Quantifying Robustness Metrics in Parameterized Static Timing Analysis

Khaled R. Heloue
ECE Department
University of Toronto
Toronto, Ontario, Canada

Chandramouli V. Kashyap
Strategic CAD Labs
Intel Corporation
Hillsboro, OR 97123

Farid N. Najm
ECE Department
University of Toronto
Toronto, Ontario, Canada

***Abstract*—Process and environmental variations continue to present significant challenges to designers of high-performance integrated circuits. In the past few years, while much research has been aimed at handling parameter variations as part of timing analysis, few proposals have actually included ways to interpret the results of this parameterized static timing analysis (PSTA) step. In this paper, we propose a new post-variational analysis metric that can be used to quantify the (timing) *robustness* of designs to parameter variations. In addition to helping designers diagnose *if* and *when* different nodes can fail, this metric can guide optimization and can give insights on *what to fix*, by identifying nodes with small robustness values and proceeding to fix those nodes first. Inspired by the rich literature on *design centering, tolerancing, and tuning* (DCTT), we use *distance* as a measure for robustness. Our analysis thus determines the minimum distance from the nominal point in the parameter space to any timing violation, and works under the assumption that parameters are specified as ranges rather than statistical distributions. We demonstrate the usefulness of this distance-based robustness metric on circuit blocks extracted from a commercial 45nm microprocessor.**

I. Introduction

With the continuous scaling of integrated circuits, the control over process and environmental parameters has become increasingly difficult. As a result, PVT (process/voltage/temperature) parameters are found to exhibit large deviations from their nominal values, which causes circuit delay variations and possibly timing failures. Therefore, one needs to account for variability as part of the timing verification step. For ASICs, corner case analysis is traditionally used, whereby timing is verified at all process corners corresponding to extreme settings of PVT parameters. For microprocessors, the resulting chips are typically "binned" at different frequencies to account for variations. In the recent past, this problem has become worse because, not only has the number of parameters subject to variations increased, leading to a larger number of corners, but also the within-die (local) variations have become more significant, and they cannot be handled using traditional corner analysis.

With the traditional approaches to timing verification becoming too expensive and unable to handle local variations, new alternatives have emerged in recent years. These techniques have focused on assessing the effects of parameter variations on timing as part of the timing analysis step. All these techniques consider circuit delay to be dependent on a number of PVT parameters, and therefore can be collectively described under the heading of *parameterized static timing analysis* (PSTA). Statistical static timing analysis (SSTA) is one example of PSTA, in which parameters are modeled as random variables with known distributions and correlations [1], [2], and timing yield is estimated from the corresponding distribution of circuit delay. In practice, however, the statistical distributions and correlations of some PVT parameters may be unknown or unavailable. Also, some parameters, such as supply voltage or temperature, are not truly random and are better modeled as simply unknown or uncertain variables. Thus, some alternative PSTA techniques have also been proposed, such as multi-corner static timing analysis (MCSTA). MCSTA models parameters as uncertain variables within given or known bounds, and attempts to verify circuit timing at all corners in a single timing run [3], [4]. Although the circuit delay is captured accurately at the worst case corner, the same cannot be said about other points in the parameter space. Recently, some PSTA techniques [5], [6] have addressed this limitation by proposing to capture, in a single timing run, circuit delay *exactly* at all points in the PVT space. This can be done by propagating in the timing graph all the paths that can become dominant (critical) at any setting of the PVT parameters, and pruning all other redundant paths. In any case, the end result of all PSTA techniques is to provide designers with parameterized timing quantities (arrival times and/or slacks) which are expressed as functions of PVT parameters.

A. Scope of This Work

While a large body of research has focused on the *analysis* step, very few proposals have presented clear answers for how to interpret and utilize the results of PSTA in design. The crux of the matter, and this is what motivated PSTA in the first place, is that the goal is to produce a *safe* design, in the sense that it must be *robust* to variations. In order to do that, an essential requirement is to be able to *measure* the safety or robustness of a given design, i.e., its *susceptibility to timing failure* due to variations. But how does one formally define robustness? How does one quantify the susceptibility to failure and determine how far a nominally safe design is from the "edge of the cliff"? Finally, what does one need to do in order to improve the robustness of a given design? One way to quantify robustness, which is used in SSTA, is to use the notion of *yield* to assess the safety of timing quantities. Indeed, timing yield is one measure of robustness - it provides designers with the probability of meeting/violating the timing constraints. However, yield analysis via SSTA requires that all parameters

be modeled as random variables with known distributions and correlations. As we noted earlier, the distributions and correlations of PVT parameters may not always be available or fully specified. Hence, for those PSTA techniques where parameters are either *i)* modeled as uncertain (non-random) variables in specified ranges, or *ii)* where distributions are unknown or unavailable, we need to define some other metric which can be used to assess the robustness of timing quantities resulting from these methods. One possible way of doing that is to use the *volume* of the feasible region as a measure of robustness. As part of their work on yield prediction, the authors of [7] have proposed two techniques for approximating the volume of the feasible region, the *parallelepiped method* and the *ellipsoid method*. However, it was found that, while both techniques are accurate, they may not scale well with the number of PVT parameters or the number of paths. Thus, this approach can be costly on large designs.

B. Overview

In this paper, we hope to answer some of the above open questions as we present a new metric that can be used to *quantify* the (timing) robustness of a design to variations in the case where parameters are given as ranges rather than fully specified distributions. Our method *processes* the parameterized timing quantities resulting from PSTA so as to extract useful information about the susceptibility to timing failures. We will define robustness as the minimum *distance*, from the nominal point in the PVT parameter space, to any other point where a timing violation occurs. Such distance-based metrics have been used in a different context in the realm of design centering, tolerancing, and tuning (DCTT) [8], [9]. In DCTT, optimal nominal values for some *designable parameters* are selected so that the *distance* from the nominal point (center point of the design) to the boundary of the *acceptability region* is maximized in the hope of maximizing the process yield. Traditional DCTT operates in the space of design parameters, whereas our distance metric is measured in the PVT parameter space. It is also important to note that design centering has only been traditionally applied to small, typically analog, circuits, not to large digital integrated circuits; this is because it relies on expensive statistical simulations to determine the acceptability region of the design parameters, not to mention that the number of those parameters increases with circuit size. We will see that, in our use of a distance metric in PVT space, these complications do not arise and the resulting approach is computationally efficient.

Thus, the novelty of this paper is in its proposed application, where we extend the use of such distance metrics to the timing verification of large logic circuits, which, to our knowledge, was never done before. Using this new distance-based robustness metric, designers can not only diagnose if and when different nodes can fail timing, but also get insight on *what to fix*. In fact, our robustness metric can be used to evaluate design quality and to guide optimization by ranking different nodes according to their robustness, thus identifying the least robust nodes and proceeding to fix those nodes first. We also show that our robustness analysis can handle parameterized timing quantities resulting from either exact [5], [6] or bounded/approximate [3], [4] PSTA techniques. Also, our metric is computed efficiently and scales well (linear complexity) with the number of PVT parameters and the number of paths.

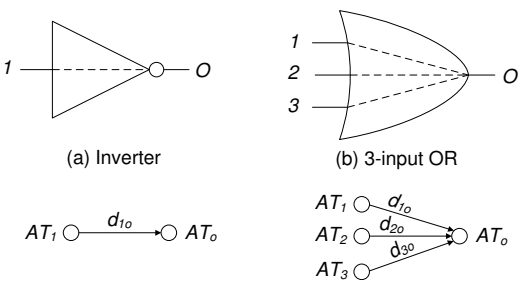


Fig. 1. Timing graphs for (a) Inverter, and (b) 3-input OR gate

The rest of the paper proceeds as follows. Section II covers some basic terminology and describes how parameterized timing quantities are represented in PSTA. In Section III, we cover robustness analysis in detail, first by comparing the notions of robustness and sensitivity, and then by defining robustness using normed distances in higher dimensions. We show some results in Section IV and conclude in Section V.

II. Preliminaries

We will review the terminology used in static timing analysis (STA) and describe how STA is extended to handle PVT variations as part of parameterized static timing analysis (PSTA).

A. Nominal Static Timing Analysis

In static timing analysis, the circuit under study is represented as a timing graph by creating a graph node for every electrical net in the circuit (primary input, output, or internal node) and a graph edge for every *timing arc* (logic gate input/output pair). The weight of every edge corresponds to the delay value from that input pin to the output pin. The *arrival time* at the output of a gate is computed first by *adding* the input arrival times to their corresponding timing arc delays (edge weights), and then taking the *max* over the result of those additions. This procedure is repeated while topologically traversing the timing graph and computing the arrival times at every node.

Fig. 1 shows the timing graphs for two simple logic gates, an inverter and a 3-input OR gate. The edge weight, d_{io}, corresponds to the arc delay from input i to the output. Since the inverter has one input, the arrival time (AT) at its output is simply:

$$AT_o = AT_1 + d_{1o} \tag{1}$$

For the OR gate, its output arrival time is the maximum of the sum of its three input arrival times and their corresponding arc delays:

$$AT_o = \max_{i=1}^{3}(AT_i + d_{io}) \tag{2}$$

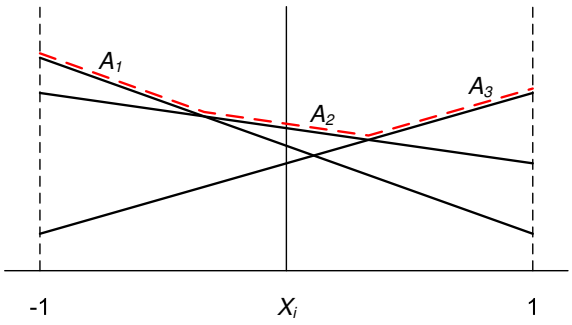


Fig. 2. Parameterized Arrival Time

While arrival times are computed during forward propagation in the timing graph, *required times* (RT), which are defined as the latest acceptable arrival times that would not violate the timing constraints, must be computed based on the information downstream, and thus require a backward propagation from the primary outputs. For a node to pass timing, its arrival time must not exceed its required time. The concept of *slack*, which is the difference between the required time and the arrival time at a node, is generally used as a measure of how close a node is to violating its timing constraint. In general, we require the slack S to be positive:

$$S = RT - AT \geq 0 \tag{3}$$

In nominal static timing analysis, all the above timing quantities (AT, RT, S) are computed under the assumption that process and environmental parameters - and consequently timing arc delays which depend on these parameters - are fixed, typically at either their *nominal* or corner values. However, due to the increasing significance of variability, parameterized static timing analysis (PSTA) techniques have emerged, with the goal of handling parameter variations as part of the timing analysis step.

B. Parameterized Static Timing Analysis

A key component of any PSTA technique is the *delay model* that captures the dependence of gate/interconnect delays on the underlying process and environmental parameters. First-order linear delay models are often used in the literature, and they generally capture well this dependence. Under such a linear model, the delay D of a timing arc is expressed as:

$$D = d_o + \sum_{i=1}^{p} d_i X_i \tag{4}$$

where d_o is the nominal delay value and d_i is the (first-order measure of) sensitivity to parameter X_i. Note that X_i can represent the variation of any parameter, such as channel length, supply voltage, or temperature. Also note that d_o and the d_i's are determined during library characterization.

Another component of PSTA is the model for the PVT parameters. As noted above, while SSTA techniques [1], [2] use random variables with known probability distributions (e.g. Gaussian) to model the parameters, other PSTA techniques [3]–[6] model them as *uncertain* variables that are specified in given ranges. We will adopt this more general model, based on uncertain parameters, because the distributions and correlations of some PVT parameters may be unknown or unavailable in practice. For simplicity, and without loss of generality, we will assume that the variation range of every uncertain parameter X_i is normalized to $[-1, 1]$. Following standard terminology, the linear model in (4) will be referred to as a delay *hyperplane*.

Because path delay is the sum of gate and interconnect delay hyperplanes on that path, it is also modeled as a hyperplane. However, arrival times are not simply hyperplanes because, when different paths converge at a node, a (nonlinear) *max* operation must be performed to determine the arrival time at that node. For example, consider Fig. 2, where A_1, A_2, and A_3 represent path delay hyperplanes. For purpose of illustration, a single parameter X_i is considered, so that the hyperplanes are simply straight line segments. The dashed piecewise linear function (in general piecewise planar) resulting from the max operation corresponds to the exact representation of the arrival time as:

$$AT = \max(A_1, A_2, A_3) \tag{5}$$

where the A_j's have the form in (4).

A similar piecewise planar function representing the *min* operation arises when one is dealing with parameterized slacks. For example, consider the circuit in Fig. 3 having two primary outputs at registers R_1 and R_2. The arrival time AT_1 at the data input of register R_1 is represented by a piecewise planar surface, say $AT_1 = \max(A_1, A_2, A_3)$. Note however, that the required time RT_1 at the data input of R_1 is not a max surface but a (single) hyperplane. This is because we are assuming that the arrival time at the clock input, r_1, is the delay hyperplane corresponding to a clock tree path. Hence, the parameterized slack S_1, at the input of register R_1 will be given as a minimum of a set of hyperplanes, as follows:

$$\begin{aligned} S_1 &= RT_1 - AT_1 \\ &= RT_1 - \max(A_1, A_2, A_3) \\ &= RT_1 + \min(-A_1, -A_2, -A_3) \\ &= \min\big((RT_1 - A_1), (RT_1 - A_2), (RT_1 - A_3)\big) \end{aligned} \tag{6}$$

where we have used the fact that $\max(a, b) = -\min(-a, -b)$. A similar reasoning follows for the parameterized slack S_2 at the input of register R_2. As a result, the minimum slack S for this circuit, being the minimum of all parameterized slacks at registers inputs will also be represented as a piecewise planar surface defined by the min operation above.

While some PSTA techniques [5], [6] capture exactly the piecewise planar surfaces representing the nonlinear max and min operations, other techniques [3], [4] approximate and/or bound those operations using a single hyperplane. In general, both exact and approximate PSTA techniques result in timing quantities (arrival times/slacks) being parameterized as func-

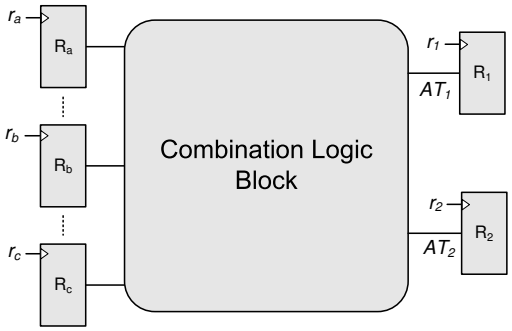


Fig. 3. Slack Computation

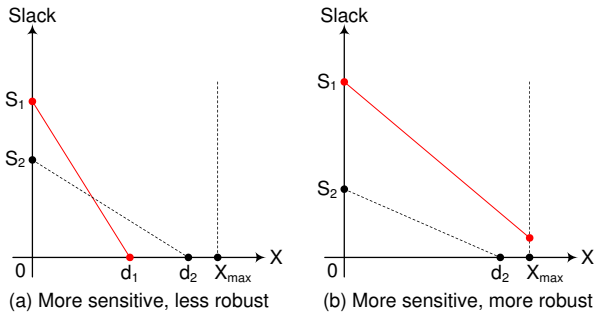


Fig. 4. Sensitivity and Robustness

tions of the PVT parameters, as follows:

$$AT(X) = \begin{cases} a_o + \sum_{i=1}^{p} a_i X_i, & \text{approx. PSTA} \\ \max_{j=1}^{n}(a_{oj} + \sum_{i=1}^{p} a_{ij} X_i), & \text{exact PSTA} \end{cases} \quad (7)$$

$$S(X) = \begin{cases} s_o + \sum_{i=1}^{p} s_i X_i, & \text{approx. PSTA} \\ \min_{j=1}^{n}(s_{oj} + \sum_{i=1}^{p} s_{ij} X_i), & \text{exact PSTA} \end{cases} \quad (8)$$

where n is the number of hyperplanes that define the piecewise planar surfaces of the max and min operations, and $-1 \leq X_i \leq 1$ for all i.

III. Robustness Analysis

Although parameterized expressions of timing quantities resulting from PSTA are very useful, they do not *directly* provide a metric of robustness of these timing quantities to variations. Some further *processing* of these expressions is required, to extract this information. We are interested in transforming complex expressions of PVT parameters, such as (7) and (8), into a measurable or quantifiable *robustness metric*. In this section, we first define such a metric as a measure of how close a node is to violating its timing constraint. We also compare robustness and sensitivity and highlight the subtle difference between the two notions. Finally, we present our mathematical formulation for robustness analysis and describe our algorithm.

A. From Sensitivity to Robustness

Suppose that one is comparing two design realizations of the same circuit, for which PSTA has provided the two different parameterized slacks at some node, $S_1(X)$ and $S_2(X)$. Alternatively, suppose that $S_1(X)$ and $S_2(X)$ are the parameterized slacks at two nodes of the same design. Either way, we are interested in comparing the robustness of $S_1(X)$ and $S_2(X)$. If, at the nominal point $X = 0$, it turns out that $S_1(0) \geq S_2(0) \geq 0$, then one might be inclined to assume that S_1 is more robust than S_2 because it is larger, and thus variations would seem to affect it less adversely. However, this is not always true, as it may turn out that S_1 is more *sensitive* to variations than S_2, and consequently more prone to failure. Hence, sensitivity is an important measure that is closely tied to robustness.

As an example, Fig. 4 shows a comparison of two parameterized slacks, $S_1(X)$ and $S_2(X)$, where we have assumed a single parameter X, varying in $[0, X_{\max}]$. Fig. 4a shows a case where the slack with the larger nominal value turns out to be less robust. In fact, although $S_1(0) \geq S_2(0)$, $S_1(X)$ fails "before" $S_2(X)$, because the value of X for which $S_1(X)$ becomes zero, d_1, is smaller than d_2, the value of X for which $S_2(X)$ becomes zero. In this case, the more sensitive slack turns out to be the one that is less robust. On the other hand, Fig. 4b shows a case where the opposite happens: even though $S_1(X)$ is more sensitive than $S_2(X)$, it is actually more robust. This is because $S_1(X)$ does not fail in $[0, X_{\max}]$, while $S_2(X)$ fails for $X = d_2$. Therefore, while robustness is related to the *susceptibility of a node to violating timing*, sensitivity is related to the magnitude of timing deviation, per unit parameter variation, irrespective of whether or not timing is actually violated. Thus, a node having the larger sensitivity yet not failing anywhere in the parameter space is "more robust" than a node having smaller sensitivity, yet failing somewhere in the parameter space.

In order to fully capture the notion of design robustness, we need to somehow make use of both the nominal values *and* the sensitivities, in relation to the threshold where timing failure occurs.

B. Quantifying Robustness

For the simplified scenario shown in Fig 4, one can define robustness as simply the value of X for which timing is violated. This would be a good metric to use because it is quantitative; it would allow one to conclude, for example, that S_2 is more robust than S_1 whenever $d_2 > d_1$. However, in the general case where several parameters are varying, and where parameterized timing quantities are piecewise planar surfaces, each with a different set of sensitivities, things can get more complicated. For one thing, finding a setting for the parameter vector X where timing is violated is not as simple as finding the x-intercept in Fig. 4, and requires a search in a higher-dimensional space. Furthermore, there could be be many such settings, making it hard to judge which one to use as a measure of robustness.

1) Distance-based Metric: We propose that a distance-based metric is a good choice for robustness analysis. Specifi-

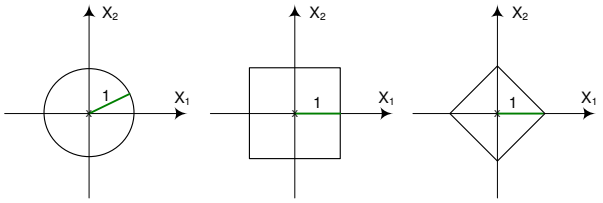


Fig. 5. Unit Ball in different norms

cally, we define the robustness metric as the *minimum distance from the nominal point* in the PVT parameter space to any point where a timing violation occurs. Distance can easily abstract the large dimensionality of the problem by presenting a simple quantifiable measure that can be computed with little effort. By measuring distance from the nominal point, we are implicitly assuming that the nominal design is feasible, i.e., it meets the timing constraints. Thus, the minimum distance to any timing violation reflects the smallest (magnitude) deviation from the nominal point that would "break" the design.

In fact, such distance-based metrics have been used in the past as part of design centering for yield maximization [8]–[10]. The goal of design centering is to determine the optimal nominal settings of the *design parameters*, which are constrained to satisfy performance specifications in the presence of tolerances. These nominal values (which define the center point of the design) are optimized such that the *distance from the design center to the boundary* of the acceptability region is maximized, in the hope of maximizing yield. Distance is therefore used in design centering as a measure of safety, since the more "distant" the center is from the boundary, the higher is the expected yield. In our case, where we work with PVT parameters, rather than design parameters, we do not try to recenter our design so that distance is maximized. Instead, we only use distance as a measure of how robust a timing quantity (or a design) is in the face of PVT variations. In fact, in our problem, the nominal PVT point is fixed, as well as the ranges of variation, whereas in design centering, the space under study is that of the design parameters, whose nominal values, as well as tolerances (ranges) are to be determined.

C. Mathematical Formulation

In this section, we present the mathematical formulation of our robustness analysis by means of *normed distances* to the boundary of the region where timing is met.

1) The Feasible Space: In general, given a parameterized timing quantity $T(X)$, we define its robustness, r, as the minimum distance (using some vector *norm*) from the nominal PVT point, $X = 0$, to any point in the PVT space where $T(X)$ violates timing. Recall that we have assumed in Section II-B that PSTA has already been used to analyze the design, and that parameterized slacks $S(X)$ and/or arrival times $AT(X)$ are available. Each timing quantity is either a hyperplane or a collection of n hyperplanes defining a piecewise planar surface, as shown in (7) and (8). In the analysis that follows, we assume that one is dealing with parameterized slacks $S(X)$, however, the same analysis can be easily applied to the case of parameterized arrival times. Recall that $S(X)$ is defined in (8) as the minimum of n hyperplanes:

$$S(X) = \min\left(S_1(X), \ldots, S_n(X)\right) \tag{9}$$

where $n \geq 1$ (for approximate PSTA, $n = 1$), and where:

$$S_j(X) = s_{oj} + \sum_{i=1}^{p} s_{ij} X_i, \quad j = 1, \ldots, n \tag{10}$$

We also assume that this set of n hyperplanes has already been reduced using pruning techniques, such as in [5], [6], so that every hyperplane $S_j(X)$ can become the minimum, *i.e.*, $S(X) = S_j(X)$, for some X.

With the above expressions for slack, the space where timing is satisfied, $\mathcal{C}$, is defined by $S(X) \geq 0$, which can be expressed as a convex polytope (intersection of linear constraints) by replacing $S(X)$ by its expression in (9), as the minimum of n hyperplanes:

$$\mathcal{C} = \left\{ X \;\middle|\; S_j(X) = s_{oj} + \sum_{i=1}^{p} s_{ij} X_i \geq 0, \quad j = 1, \ldots, n \right\} \tag{11}$$

Also, we assume that the ranges of X_i's are normalized to $[-1, 1]$, so that the parameter space, $\mathcal{D}$, is defined as the following p-cube:

$$\mathcal{D} = \left\{ X \;\middle|\; -1 \leq X_i \leq 1, \quad i = 1, \ldots, p \right\} \tag{12}$$

Therefore, the intersection of $\mathcal{C}$ with $\mathcal{D}$ corresponds to all X in $\mathcal{D}$ for which timing is met, *i.e.*, $S(X) \geq 0$. We refer to this $\mathcal{C} \cap \mathcal{D}$ as the *feasible region*. As mentioned earlier, we assume that the nominal design is feasible (meets timing), so that the nominal point $X = 0$ is inside the feasible region. If one starts at the nominal point and moves outward, then two cases may arise. Either the boundary of $\mathcal{C}$ is encountered and crossed first at which point timing is violated (*i.e.*, $S_j(X) < 0$ for one or more j), or the boundary of $\mathcal{D}$ is encountered and crossed first, at which point the range is exceeded (*i.e.*, $|X_j| > 1$ for one or more j). For robustness analysis, we are interested in the minimum distance from the nominal point to any point within the range $\mathcal{D}$ at which timing is violated. Therefore, we are interested in the minimum distance to the boundary of $\mathcal{C}$, obviously provided that $S(X)$ fails somewhere inside $\mathcal{D}$.

2) Normed distance to a hyperplane: We want the minimum distance to the boundary of the convex polytope $\mathcal{C}$ defined by the set of n linear constraints in (11), $S_j(X) \geq 0, \forall j$. The boundary corresponds to n hyperplanes, h_j, $j = 1, \ldots, n$, where:

$$h_j \;:\; S_j(X) = 0 \tag{13}$$

$$:\; \sum_{i=1}^{p} s_{ij} X_i = -s_{oj} \tag{14}$$

$$:\; a_j^T X = b_j \tag{15}$$

where a_j is the vector $a_j^T \triangleq [s_{1j}\ s_{2j}\ \cdots\ s_{pj}]$ and $b_j \triangleq -s_{oj}$.

Let $d_n(X_o, h_j)$ be the distance, in an arbitrary vector norm $\|X\|$, from a point X_o to the hyperplane h_j. This so-called *normed distance* can be expressed in terms of the norm's unit ball, $\mathcal{B}_n = \{X \mid \|X\| \leq 1\}$, as follows [11]:

$$d_n(X_o, h_j) = \min\{|\lambda| \mid (X_o + \lambda\mathcal{B}_n) \cap h_j \neq \emptyset\} \tag{16}$$

Therefore, in order to determine the normed distance from X_o to h_j, one needs to dilate the unit ball by λ around X_o until it *touches* the hyperplane. Fig. 5 shows different unit balls $\mathcal{B}_2$, $\mathcal{B}_\infty$, and $\mathcal{B}_1$ (in 2-D) for the L_2-, L_∞-, and L_1-norms, respectively, where:

$$L_2\text{-norm} \quad = \quad \|X\|_2 = \sqrt{\sum_{i=1}^{p} X_i^2} \tag{17}$$

$$L_\infty\text{-norm} \quad = \quad \|X\|_\infty = \max_{i=1}^{p} |X_i| \tag{18}$$

$$L_1\text{-norm} \quad = \quad \|X\|_1 = \sum_{i=1}^{p} |X_i| \tag{19}$$

The normed distance in (16) can also be expressed in terms of the dual norm $\|X\|^\star$, as follows [10], [11]:

$$d_n(X_o, h_j) = \frac{|a_j^T X_o - b_j|}{\|a_j\|^\star} \tag{20}$$

where the dual norm is $\|u\|^\star = \sup\{u^T v \mid \|v\| \leq 1\}$. For the L_p-norm $\|X\|_p$, defined by:

$$\|X\|_p = \left(\sum_i |X_i|^p\right)^{\frac{1}{p}}, \tag{21}$$

the dual norm is the L_q-norm $\|X\|_q$, such that:

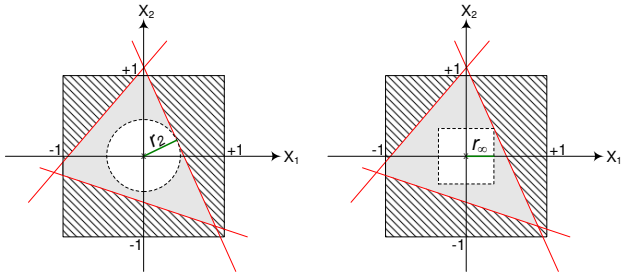
$$\frac{1}{p} + \frac{1}{q} = 1 \tag{22}$$

Note that the L_2-norm is self dual, and that the L_1 and L_∞ norms are duals of one another.

Therefore, using the very simple normed distance expression in (20), we can efficiently determine the distance (in any L_p-norm) from the nominal PVT point $X = 0$ to every hyperplane h_j defining the boundary of $\mathcal{C}$, and record the smallest such distance as the robustness metric, r, of $S(X)$.

3) Algorithm and Illustration: A description of the algorithm, Find_Robustness, is shown in Algorithm 1. It takes as input a parameterized slack, $S(X)$, and returns its robustness, r, as defined above. The algorithm starts by checking two corner cases. First, the nominal slack is checked, at the nominal point $X = 0$ (line 1). If $S(0) \leq 0$, then the nominal slack violates timing. In that case, $X = 0$ is not feasible, and we simply set $r = 0$. Nodes with $r = 0$ are the least robust because they violate timing even before considering parameter variations. The second corner case to check is whether the minimum value (over X) of $S(X)$ is positive (line 3). Since $S(X)$ is the minimum of $S_j(X)$'s and $-1 \leq X_i \leq 1$, this can be easily checked as follows:

$$\min_X \{S(X)\} = \min_{j=1}^{n} \left\{ s_{oj} - \sum_{i=1}^{p} |s_{ij}| \right\} \tag{23}$$



(a) L_2-normed distance (b) L_∞-normed distance

Fig. 6. Robustness Analysis

If the above expression is positive, it simply means that $S(X)$ does not fail anywhere in the parameter space and, in that case, we set $r = \infty$. Nodes with $r = \infty$ are the most robust since they are not prone to timing violations anywhere in $\mathcal{D}$.

Algorithm 1 $r \leftarrow$ Find_Robustness$(S(X))$

Input: $S(X) = \min(S_1(X), \ldots, S_n(X))$
where $S_j(X) = s_{oj} + \sum_{i=1}^{p} s_{ij} X_i$
Output: $r \in \mathbb{R}$

1: **if** $(S(0) \leq 0)$ **then**
2: **return** $r = 0$
3: **else if** $(\min_X\{S(X)\} > 0)$ **then**
4: **return** $r = \infty$
5: **else**
6: $r = \infty$
7: **for** $(j = 1, \ldots, n)$ **do**
8: $S_j(X) = 0 \rightarrow h_j : s_j^T X = -s_{oj}$
9: $r_j = d_n(0, h_j) = \frac{|s_{oj}|}{\|s_j\|^\star}$
10: **if** $(r_j < r)$ **then**
11: $r = r_j$
12: **return** r

If both these corner conditions are not met, then $S(X)$ will fail somewhere in the parameter space $\mathcal{D}$. In that case (line 5), we first set $r = \infty$ (or some upper bound value). Then, we compute the normed distance, r_j, from the nominal point $X = 0$ to the (boundary) hyperplane h_j defined by $S_j(X) = 0$. To do that, we use the formula in (20) for some choice of L_p-norm and its corresponding dual L_q-norm. Typically, L_2-normed distances are mostly prevalent in the literature on design centering, with some use of L_∞-norm (and its corresponding L_1 dual norm). This is done for all boundary hyperplanes h_j's, and the smallest value of r_j is recorded as the robustness of $S(X)$ (lines 10-11).

Fig. 6 is a simple 2-D example depicting graphically how robustness analysis works, when both the L_2 and L_∞ norms are used. Note that the parameter space is defined by the square region where the two parameters are restricted to vary in $-1 \leq X_1, X_2 \leq 1$, and the feasible space where timing is met is defined by the grey triangle-like region that contains the nominal PVT point $X = 0$. The striped region outside the boundary of the feasible space is the region where timing is violated. Fig. 6a shows the robustness, r_2, computed in L_2-norm. This is equivalent to inflating an L_2-normed unit ball (disk) around $X = 0$ as shown, until it touches one of the

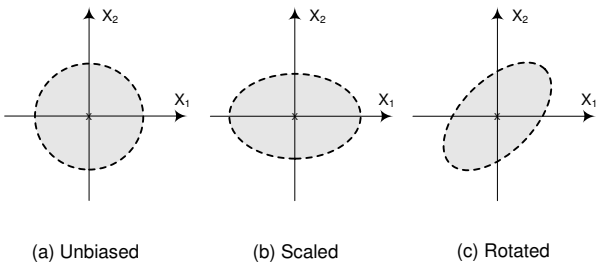


Fig. 7. Biasing the Norm

hyperplanes at the boundary. Similarly, a L_∞-normed unit ball (square) is inflated around $X = 0$ in Fig. 6b to obtain the robustness, r_∞, in L_∞-norm.

D. Unbiased vs Biased Analysis

The robustness analysis presented so far has assumed that the PVT parameters are uncertain variables given in ranges and having equal spreads (assumed to be [-1,1]). As a result, the analysis implicitly gives equal weights to all possible directions in the space. This option would be the most appropriate if absolutely no additional information is known about parameter variations and their interactions. We will refer to this type of analysis as *unbiased analysis*. On the other hand, if additional or partial information is available, whether it is from historical data or from process experience, then one could make use of this information so as to *bias* the robustness metric computation. For example, if parameters have different spreads, then one can use a scaling (diagonal) matrix to scale the norm itself and compute the distance-based metric in the new scaled norm. Also, if it is observed that certain parameters exhibit some *covariance*, that is, if certain parameters are likely to vary in the same (or opposite) direction, then one can rotate the norm by a nondiagonal scaling matrix. It is shown in [10] that, given any (diagonal or nondiagonal) scaling matrix W, the scaled norm is given by $\|X\|_W = \left\|W^{-1}X\right\|$, and its scaled dual norm is given by $\|X\|_W^\star = \left\|W^T X\right\|$. Both unbiased and biased analyses are depicted in Fig. 7, where (b) we have scaled the unbiased L_2-norm by reducing the range of X_2, and (c) have rotated the norm by $45°$, emphasizing the fact that X_1 and X_2 are likely to vary in the same direction.

IV. Results

In this section, we present the simulation results that were obtained on a 45nm commercial microprocessor design. Two parameterized static timing analysis flows were implemented in `C++` on top of an STA timing engine. The first is an exact PSTA flow that parameterizes timing quantities in the form of piecewise planar surfaces (collection of hyperplanes) defined by the max or min operations described in (7) and (8). The exact PSTA implementation is based on the pruning techniques of [5], [6]. The second flow is an implementation of the approximate PSTA technique of [4], which parameterizes every timing quantity as a single hyperplane. For both flows, we have considered global variations in four different parameter types, namely supply voltage (V_{dd}), Miller Coupling Factor

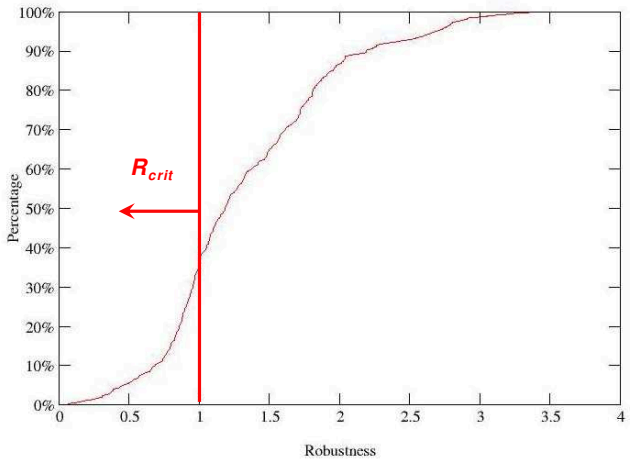


Fig. 8. Cumulative robustness distribution of failed slacks

(MCF), and channel length for both NMOS and PMOS devices (L_n and L_p). In addition, L_n and L_p are each divided into two types, based on whether the device is nominal or low power, and further into three types based on layout dependent information. Parameter variations are assumed to be independent, so a total 14 different PVT parameters were considered in the analysis (12 for L, MCF, and V_{dd}). In our robustness analysis, the L_2-norm was used to compute all normed distances from the nominal point to the boundary hyperplanes, as is typical in design centering.

We ran both exact and approximate PSTA on different microprocessor blocks and have determined parameterized arrival times at every node and parameterized slacks at the inputs of all registers in the blocks. Our robustness analysis was then applied on the parameterized slacks to quantify their robustness, as described in Section III. Recall that we have normalized the variation range of every parameter to $[-1, 1]$, and have considered 14 parameters. Therefore, if a slack fails somewhere in the parameter space, then its robustness r must fall between $r_{\min} = 0$ and $r_{\max}$ (based on L_2-norm), where:

$$r_{\max} = \sqrt{\sum_{i=1}^{14} (\pm 1)^2} \approx 3.74 \tag{24}$$

Fig. 8 shows a plot of the cumulative robustness distribution of all failed slacks for one microprocessor block, `ckt1`. It provides a ranking of all failed slacks according to where in the range $[0, 3.74]$ their robustness falls. Looking at the plot, it makes sense to start by fixing the slacks that have the smallest values of robustness, since those are the ones that are most prone to failure. In general, it is useful to have a robustness threshold, R_{crit}, such that all slacks with robustness less than R_{crit} would be considered critical and thus fixed. Note that R_{crit} does not have to be large since one is interested in detecting the slacks that are failing for a *small* deviation around the nominal point, at least for microprocessors.

Fig. 9 shows a plot of nominal slack vs robustness for different nodes. Slack values were normalized, and 16 nodes with very similar nominal slacks were picked (as shown). We have also assumed, for purpose of illustration, that if a slack

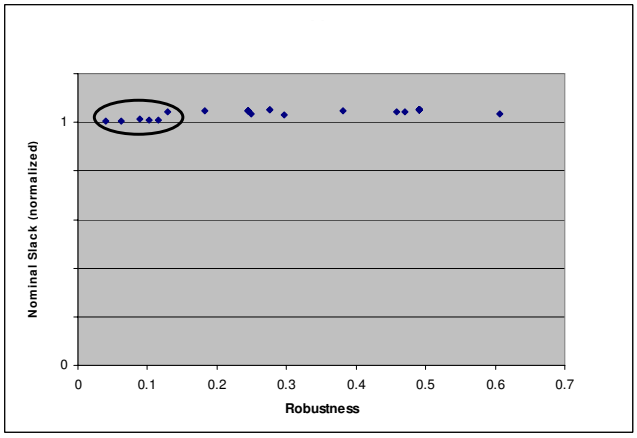


Fig. 9. Nominal Slack vs Robustness

goes below 90% of its nominal value (due to variations), then this would be considered a timing failure. In other words, we're simply using 90% of nominal slack instead of 0 as the threshold for failed slack. Based on this slack threshold, we have computed the robustness of the different slacks, and plotted nominal slack vs robustness. As shown on the plot, robustness values fall in $[0.05, 0.6]$, which is a large spread given that the nominal slacks are almost the same. In fact, if one had no access to robustness information, all the 16 slacks would *seem* to be equally robust looking only at their nominal slack. However, after factoring in our quantifiable robustness metric, one can easily determine which nodes are the most susceptible to variations. In a sense, the circled slacks are *closer to the edge of the cliff* than the ones with larger robustness.

We also checked if the results of robustness analysis are consistent when applied to *i)* exact PSTA vs *ii)* approximate PSTA. First, exact PSTA is invoked on another microprocessor block, `ckt2`, to obtain parameterized arrival times at every node and parameterized slacks at the inputs of all registers in the block. There were ≈ 1500 parameterized slacks, with n (number of hyperplanes defining every slack) ranging from 1 to 346 hyperplanes. Robustness analysis is then applied on the parameterized slacks with slack threshold set to 0, and (exact) robustness is recorded. Then, approximate PSTA is invoked, and parameterized slacks were obtained, each consisting of only a single hyperplane. Robustness analysis is then applied and (approximate) robustness is recorded. Out of the 1500 parameterized slacks, only 41 slacks failed somewhere in the parameter space, and thus have robustness values in $[0, 3.74]$. Fig. 10 shows a plot of approximate robustness (based on approximate PSTA) versus exact robustness (based on exact PSTA). In general, for the results of both robustness analyses to be consistent, the ranking of slacks in terms of their robustness should be preserved. In other words, the points should follow some straight line (not necessarily $y = x$, although $y = x$ would be an ideal case). This is what we see in this plot; the points are highly correlated (we have found the correlation coefficient to be $\rho = 0.93$) and they fall close to $y = x$.

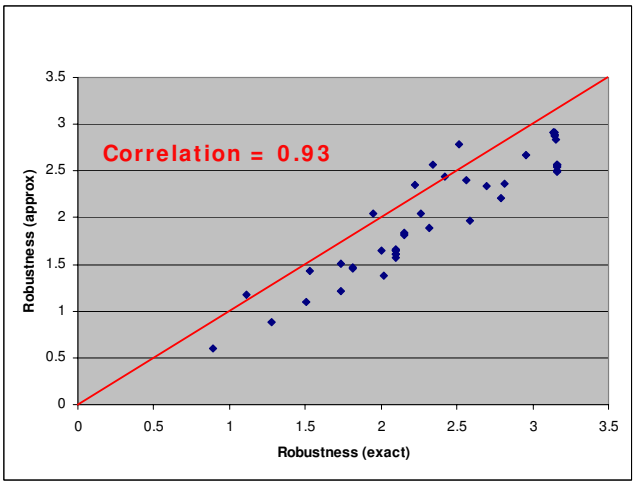


Fig. 10. Ranking nodes according to their robustness - Exact PSTA vs Approximate PSTA

V. Conclusion

In this paper, we presented a new robustness metric that can be used to quantify the vulnerability of designs to parameter variations. Our robustness metric provides a novel way to easily interpret the results of parameterized static timing analysis by *i)* determining if and when nodes can fail, *ii)* ranking those nodes according to their robustness, and *iii)* fixing the ones that are least robust. Using distance as the metric for robustness, we find the smallest normed distance from the nominal point in the parameter space to any timing violation, which can be computed efficiently using closed form expressions.

References

[1] H. Chang and S. S. Sapatnekar. Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-Like Traversal. In *IEEE/ACM International Conference on Computer-aided Design*, pages 621–625, 2003.

[2] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-Order Incremental Block-Based Statistical Timing Analysis. In *DAC*, pages 331–336, 2004.

[3] S. Onaissi and F. N. Najm. A linear-time approach for static timing analysis covering all process corners. In *ICCAD*, pages 217–224, 2006.

[4] K. R. Heloue and F. N. Najm. Parameterized timing analysis with general delay models and arbitrary variation sources. In *Design Automation Conference*, pages 403–408, 2008.

[5] S. V. Kumar, C. V. Kashyap, and S. S. Sapatnekar. A framework for block-based timing sensitivity analysis. In *Design Automation Conference*, pages 688–693, 2008.

[6] K. R. Heloue, S. Onaissi, and F. N. Najm. Efficient block-based parameterized timing analysis covering all potentially critical paths. In *IEEE/ACM International Conference on Computer-aided Design*, pages 173–180, 2008.

[7] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C Visweswariah. Satitstical Timing for Parameteric Yield Prediction of Digital Integrated Circuits. In *DAC*, pages 932–937, 2003.

[8] J. Bandler and H. Abdel-Malek. Optimal centering, tolerancing, and yield determination via updated approximations and cuts. *IEEE Transactions on Circuits and Systems*, 25(10):853–871, 1978.

[9] K. K Low and S. W Director. A new methodology for the design centering of IC fabrication processes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):895–903, 1991.

[10] R. Brayton, S. Director, and G. Hachtel. Yield maximization and worst-case design with arbitrary statistical distributions. *IEEE Transactions on Circuits and Systems*, 27(9):756–764, 1980.

[11] A. Schöbel. *Locating Lines and Hyperplanes: Theory and Algorithms*. Kluwer Academic Pub, 1999.