Abstract—Correct ordering of timing quantities is essential for both timing analysis and design optimization. The problem becomes, however, complicated when more effects of random process variations on design need to be considered, because timing quantities are no longer a deterministic value, but a distribution. How to properly order these correlated timing quantities is the key to leveraging various statistical techniques to solve design problems in the presence of process variation, such as reporting, optimization, and testing. In this paper, we propose a novel metric, called tiered criticalities, that guarantees to provide a unique order for a set of correlated timing quantities while properly taking into account full process space coverage. Efficient algorithms are developed to compute this metric in linear time, and its effectiveness on path ranking for at-speed testing is also demonstrated.

I. INTRODUCTION

A static timing analysis program serves two primary purposes. The first is to determine whether a design meets all timing requirements for sign-off, a goal that is (barely) met at the tail end of a design project. The second is to provide feedback and diagnostics to aid designers in optimizing timing and fixing timing fails.

In the context of deterministic timing, the most common feedback is provided by ordering timing quantities, typically slacks such as edge, node, path slacks, in order of criticality so that the designer knows which part of the design to focus on in order to make progress towards timing sign-off. Slack ordering is often required for a number of purposes, including circuit optimization, transistor sizing, timing-driven physical synthesis. For example, listing paths by ranking their path slacks is useful for both SPICE-verification of critical paths and at-speed delay testing.

With the advent of increased process variability in deep sub-micron technology, statistical timing is increasingly used for timing verification. Unfortunately, in the context of statistical timing, although the need for slack ranking in order of criticality persists, the situation is not as simple as in deterministic timing. Each node, edge and path of the timing graph has a probability of being critical among all manufactured chips, which is defined as criticality probability [1], [2], [3].

For a timing graph, the computation of criticality probabilities for nodes and edges was first proposed in [1], but that method does not correctly take into account correlations. Subsequently, various techniques [2], [3], [4] were developed to solve the correlation problem. For example, [2] derived an analytical formula that can be used to compute criticalities by traversing the timing graph backward once; while [3], [4] suggested a cutset-based method for edge and node criticality computation. Recently, [5] showed that criticality probabilities can be computed incrementally, and proposed, for the first time, a method of computing path criticalities. With the capability of ordering different correlated timing quantities (such as slacks) based on criticality probability, researchers have found that there are various CAD applications that can take advantage of this technique to combat the impact of process variation on design, for example, statistical circuit optimization [6], [3] that leverages criticality probabilities for nodes and edges, and path selection for at-speed testing [7], [8] that makes use of path criticalities.

Ranking slacks based on criticality probability is useful to identify the most critical slack while taking into account the entire process space. However, it suffers from the problem of probability masking. That is, when the top most critical slacks are determined, the ordering of the remaining slacks may not be properly determined. Dominance masking and correlation masking are the two major masking mechanisms.

The major contributions of this work are as follows. Based on our experience from industrial timing analysis, we share some insights on the difficulty of properly ordering statistical timing quantities resulting from the probability masking problem. A new metric, called tiered criticality, is proposed to address these masking issues. The new metric has more discriminative capability and it can guarantee to provide a unique ordering of a set of correlated statistical timing quantities, while taking into account full process space coverage. This feature is desirable and may find many potential applications in various CAD problems, such as timing report, path testing, and design optimizations. As an illustration, we apply tiered criticalities to select critical paths for at-speed testing. Novel algorithms are also developed to compute tiered criticalities efficiently, and the complexity is almost the same as existing non-tiered criticality computation. Experiment results show that the tiered criticalities can indeed provide a unique ordering of timing quantities, and its application on path selection also improves path quality for at-speed testing significantly.

The organization of the rest of this paper is as follows. Section II provides the background and reviews existing work on criticality probabilities. Section III formally defines the probability masking issues, and proposed a new metric called tiered criticality in the context of parameterized statistical timing. Algorithms for computing tiered criticality with various speed-up techniques will be presented. Section IV presents
experimental results on tiered criticality computation and its application in path selection for at-speed testing. We conclude this paper with discussion on our future work in Section V.

II. BACKGROUND

A. Statistical Timing Analysis

Statistical static timing analysis (SSTA) is a technique to mathematically capture the impact of random process variations on timing, in which all timing quantities in the timing graph, such as AT, RAT, and slacks, are represented as functions of the underlying process parameters

\[ S = F(\Delta X_1, \Delta X_2, \ldots, \Delta X_n), \]  

where \( \Delta X_i \) is a random variable that models the variation of a process parameter. Through different modeling techniques, \( \Delta X_i \) can capture chip-to-chip, across-chip (spatial), and local random variations. In the literature, the studied functional forms include linear [9], [1] quadratic [10], or general non-linear [11]; and \( \Delta X_i \) can be either Gaussian or non-Gaussian.

The atomic operations required for a block-based SSTA include addition, subtraction, min and max, and all operations are closed for the chosen functional form. In other words, given \( S_1 \) and \( S_2 \) in the form of (1), the result on \( S_1 \) and \( S_2 \) after any atomic operation would be in the same form as (1). All these operations have been developed for various SSTA techniques such as [9], [1], [10], [11].

B. Criticality Probabilities

In the context of timing and optimization, designers need to constantly compare different timing quantities such as slacks (e.g., node, edge, and path slacks) to make a judgment on the goodness of a design before and after a change. The comparison of deterministic slacks is straightforward, i.e., the smaller the value, the more critical the corresponding slack. But ranking different slacks in the form of (1) is not easy, as \( S \) now represents a distribution.

One way to transform a distribution to a value for comparison purposes is via projection. For example, the conventional corner projection is to project each \( \Delta X_i \) to one of its two boundary (typically ±3 sigma) values, and obtain one projected value \( \bar{\pi} \) as

\[ \bar{\pi} = F(\delta x_1|\pm 3, \delta x_2|\pm 3, \ldots, \delta x_n|\pm 3). \]  

And the worst corner projection is the combination of +3 sigma or -3 sigma of each \( \Delta X_i \) that jointly pushes the projected slack \( \bar{\pi} \) to the worst (i.e., smallest).

But using the projected value \( \bar{\pi} \) as a surrogate for the comparison of \( S \) may not provide effective guidance for design optimization. For example, Fig. 1 shows the distributions of two slacks \( B \) and \( C \) as a function of one random variable \( \Delta X \). If we were to compare \( B \) and \( C \) based on worst corner projection, then \( C \) would appear to be more critical than \( B \). But if we consider all possible (weighted) occurrences of process parameter \( \Delta X \), it is clear that slack \( B \) has a higher probability of being smaller than \( C \). To effectively guide design optimization with limited resources, we would prefer first optimizing \( B \) to \( C \). In other words, to correctly rank slack \( S \), we need to consider the multi-dimensional space of process parameters (e.g., the weighted occurrences of \( \Delta X \) in one dimension as in Fig. 1).

\[ p_i = P(S_i \leq \min_{j \neq i}(S_j)). \]  

If we use criticality as a metric to rank \( B \) and \( C \) as in Fig. 1, we would determine that \( B \) is more critical than \( C \), as \( B \) has higher probability of being smaller than \( C \).

III. STATISTICAL ORDERING BY TIERED CRITICALITY

A. Probability Masking

Although the criticality probability as defined in (3) is good at capturing the most critical slack, it suffers from two types of probability masking problems: dominance masking and correlation masking.

![PDF](image1)

Fig. 1. Criticality probability versus slack projection.

This was captured by a concept called criticality probability as first defined in [1]. The criticality of \( S_i \) is the probability of \( S_i \) being the smallest among all other \( S_j \), i.e.,

\[ p_i = P(S_i \leq \min_{j \neq i}(S_j)). \]  

If we use criticality as a metric to rank \( B \) and \( C \) as in Fig. 1, we would determine that \( B \) is more critical than \( C \), as \( B \) has higher probability of being smaller than \( C \).

![PDF](image2)

Fig. 2. Illustration of dominance masking.

Dominance masking happens to dominated slacks with zero criticality and criticality gives us no guidance to order them. For example, Fig. 2 shows the distributions of three slacks \( A, B, \) and \( C \) as a function of one random variable \( \Delta X \). Because \( A \) is clearly dominating (smaller than) both \( B \) and \( C \) in the entire space of \( \Delta X \), the criticality probabilities of \( A, B, \) and \( C \) according to (3) are one, zero, and zero, respectively. Though we are able to rank \( A \) as the most critical slack, we cannot properly order \( B \) and \( C \) based purely on their criticality probabilities. But for most applications, we would like to rank \( B \) as more critical than \( C \), as \( B \) has a higher probability of being smaller than \( C \).

Correlation masking refers to situations of highly correlated slacks in which criticality probabilities cannot properly order
them. If one of them is smaller than the rest by a small amount, this slack would get all the criticality probability while the rest would get zero criticality probabilities. For example, as shown in Fig. 3, assuming that $A$ and $B$ are almost perfectly correlated, and $A$ is only smaller than $B$ by a small amount, the criticality probability of $B$ would be zero, totally dominated by $A$ due to high correlation between them. As a result, $B$ would be ranked as less critical than $C$. But from most applications’ perspectives, it is clear that $B$ should be considered as more critical than $C$. Also, the small difference between $A$ and $B$ may be caused by a small modeling or numerical error, then the decision of $A$ being more critical than $B$ is not reliable.

For example, one application that suffers from the probability masking problem is path selection for at-speed testing [7], [8]. In the presence of process variation, different paths will become frequency-limiting for chips manufactured under different process conditions. Therefore, it is important to select a set of critical paths to undergo at-speed testing such that every chip, no matter under what process conditions it is manufactured, would have its most critical paths tested. By doing so we can screen out all bad chips and only ship good chips to customers, thus achieving the best yield without hurting the level of shipped-product quality loss.

As the number of paths in a design is exponential, and we only have a limited path budget for at-speed testing, ranking paths is important. The authors of [7], [8] proposed a two-step procedure to address this problem. (1) They first rank all end points in terms of node criticality as computed by [4], and then select all end points with non-zero criticalities. (2) For each chosen critical end point, they select a number of critical paths leading to that end point based on either deterministic slacks, projected slacks [7], or a test coverage metric (TQM) [8].

However, because of the probability masking issues, the selected end points with non-zero criticalities in the first step may be limited; while dominance and correlation masking issues may prevent some “useful” end points with zero criticalities from being included. An end point whose slack exhibits distribution same as $B$ in Fig. 2 and 3 is valuable to have for at-speed testing. The reasons are multi-fold. First, if the first path $A$ has a negative slack and must be “fixed,” then as soon as its slack is improved, the second path $B$ will require attention. Second, due to a small modeling inaccuracy, the second path $B$ may be the one in reality with the high criticality probability. Third, if the first path $A$ is selected for testing purposes and turns out to be either unsensitizable or hard-to-sensitize during the Automatic Test Pattern Generation (ATPG) step, the second path $B$ is extremely valuable for at-speed test to detect process variations.

Therefore, a more sophisticated and discriminating metric is required.

### B. Tiered Criticality Definition

To overcome probability masking issues, we propose a new metric called tiered criticality probability in this paper.

Before delving into a mathematical definition, we present the intuition behind tiered criticality. Given a set of slacks, we first find the one with the highest probability of being the smallest. In order to avoid masking problems, we then remove this slack from the slack set, and repeat the procedure of finding the slack with the highest probability of being the smallest among the remaining slacks. Carrying out this procedure on the examples in both Fig. 2 and Fig. 3 would result in a correct ordering of slacks, i.e., $A$, $B$, $C$.

Denote the set of all paths in a design as $S_0 = \{S_1, S_2, \ldots, S_n\}$. Then $t^{th}$ tier criticality probability of slack $S^{(t)}$ is mathematically defined as

$$p^{(t)} = \max (p_i = P(S_i \leq S_j)) \forall S_i \neq S_j \in S_{t-1}$$

$$S^{(t)} = \{ S_i | P(S_i \leq S_j) = p^{(t)} \} \forall S_i \neq S_j \in S_{t-1}$$

with the set of slacks $S_t$ defined as

$$S_t = S_0 \setminus S^{(1)} \cup \ldots \cup S^{(t)} \forall 1 \leq t \leq n$$

In other words, we organize slacks into a hierarchy of tiers. Each tier $t$ in this hierarchy has exactly one slack $S_t$, whose tiered criticality probability $p^{(t)}$ is the largest among all slacks excluding those belonging to lower tiers. We have the following theorem.

**Theorem 1:** Ordering slacks $S = \{S_1, S_2, \ldots, S_n\}$ based on tiered criticality probabilities as defined by (4) to (6) is unique, and the lower the tier number of the slack, the more critical that slack.

The proof is straightforward, since there is exactly one slack $S_t$, whose tiered criticality probability $p^{(t)}$ is the largest among all slacks in higher tiers.

For the same examples as shown in Fig. 2 and Fig. 3, we would obtain a hierarchy of three tiers, and the first tier would have slack $A$, and the second tier would have slack $B$, and the last tier would have slack $C$. Therefore, we would order these three slacks properly as $A$ is the most critical one, $B$ is the second critical one, while $C$ is the least critical one.

### C. Algorithms

The computation of tiered criticality probabilities can be done in a straightforward manner. According to (4) and (5), for each tier $t$, we compute $p_t$ for all remaining slacks, and find the one with the highest value of $p_t$, which would be the tiered criticality $p^{(t)}$, and the corresponding slack would be put into the $t^{th}$-tier as $S^{(t)}$. For each tier, we need to call the algorithm to compute $p_t$ as shown in (3). In other words, the computation of $p_t$ as shown in (3) has to be repeated $t$ times.
times. In the following, we discuss two speed-up techniques to reduce this computation, one for Monte Carlo and one for analytical computation [4].

1) Monte Carlo Simulation: Because the complexity of one Monte Carlo run of \( m \) trials for \( n \) slacks is \( O(nm) \), running Monte Carlo \( t \) times would have a total complexity of \( O(tm) \). The goal of speeding up Monte Carlo based tiered criticality computation is to run Monte Carlo only once instead of \( t \) times, hence reducing the complexity to \( O(nm) \). The idea is to use a smart book-keeping technique on the Monte Carlo trial data, and to compute the tiered criticality probabilities by repeatedly using the same set of Monte Carlo trial data.

2) Analytical Computation: For a given set of \( n \) slacks, their criticality probabilities \( p_i \) (3) can be also computed by using the statistical min operation as discussed in [1]. Moreover, by leveraging a binary tree data structure, the authors of [4] have shown that the \( p_i \) via (3) can be computed in linear time complexity \( O(n) \) as follows.

We build a balanced binary partition tree as shown in Fig. 4, where each leaf node corresponds to one of the \( n \) slacks. Through the first bottom-up traversal of the tree as shown in the top plot of Fig. 4, we compute the minimum slack of every node’s two direct child nodes, and save it at the node (at the root, we have the minimum slack of all \( n \) slacks). Through the second top-down traversal of the tree as shown in the bottom plot of Fig. 4, we compute the complement slack of every node by taking the minimum of its parent node’s complement slack and its sibling node’s slack (with the root’s complement slack initialized as infinity). At the leaf node \( i \), the complement slack would be the minimum slack of the remaining \( n-1 \) slacks, i.e., \( \min_{j\neq i}(S_j) \). Then \( p_i \) via (3) for each slack \( S_i \) can be computed.

Both the bottom-up and top-down traversals have complexity of \( O(n) \), so this algorithm has complexity of \( O(n) \). For \( t \) tiered criticalities, we need to call this binary tree based algorithm \( t \) times, thus the total complexity is \( O(nt) \).

![Fig. 4. Binary tree for fast computation of tiered criticalities.](image)

We propose a speedup technique to compute all tiered criticalities by re-using most of the results saved in the binary tree for the prior tiers’ criticality computation without reconstructing the binary tree \( t \) times. The idea is better illustrated via a simple example as shown in Fig. 4, where \( S_1 \) through \( S_8 \) represent eight slacks. Assume that for the first tier computation, we find that slack \( S_8 \) is the most critical one, and we put \( S_8 \) into the first tier and record its tiered criticality. For the second tier, our goal is to compute \( p_i \) for the remaining slacks but without reconstructing the binary tree for them. The idea is to replace the slack \( S_8 \) with an infinity valued slack, and then do a bottom-up traversal along the path from this leaf node to the root as shown in the top plot of Fig. 4, and update the minimum slack stored at each node along the way. Next, we conduct a top-down traversal from the root to all leaf nodes and update the complement slacks stored at all nodes as before. Then \( p_i \) for the remaining slacks can be easily computed at valid leaf nodes (nodes whose slack is not infinity). From this computation, we can determine the second tier criticality easily. We repeat this procedure until all \( t \) tiered criticalities are found.

In terms of complexity, this is the same as \( O(nt) \), but in practice, the multiplicative constant in the complexity analysis is much lower than reconstruction of the binary tree for each tier.

IV. EXPERIMENTAL RESULTS

We have implemented the proposed algorithms to compute the tiered criticality probabilities inside a statistical static timing analysis tool that was developed internally. Industrial 90 nm ASIC designs are used for our testing with the process variations determined by foundry data.

A. Tiered Criticality

For comparison purposes, we have also implemented the algorithm of [4] to compute the criticality probability as defined by (3), which we denote as non-tiered criticalities.

Table I shows the comparison of the order sequences of the top few end points for one design. We report both the ordering, the name of end points, their respective mean and sigma of end point slacks. The first set of rows shows results based on non-tiered criticalities; the second set shows results from tiered criticalities; and the third set shows a different implementation of tiered criticalities as we shall discuss below.

We observe that the most critical end point, GRP.1, is the same for both approaches, and this is expected as both take into account the full process space and the most critical slack has the highest probability of being smaller than the rest. However, we can clearly see that the rest of the ordering sequences based on these two metrics are quite different.

For example, by comparing the slack values of RXD.17 with those of GRP.7, we can tell that GRP.7 has higher probability of being smaller than RXD.17. But according to non-tiered criticalities, RXD.17 is ranked as second while GRP.7 was ranked as seventh. This illustrates the probability masking problem of correlation masking. By computing the correlation between GRP.7 and GRP.1, we find that they have very high correlation (\( \rho = 0.98 \)). In computing their respective non-tiered criticalities, there is almost no chance for GRP.7 being smaller than GRP.1. Hence the non-tiered criticality for GRP.7 is much lower. On the other hand, the slack of RXD.17 is
much larger than GRP.7, but because the correlation between RXD.17 and GRP.1 is low, there is still some chance for RXD.17 being smaller than GRP.1. In other words, the non-tiered criticality for RXD.17 is non-zero, and it is thus ranked as more critical than GRP.7.

On the contrary, if we order all slacks based on tiered criticalities as defined by (4), we obtain the proper ordering among different slacks. For example, now GRP.7 is ranked as second while RXD.17 is ranked as 19th. This is true for other end point slacks as well. We should also point out that it is important to keep only one slack for one tier if we want to fully avoid the probability masking problems. But in practice, we may also implement the tiered criticality computation in a slightly different way. That is, instead of keeping one slack for each tier as defined by (5), we may pull out multiple slacks for each tier. For example, with a fixed tier size of \( q \), we would pull out the worst \( q \) slacks for each tier and assign them to that tier; or we could pick up a subset of slacks whose criticalities are above a threshold value \( \overline{pr_{th}} \), i.e., replacing (5) with

\[
S^{(i)} = \{S_i \mid P(S_i \leq S_j) \geq \overline{pr_{th}} \} \forall S_i \neq S_j \in S_{t-1}. \tag{7}
\]

For the same example, when we choose \( \overline{pr_{th}} = 0.05 \) to get the hierarchy of tiers and order slacks according to this hierarchy, we obtain the third set of rows as shown in Table I, where we have GRP.1 ranked as the first, RXD.17 ranked as the second, while GRP.7 is ranked as the third. This result is better than the one based on non-tiered criticality, but still, the correlation masking problem exists that prevents GRP.7 from being ranked more critical than RXD.17.

All the above observations are consistent across various designs that we have tested.

**TABLE I**

<table>
<thead>
<tr>
<th>Order</th>
<th>End point</th>
<th>Mean</th>
<th>Sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-tiered Criticalities based on (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>GRP.1</td>
<td>0.315</td>
<td>0.0715</td>
</tr>
<tr>
<td>2</td>
<td>RXD.17</td>
<td>0.361</td>
<td>0.009</td>
</tr>
<tr>
<td>3</td>
<td>RXD.3</td>
<td>0.362</td>
<td>0.009</td>
</tr>
<tr>
<td>4</td>
<td>RXD.0</td>
<td>0.363</td>
<td>0.009</td>
</tr>
<tr>
<td>5</td>
<td>RXD.1</td>
<td>0.322</td>
<td>0.071</td>
</tr>
<tr>
<td>6</td>
<td>GRP.9</td>
<td>0.322</td>
<td>0.009</td>
</tr>
<tr>
<td>7</td>
<td>GRP.7</td>
<td>0.319</td>
<td>0.072</td>
</tr>
</tbody>
</table>

| Tiered Criticalities based on (4) and (5) |      |       |
| 1     | GRP.1     | 0.315 | 0.0715 |
| 2     | GRP.7     | 0.319 | 0.072  |
| 3     | GRP.11    | 0.322 | 0.009  |
| 4     | GRP.9     | 0.322 | 0.071  |
| 5     | RXD.17    | 0.361 | 0.009  |
| Tiered Criticalities based on (4) and (7) with \( \overline{pr_{th}} = 0.05 \) |      |       |
| 1     | GRP.1     | 0.315 | 0.0715 |
| 2     | RXD.17    | 0.361 | 0.009  |
| 3     | GRP.7     | 0.319 | 0.072  |

**B. Application**

Next, we show how the tiered criticalities can be used to guide path selection for at-speed testing. We have implemented the same branch-and-bound path selection algorithm as [8]. For comparison, in the first run, we find the top \( m \) critical end points based on non-tiered criticalities; while in the second run, we also find \( m \) critical end points but based on tiered criticalities. Afterwards, both approaches employ the same test quality metric (TQM) to guide the path selection procedure to select \( n \) paths with the best TQM leading to these \( m \) end points. To compare the testing quality of these two sets of paths, we sort their individual path TQM, and report the difference between the corresponding sorted path TQM.

Fig. 5 shows the TQM differences between paths selected based on tiered criticalities and paths selected based on non-tiered criticalities. The positive difference in TQM shows that paths selected from tiered criticalities have higher TQM, hence higher testing quality. According to Fig. 5, we observe that the first hundred paths resulting from both approaches have zero differences in TQM, which means that both are able to find the top hundred critical paths. But for the remaining thousands of paths, the differences are always positive, and this convincingly shows that paths selected from tiered criticalities have better TQM. This result is expected, as tiered criticalities provide more accurate ability to distinguish among sub-critical paths, selection of which is also important to protect us against both inaccurate modeling issues and sensitization issues.

Finally, we report runtime results to compute tiered criticalities. When Monte Carlo simulation is used, we observe that the proposed speed-up technique indeed consumes almost as much time as one run of Monte Carlo. On the other hand, if we do not use the speedup technique, the total runtime would grow linearly as more tiers of criticality are requested. When the binary tree based analytical computation is used for tiered criticality computation, employing the proposed speedup technique can save us about 30% of runtime compared to the same analytical computation without using the speedup technique.

**V. Conclusion and Discussion**

We have shown that statistically ordering a set of correlated timing quantities is difficult due to both dominance
and correlation masking problems. A novel metric, called tiered criticalities, have been proposed to solve these masking problems. We have shown that ordering based on tiered criticalities can guarantee the uniqueness of ordering while correctly taking into account the full process space coverage. Two novel algorithms have been developed to compute tiered criticalities efficiently, one for Monte Carlo simulation, and one for analytical computation.

We envision that this new metric holds the key to solving many potential CAD problems in the presence of process variations, for example, statistical timing report, design optimization, path traversal and selection for testing. As a demonstration, we have applied tiered criticalities in selecting paths for at-speed delay testing. Our results have shown that the quality of path selection is improved significantly compared to the state-of-the-art. Our future work will explore the application of this new metric for other CAD problems.

REFERENCES


