a first-order Taylor series expansion. It is obvious from (59) that S_{node} is not linearly dependent on process parameter variations, so we describe a linear approximation for it. Let $S_{\text{node}}^{\text{nom}}$ be the value of S_{node} at the nominal point, i.e.,

$$S_{\text{node}}^{\text{nom}} = \sqrt{\left(S_{\text{step}}^{\text{nom}}\right)^2 + \left(S_{\text{in-tree}}^{\text{nom}}\right)^2}. \tag{61}$$

Using a first-order Taylor series expansion of (59), we can write

$$S_{\text{node}} = S_{\text{node}}^{\text{nom}} + \sum_{i=j}^{n} \psi_j X_j \tag{62}$$

where

$$\psi_j = \left(\frac{\partial}{\partial X_j}\sqrt{S_{\text{step}}^2 + S_{\text{in-tree}}^2}\right)\Bigg|_{\text{nom}} = \frac{S_{\text{step}}^{\text{nom}}\lambda_j + S_{\text{in-tree}}^{\text{nom}}\alpha_j}{S_{\text{node}}^{\text{nom}}}. \tag{63}$$

Note that the S_{step} hyperplane is not a true upper bound on the actual step response slews at every process corner. Thus, the linear approximation that we found for S_{node} will not be an upper bound on actual response slews at various process corners.

As we did in the case of delays, and in order to verify the accuracy of the approximation in (62), we performed corner analysis and ran our algorithm on circuit "c432" of the ISCAS '85 benchmark suite. At every node of every interconnect structure, we compared the maximum slew as predicted by the slew hyperplane to maximum corner slew found by using corner analysis. Fig. 5 shows the percentage errors between maximum corner slews as approximated by the slew hyperplane in (62) and as found by corner analysis at these RC-tree nodes. This histogram shows that, in most instances, the slew hyperplane approximates the maximum corner slew in a reasonably accurate manner.

3) Complexity: Finding the nominal delay and slew at the output node of an interconnect structure takes constant time irrespective of n, the number of process parameters under consideration. Moreover, finding the sensitivity of delay or slew to a particular process parameter at a given node takes constant time, and as a result, finding the delay and slew hyperplanes at a given interconnect node is of computational complexity $\mathcal{O}(n)$. Therefore, for an interconnect structure with f fan-out nets, the

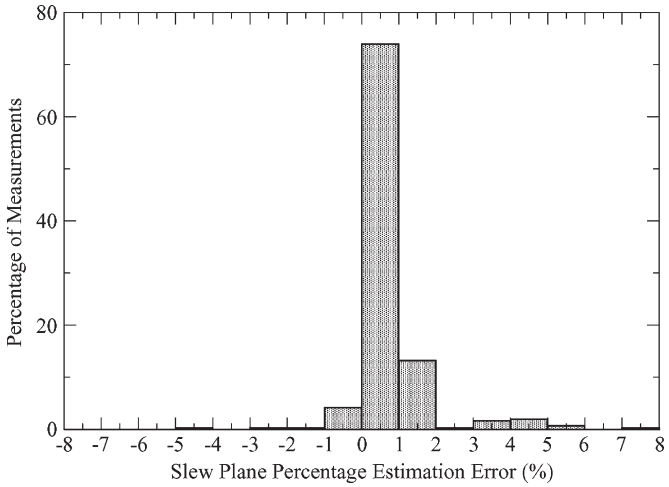

Fig. 5. Percentage error of slew plane corner estimation for circuit c432 of ISCAS '85.

total computational complexity for finding the delay and slew hyperplanes at all its fan-out nodes is $\mathcal{O}(fn)$.

C. Overall Complexity for a Logic Stage

An investigation of all the operations that our method requires at a logic stage shows that the total complexity for a single logic stage is $\mathcal{O}(kn + fn)$, where k could either be the number of inputs of the cell u or twice that number, f is the number of fan-out nets, and n is the number of process parameters being considered. Thus, the complexity of our method at the logic stage can be written as $\mathcal{O}(un + fn)$. Because, for obvious technology reasons, both u and f are bounded by some small fixed number, such as, perhaps, ten, and because they do not scale with the size of the problem, then the overall complexity for analysis of one logic stage is $\mathcal{O}(n)$.

V. Method at Circuit Level

As stated earlier, our approach is quite similar to traditional STA at the circuit level. Thus, in order to find the maximum delay and slew of the circuit under process variations, we apply our method to logic stages whose input delay and slew information (hyperplanes) is available and then propagate the resulting delay and slew hyperplanes to subsequent logic stages. This process is repeated until we get the delay and slew hyperplanes at the primary output nodes of the circuit. In order to find the maximum circuit delay and slew, we add a dummy logic stage to our timing graph such that the primary outputs of the circuit form its inputs and that no delay or slew is introduced by this stage. Our algorithm is now applicable one final time to give a single hyperplane for the overall circuit delay and another for circuit slew. It is then straightforward to find the maximum values of delay and slew that these planes can produce. The complexity of this step is $\mathcal{O}(zn)$, where z is the number of primary output nodes of the circuit. If a circuit has m stages in total, and because the analysis of a single stage is $\mathcal{O}(n)$, and because $z \leq m$, then the overall complexity of STA for the whole circuit, covering all 2^n process corners, is only $\mathcal{O}(mn)$. Thus, for a given fixed circuit size m, the complexity grows linearly with the number of variable process parameters n.

VI. Implementation

As is to be expected, our algorithm requires a precharacterized cell library and some variational model of interconnect delay.

A. Logic-Cell Characterization

We constructed and characterized a small CMOS cell library in 90-nm technology, and these cells were used as the standard cell library for our testing. For every input arc in every logic cell, the rise and fall delays and slew rates of the cell were computed, using HSPICE, for a number of input slew (slope) rates and load capacitances. The results were stored in tables called nominal delay tables and nominal slew tables, where, given an input and its slew rate (slope), and the load capacitance on the cell, the delay and slew can be computed through interpolation between appropriate values from the delay and slew tables, respectively. This is the standard modern approach for modeling the delays and slews of logic cells [12].

Then, the sensitivities of the delays and slews were found for variations in four process parameters: the NMOS threshold voltage (ΔV_{tn}), the NMOS channel length (ΔL_n), the PMOS threshold voltage (ΔV_{tp}), and the PMOS channel length (ΔL_p). The sensitivities to every process parameter were found by fixing all other process parameters at their nominal values, choosing a number of values for this parameter, and finding the delay and slew of the cell in each case using HSPICE. After that, linear regression was performed on the resulting values to find the unnormalized sensitivities of cell delays and slews to the process parameter. The sensitivities were then normalized so that the parameters vary between -1 and 1.

For every input arc, this characterization is performed for a transition in the input which causes a rise in the output signal and for a transition which causes a fall in the output signal. Thus, for the case of, for example, delay, every input arc has a rise delay hyperplane that uses rise sensitivities and has a fall delay hyperplane that uses fall sensitivities, and the same is true for slew rates. It is worth noting that median values were chosen for the input slew rates and load capacitances in the characterization process and that we keep a record of the nominal delay and slew of the cell for these slew and load capacitance values. The significance of this point will become clear shortly.

B. Interconnect Characterization

As mentioned previously, an interconnect fan-out structure is described as a tree of lumped resistances and capacitances, i.e., an RC tree. Our test circuits were arbitrarily placed and routed. For all interconnect RC trees, we let values of resistances lie between 100 and 300 Ω and values of capacitances lie between 15 and 25 fF. One is given the sensitivities of every resistor and capacitor to the relevant process parameters, as in [10]. From this, one finds the nominal delay and slew at every

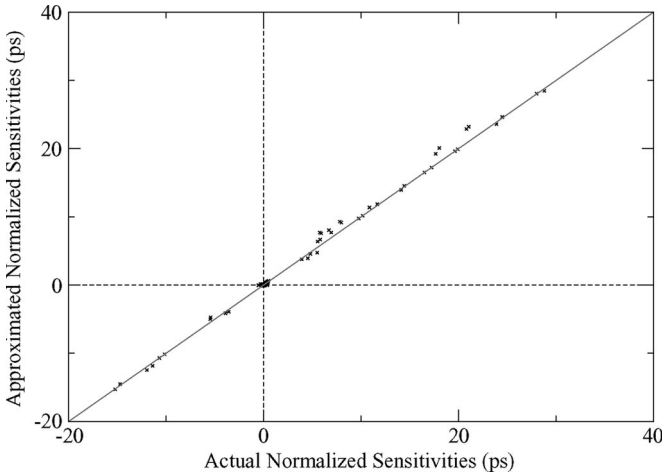

Fig. 6. Approximated versus actual delay sensitivities of a NAND gate.

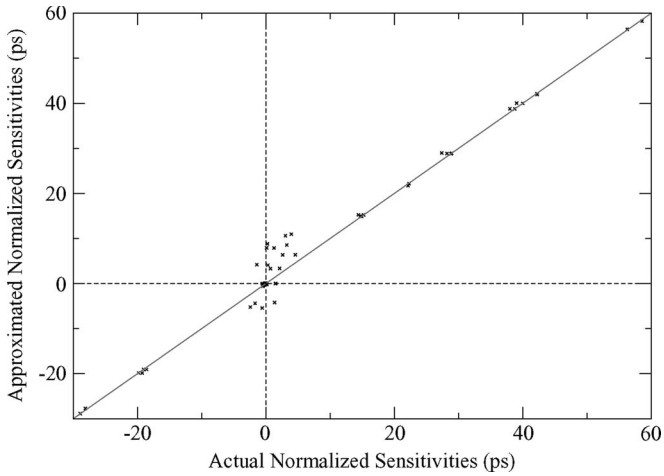

Fig. 7. Approximated versus actual slew sensitivities of a NAND gate.

node of the tree, as well as the sensitivities of the delay and slew at every node to the same process parameters by using the methods described in Section IV-B. The process parameters of interest in this paper, as in [10], are the metal width (ΔW), the metal thickness (ΔT), and the interlayer dielectric thickness (ΔH). For the test cases considered in this paper, we limited the effect that a process parameter can have on the value of a resistor or a capacitor to no more than 10%.

C. Relative Sensitivities

The sensitivities to physical parameters, which were measured during cell characterization (with some median input slope and output load applied), need to be modified according to the context of the cell in the given circuit, i.e., according to the actual slope and load presented to the cell in the circuit. This is a subtle point, which we have found that it can be overcome with good accuracy by a straightforward scaling operation. Consider the case of logic-cell delay sensitivities. For a particular timing arc of a logic cell in a single nominal characterization context $(S_{\text{in}}^{\text{ch}}, C_{\text{out}}^{\text{ch}})$, let the nominal output delay be D_0 and the delay sensitivity to process parameter X_j be δ_j. In a circuit context, where the nominal delay of this timing arc is, for example, $D_{g,\text{arc}}^{\text{nom}}$, we approximate the sensitivity, δ_j', of this timing arc delay to X_j as follows:

$$\delta_j' \approx \frac{D_{g,\text{arc}}^{\text{nom}}}{D_0}\delta_j. \tag{64}$$

We found this approximation to be simple and reasonably accurate. Fig. 6 shows a comparison of actual normalized sensitivities of a gate delay to different process parameters on the one hand and these sensitivities as predicted by our method on the other hand. This figure shows that our method approximates sensitivities quite accurately.

We also use the method of relative sensitivities in the case of slews to approximate the value of sensitivities in a particular circuit context. Let the nominal slew at the output of a timing arc of the cell be S_0, and its sensitivity to process parameter X_j be α_j, in the nominal characterization context $(S_{\text{in}}^{\text{ch}}, C_{\text{out}}^{\text{ch}})$. For a circuit context where the nominal slew at the output of the timing arc is $S_{\text{arc}}^{\text{nom}}$, the sensitivity α_j' of the timing arc slew to X_j is approximated by using

$$\alpha_j' \approx \frac{S_{\text{arc}}^{\text{nom}}}{S_0}\alpha_j. \tag{65}$$

Fig. 7 shows a comparison of slew sensitivities estimated using this method to actual characterized slew sensitivities. This figure shows that the error incurred when using this method is relatively low.

TABLE I
OUR APPROACH VERSUS THE CORNER APPROACH

ISCAS-85 Circuit	Corner Approach Max. Delay	Our Approach Max. Delay	Percentage Error (%)
c432	10.27 ns	10.21 ns	−0.54%
c499	4.75 ns	4.78 ns	0.53%
c880	9.17 ns	9.25 ns	0.83%
c1355	8.76 ns	8.84 ns	0.92%
c1908	12.10 ns	12.07 ns	−0.25%
c2670	12.31 ns	12.26 ns	−0.43%
c3540	15.20 ns	15.07 ns	−0.87%
c5315	14.72 ns	14.73 ns	0.10%
c6288	37.24 ns	37.55 ns	0.84%
c7552	12.53 ns	12.66 ns	1.00%

VII. RESULTS

We ran our algorithm on all circuits of the ISCAS '85 benchmark suite [3], mapped to our 90-nm cell library. In order to test the accuracy of our approach, we compared the maximum delay of every circuit as computed by our algorithm to the maximum delay computed by finding the delay of the circuit at every process corner (process variables vary between −10% and 10% around their nominal values). Note that, when finding the delay of a circuit at a process corner, characterized sensitivities are used to compute the delays of individual logic cells. The results and comparison between the two approaches are shown in Table I, which shows that our approach predicts the maximum delay of these circuits quite accurately. The bar diagrams in Figs. 8 and 9 show these same results, with three bars for every circuit. The first bar is found by running traditional STA for every process corner and recording the

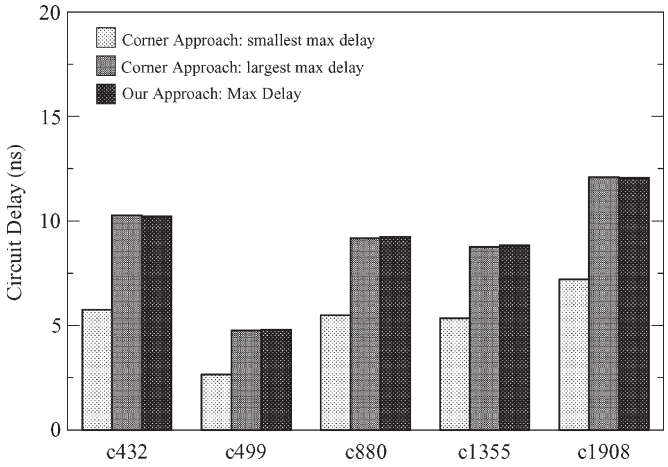

Fig. 8. Maximum delays for smaller circuits.

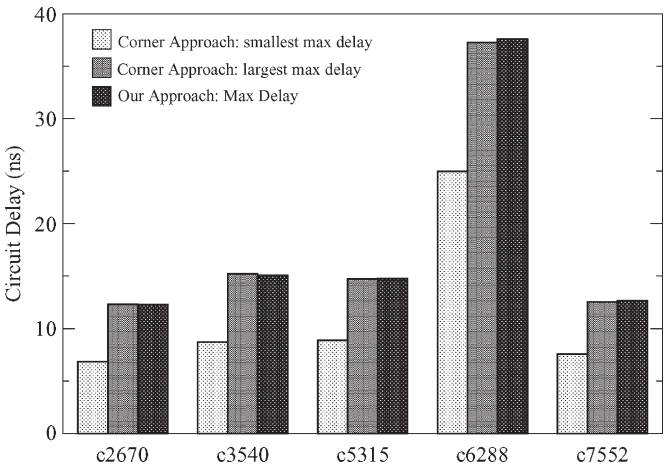

Fig. 9. Maximum delays for larger circuits.

smallest value found (over all corners) for the maximum delay of the circuit. The second bar shows the largest value found (over all corners) for the maximum delay of the circuit. The third bar shows the worst case delay reported by our approach. It is noteworthy that there is a significant difference between the first and second bars, indicating that process variations cause a significant delay spread for these circuits, and that our approach finds, in linear time, a tight upper bound on the worst case delay. We also compared the maximum slew rates of output signals as computed by our algorithm to the maximum slew rates found by running traditional STA at all process corners. Figs. 10 and 11 show the results for max slew. Note that the estimated delays and slews are, in some cases, smaller than those of the corner approach. The reason behind this is that the linearized interconnect delay, slew, and effective capacitance expressions are not conservative and might underestimate values at certain corners. This means that, although our "max" operation is conservative with respect to its inputs, the inputs themselves might not be conservative with respect to the true values. Finding a conservative linear approximation for interconnect delay, slew, and effective capacitance under process variability is beyond the scope of this paper, and for now, this remains a limitation of our method.

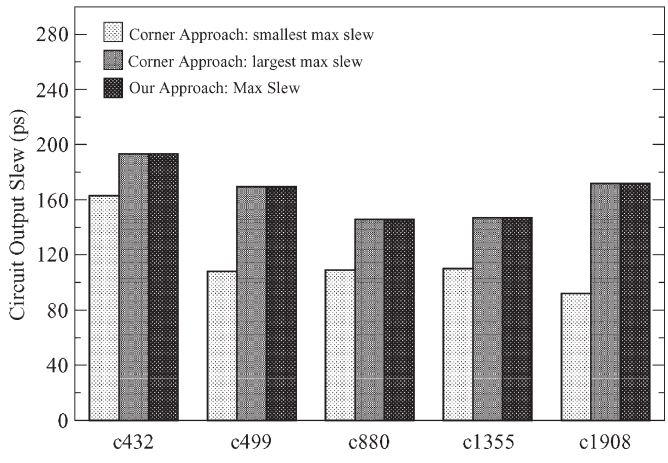

Fig. 10. Maximum slews for smaller circuits.

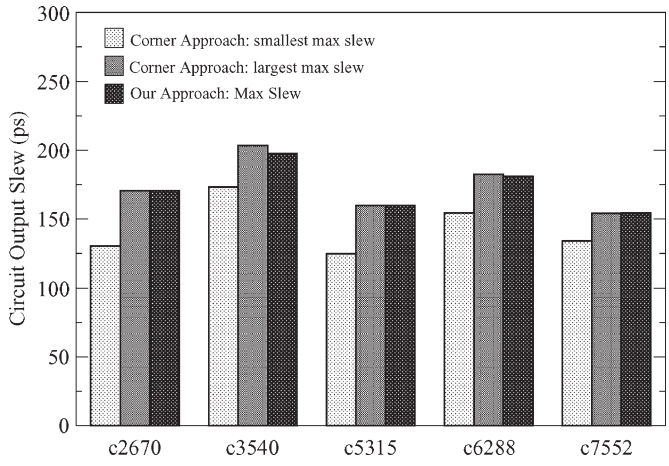

Fig. 11. Maximum slews for larger circuits.

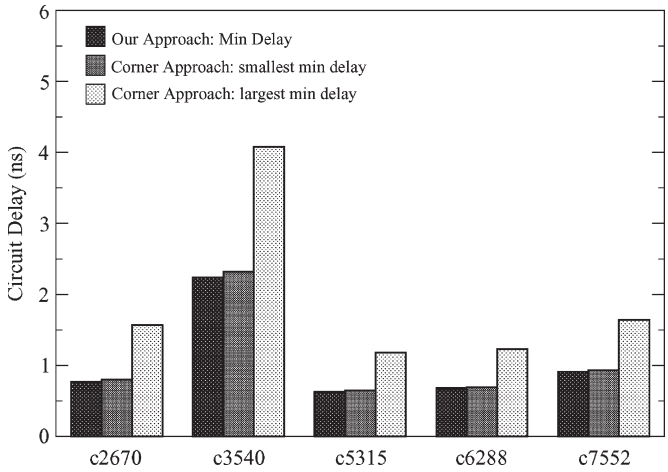

Fig. 12. Minimum delays for larger circuits.

Finally, as mentioned earlier, our approach is applicable to the min delay and min slew cases as well, for checking hold time violations. In this respect, Fig. 12 shows similar data for the min delay case.

We also recorded the run times of our approach for these circuits and compared them with the run times of the corner approach. As expected, the results in Table II show that our

TABLE II
RUN-TIME COMPARISON

ISCAS-85 Circuit	Corner Approach Run Time	Our Approach Run Time	Speed-up (×)
c432	13.48 s	2.39 s	5.64
c499	15.09 s	2.15 s	7.01
c880	25.71 s	4.26 s	6.03
c1355	38.59 s	6.07 s	6.36
c1908	65.75 s	9.27 s	7.09
c2670	85.84 s	12.12 s	7.08
c3540	123.69 s	17.62 s	7.01
c5315	188.42 s	26.88 s	7.01
c6288	176.83 s	28.22 s	6.27
c7552	264.07 s	39.69 s	6.65

approach achieves a speedup of approximately six to seven times when compared with the corner approach. While our approach is $\mathcal{O}(mn)$ (for a circuit with m gates and n relevant process parameters), the computational complexity of the corner approach is $\mathcal{O}(m2^n)$, hence the observed speedup. Moreover, as the number of process parameters being considered increases with future technology, this speedup will become even more dramatic.

VIII. CONCLUSION

All-corner timing analysis in linear time has been the "holy grail" in timing analysis for some time. In this paper, a linear-time approach has been presented, which does this with negligible error. The error is conservative in most cases. This technique uses standard/traditional timing models for cells and interconnect and, hence, is very easy to integrate into today's design methodology. It is hoped that this paper will lead to practical techniques for handling variability in VLSI design.

ACKNOWLEDGMENT

The authors would like to thank K. Heloue for his helpful feedback and contributions to this paper.

REFERENCES

[1] S. Abbaspour, R. Banerji, P. Feldman, and D. D. Ling, "Efficient variational interconnect modeling for statistical timing analysis by combined sensitivity analysis and model-order reduction," in *Proc. ACM/IEEE Int. Workshop TAU*, Feb. 2007, pp. 86–91.

[2] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, "Computation and refinement of statistical bounds on circuit delay," in *Proc. Des. Autom. Conf.*, Jun. 2–6, 2003, pp. 348–353.

[3] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," in *Proc. IEEE ISCAS*, Jun. 1985, pp. 663–698.

[4] F. Y. Liu, C. V. Kashyap, C. J. Alpert, and A. Devgan, "Closed-form delay and slew metrics made easy," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 12, pp. 1661–1669, Dec. 2004.

[5] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. IEEE/ACM ICCAD*, Nov. 9–13, 2003, pp. 621–625.

[6] A. Gattiker, S. Nassif, R. Dinakar, and C. Long, "Timing yield estimation from static timing analysis," in *Proc. Int. Symp. Quality Electron. Des.*, Mar. 2001, pp. 437–442.

[7] K. R. Heloue and F. N. Najm, "Statistical timing analysis with two-sided constraints," in *Proc. ICCAD*, Nov. 2005, pp. 829–863.

[8] J. Soreff, J. F. Lee, D. L. Ostapko, and C. K. Wong, "On the signal bounding problem in timing analysis," in *Proc. IEEE/ACM ICCAD*, 2001, pp. 507–514.

[9] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah, "Statistical timing for parametric yield prediction of digital integrated circuits," in *Proc. Des. Autom. Conf.*, Jun. 2–6, 2003, pp. 932–937.

[10] D. Sylvester, K. Agarwal, M. Agarwal, and D. Blaauw, "Statistical interconnect metrics for physical-design optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 7, pp. 1273–1288, Jul. 2006.

[11] J. Qian, S. Pullela, and L. Pillage, "Modeling the 'effective capacitance' for the RC interconnect of CMOS gates," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 12, pp. 1526–1535, Dec. 1994.

[12] S. Sapatnekar, *Timing*, 1st ed. Norwell, MA: Kluwer, 2004.

[13] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. Des. Autom. Conf.*, Jun. 2004, pp. 331–336.

[14] J. Vygen, "Slack in static timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1876–1885, Sep. 2006.

Sari Onaissi (S'08) received the B.E. degree in computer and communications engineering (with high distinction) from the American University of Beirut, Beirut, Lebanon, in 2005, and the M.A.Sc. degree in electrical and computer engineering (ECE) from the University of Toronto, Toronto, ON, Canada, in 2007, where he is currently working toward the Ph.D. degree.

His research is on computer-aided design for integrated circuits and is focused on timing under variability.

Farid N. Najm (S'85–M'89–SM'96–F'03) received the B.E. degree in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1983, and the Ph.D. degree in electrical and computer engineering (ECE) from the University of Illinois at Urbana–Champaign (UIUC), Urbana, in 1989.

From 1989 to 1992, he was with Texas Instruments, Dallas, TX. He was then with the ECE Department, UIUC, as an Assistant Professor and became an Associate Professor in 1997. Since 1999, he has been with the ECE Department, University of Toronto, Toronto, ON, Canada, where he is currently a Professor. His research is on computer-aided design (CAD) for very large scale integration (VLSI), with an emphasis on circuit-level issues related to power, timing, variability, and reliability.

Dr. Najm was an Associate Editor for the IEEE TRANSACTIONS ON VLSI SYSTEMS from 1997 to 2002 and is currently an Associate Editor for the IEEE TRANSACTIONS ON CAD OF INTEGRATED CIRCUITS AND SYSTEMS. He received the IEEE TRANSACTIONS ON CAD Best Paper Award in 1992, the NSF Research Initiation Award in 1993, and the NSF CAREER Award in 1996.