Incremental Criticality and Yield Gradients

Jinjun Xiong, Vladimir Zolotov, Chandu Visweswariah
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
{*jinjun, zolotov, chandu*}*@us.ibm.com*

Abstract— **Criticality and yield gradients are two crucial diagnostic metrics obtained from Statistical Static Timing Analysis (SSTA). They provide valuable information to guide timing optimization and timing-driven physical synthesis. Existing work in the literature, however, computes both metrics in a non-incremental manner, i.e., after one or more changes are made in a previously-timed circuit, both metrics need to be recomputed from scratch, which is obviously undesirable for optimizing large circuits. The major contribution of this paper is to propose two novel techniques to compute both criticality and yield gradients efficiently and incrementally. In addition, while node and edge criticalities are addressed in the literature, this paper for the first time describes a technique to compute path criticalities. To further improve algorithmic efficiency, this paper also proposes a novel technique to update "chip slack" incrementally. Numerical results show our methods to be over two orders of magnitude faster than previous work.**

I. INTRODUCTION

As CMOS technology continues to scale down to 45 nm and beyond, process variation effects become increasingly important for design closure. Statistical Static Timing Analysis (SSTA) is a popular method of dealing with process variations [1], [2], [3], [4]. Unlike deterministic timing, the critical path in statistical timing is not unique, as different paths may be critical in chips manufactured under different process conditions. The non-uniqueness of the critical path gives rise to problems in optimization in which all the critical paths across the entire process space must be targeted for optimization.

Criticality probability is a good metric for discrete circuit optimization in the presence of process variations. The criticality of a path is the probability of manufacturing a chip in which the path of interest is critical. The criticality of a node or edge of the timing graph is defined as the probability of this node or edge being on the critical path [2]. Knowing the criticality probability helps designers to rank cells in variation-aware physical optimization [5]. Early attempts at criticality computation suffered from large errors due to neglecting correlation [2]. The cutset-based method [6] was the first algorithm that correctly computed edge/node criticalities. However, computing path criticalities is not directly possible through the cutset method. This problem is still open in the literature.

Incremental timing [7] is a corner-stone of optimization and physical synthesis, in which timing quantities can be updated incrementally after one or more changes are made to the circuit. Unfortunately, the cutset-based method for criticality computation is inherently non-incremental. This makes criticality less favorable for optimization applications, described in [8], [5].

In contrast to criticalities, *yield gradients* comprise a more appealing metric for continuous circuit optimization because they include information about sensitivities of the whole circuit delay distribution with respect to the delay distribution of each timing edge. The method proposed in [9] computes only the sensitivity of the mean value of the circuit delay with respect to the mean of timing edge delay. However, this is not enough to guide optimization because any change of cell sizes affects not only the mean but other parameters of the delay distributions. Even computing sensitivities of variance does not fully solve the problem because it does not take into account correlations between delays. Moreover, the technique proposed in [9] requires traversal of the entire timing graph, which makes it too expensive for incremental computation. Another method for computing yield sensitivity was proposed in [10] by using numerical perturbation, which is neither efficient nor accurate. A comprehensive yield gradient computation technique was recently developed in [11]. But that technique utilizes the same cutset-based concept as in [6], thus rendering it non-incremental.

The main contributions of this paper are as follows: (1) two novel techniques for incremental criticality and yield gradient computation that are two orders of magnitude faster than the state-of-the-art; (2) efficient incremental computation of path criticality; and (3) a method to efficiently update the chip slack of the design incrementally.

The rest of this paper is organized as follows. We review parameterized SSTA and cutset-based criticality computation in Section II. We then propose our novel incremental criticality and yield gradient computation algorithms in Sections III and IV. Efficient "chip slack" updates are described in Section V and experimental results in Section VI. We conclude this paper in Section VII.

II. PRELIMINARIES

A. Statistical Static Timing Analysis

In static timing analysis (STA), we represent a sequential digital circuit as a directed acyclic timing graph. Fig. 1 shows an example of a sequential circuit with both data and clock networks. The timing graph corresponding to the data network of this circuit is shown in Fig. 2 in solid lines.

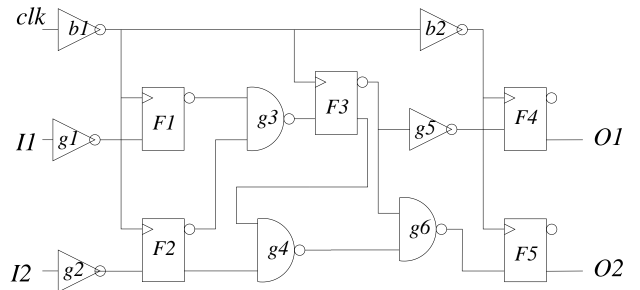

Fig. 1. Small sample sequential circuit.

In deterministic STA, all timing quantities are scalars. In contrast, for parameterized statistical STA (or SSTA), all timing quantities are represented as random variables parameterized by the underlying sources of variation, which in turn are modeled as random variables with certain known distributions. Recent progress in SSTA can handle cases when the variations can be either Gaussian [1], [2] or non-Gaussian [12], [13]; the functional form of the timing quantities can be either linear [1], [2] or nonlinear [14], [15], [16], [13].

SSTA computes the earliest and latest arrival times (ATs) at nodes

of the timing graph by a levelized propagation[1]. At a node, AT is computed as the maximum of ATs at its incoming edges, while the required arrival time (RAT) is computed as the minimum of RATs at its outgoing edges. The difference between RAT and AT is timing slack. Modern block-based SSTA engines can compute all ATs, RATs, and slacks in the timing graph efficiently and incrementally.

B. Cutset-Based Criticality Computation

For convenience, the timing graph is augmented by a single *source* node and a single *sink* node as shown in Fig. 2. The source node S connects to all primary inputs via virtual edges whose delays are the ATs at the primary inputs. The sink node K connects to all primary outputs or timing test nodes (corresponding to the setup and hold tests between ATs at data and clock pins of flip-flops) with virtual edges. The delay of such a virtual edge is the negative of the RAT of the corresponding node. Virtual edges are shown in dotted lines in Fig. 2. By construction, path delay in this *augmented timing graph* is the negative of the path slack in the original timing graph. Therefore, for simplicity, delays and slacks are used here interchangeably whenever there is no ambiguity. The AT at the sink node equals the *chip slack* C, i.e., the maximum of the delay of all paths from the source to the sink. Strictly speaking, the slack of the design is the negative of this quantity.

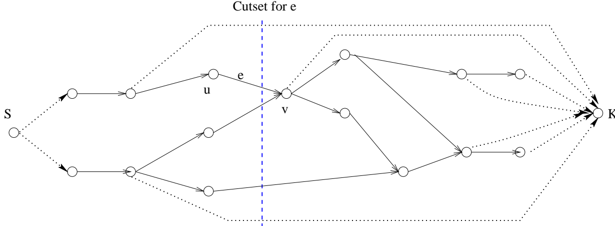

Fig. 2. Timing graph and a cutset for edge e.

The *edge slack* A of a particular edge is the maximum delay of all paths going through this edge. For edge e going from node u to v, the edge slack can be expressed as

$$A = AT_u + d_e - RAT_v, \tag{1}$$

where d_e is the delay of edge e; AT_u is AT at node u; RAT_v is RAT at node v. From static timing, it follows that AT_u is the maximum delay of all paths from the source node to u; and RAT_v is the minimum of the negative of delays of all paths from v to the sink node.

The main idea of the cutset method [6] is to divide all paths from the source to the sink into two disjoint sets. The first group comprises paths going through the edge e. The second group comprises paths not going through e. The $\max$ delay of the paths going through e is the *edge slack* A, computed as (1). The $\max$ delay of the paths not going through e is called the *complement edge slack* B. According to the definition, the edge criticality of e is the probability of edge slack A being greater than or equal to the complement edge slack B, i.e.,

$$t = P(A \geq B). \tag{2}$$

We also call t the *tightness probability* of A over B.

To compute complement edge slack B, [6] proposed to find a cutset that intersects the edge of interest and separates the source and sink as shown in Fig. 2. Each path of the graph that does not pass through e passes through one and only one of the other cutset edges. Then the complement edge slack B is computed as maximum of those edges' slacks, i.e.,

$$B = \max_{i \in cutset/e} A_i. \tag{3}$$

However, the cutset method cannot compute path criticality because of the difficulty in computing complement path slacks. So far, there is no existing work that computes path criticalities efficiently.

It has been shown in [6] that the cutset method computes all edge/node criticalities in linear time with respect to circuit size. However, this does not mean that the cutset method can compute an individual edge/node's criticality in constant time. The complexity of updating criticality of a single edge/node by the cutset method is still linear because of the two steps involved: (1) finding a proper cutset for the edge of interest; and (2) computing of all edge slacks A_i in the cutset and calculating their maximum. Therefore, incremental updating of edge/node criticalities by the cutset method is inefficient.

In the following, we propose two novel methods to overcome this drawback. Both methods are incremental because they only require knowledge of ATs, RATs and edge delays which can be updated incrementally by SSTA. Moreover, both methods permit us to compute path criticalities incrementally, too.

III. Incrementality via Probability Identity

Our first incremental criticality computation technique relies on a simple identity from probability theory.

Theorem 1: For two random variables (possibly correlated), A and B, with arbitrary distributions, the following identity holds:

$$P(A \geq B) = P(A \geq A \& A \geq B) = P(A \geq \max(A, B)). \tag{4}$$

In other words, the probability that A exceeds B is the same as the probability that A exceeds $\max(A, B)$.

According to Theorem 1, we can compute edge criticality as

$$t = P(A \geq B) = P(A \geq \max(A, B)) = P(A \geq C), \tag{5}$$

where A is the edge slack, B is the complement edge slack, and $C = \max(A, B)$ is the chip slack, which is the $\max$ delay of all paths from the source to the sink or simply AT at the sink node. In other words, the criticality of edge e is the probability that its edge slack dominates the chip slack. Thus, we have

$$t = P((AT_u + d_e - RAT_v) \geq AT_{sink}). \tag{6}$$

Therefore, to update edge criticality t, we need update only AT_u, d_e, RAT_v, and AT_{sink}. An incremental block-based SSTA engine, such as [2], can do this efficiently and incrementally. In fact, after incrementally updating these quantities, computing edge criticality by (6) can be done in constant time, as it only involves one statistical addition, one statistical subtraction and tightness probability computation.

Similarly, we can compute criticality of node u in constant time as

$$t = P(A \geq B) = P((AT_u - RAT_u) \geq AT_{sink}), \tag{7}$$

where A is the $\max$ of all paths through u and B is the $\max$ of all paths not through u.

Given a path Π with a path slack A, we can compute the path criticality in constant time too:

$$t = P(A \geq B) = P(\sum_{e \in \Pi} d_e \geq AT_{sink}), \tag{8}$$

[1]For brevity, only late-mode timing is considered here.

where A is the slack of path Π and B is the max of all path slacks except Π.

This method for computing edge, node and path criticality does not depend on any specific distributions or any particular delay models, be they Gaussian or non-Gaussian, linear or nonlinear. The method can be applied in the context of any block-based SSTA engine. All that is required is the computation of the $\max$ operation and tightness probabilities. This method is exact as long as we use an exact $\max$ operation. In practice, the statistical $\max$ operation is approximated, leading to inaccuracy in criticality computation. However, in the context of discrete circuit optimization, the exact accuracy is not as important as the correct ranking of cells. Therefore, it is desirable to keep the fidelity between the order based on metric prediction and the order of real criticality. The high fidelity of the proposed method will be confirmed in our experiments.

IV. Incrementality via Reconstruction of Complement Slack

In this section, we show that, when Clark's formulas [17] are used to approximate the $\max$ operation [1], [2], as most efficient existing SSTA engines do, we can accurately reconstruct one operand from the result and the other operand. We show that unlike the deterministic maximum, Clark's approximation of the statistical maximum always allows this reconstruction. Using this method, we can efficiently compute both criticality and yield gradients in constant time. This technique also enables incremental computation.

A. Incremental Criticality

The major hurdle of incremental computation of edge slack by the cutset method is the explicit computation of the complement edge slack B via (3). Here, we propose a new way of computing B by directly solving the equation for chip slack

$$C = max(A, B), \tag{9}$$

where A is the edge slack, easily computable by (1).

Under the linear parameterized delay model [1], [2], chip slack C is represented as

$$C = c_0 + \sum_{i=1}^{N} c_i X_i + c_r X_c, \tag{10}$$

where c_0 is the mean value of C, X_i model the N process parameters following a standard Gaussian distribution, X_c models uncorrelated random variation that also follows a standard Gaussian distribution; and c_i and c_r are the sensitivities of C to the corresponding random variables. Then the variance of C is given by

$$\sigma_C^2 = \sum_{i=1}^{N} c_i^2 + c_r^2. \tag{11}$$

Similarly, the edge slack A and the complement edge slack B can be represented as $A = \alpha_0 + \sum_{i=1}^{N} \alpha_i X_i + \alpha_r X_a$ and $B = \beta_0 + \sum_{i=1}^{N} \beta_i X_i + \beta_r X_b$. Their covariance is given by

$$cov(A, B) = \sum_{i=1}^{N} \alpha_i \beta_i. \tag{12}$$

Their correlation coefficient is $\rho_{AB} = \frac{cov(A,B)}{\sigma_A \sigma_B}$ with σ_A^2 and σ_B^2 being variances of A and B, respectively.

The tightness probability [17] of A is given by

$$t = P(A \geq B) = \Phi\left[\frac{\alpha_0 - \beta_0}{\theta}\right], \tag{13}$$

where $\Phi[x]$ is the standard Gaussian cumulative distribution function (CDF), and

$$\theta \equiv \left\{\sigma_A^2 + \sigma_B^2 - 2\rho_{AB}\sigma_A\sigma_B\right\}^{1/2}. \tag{14}$$

Then the computation of C in the form of (10) is given by

$$c_0 = \alpha_0 t + \beta_0(1-t) + \theta\phi\left[\frac{\alpha_0 - \beta_0}{\theta}\right] \tag{15}$$

$$c_i = \alpha_i t + \beta_i(1-t) \tag{16}$$

$$c_r = \left(\sigma_C^2 - \sum_{i=1}^{N} c_i^2\right)^{1/2}, \tag{17}$$

where the variance of C is given by

$$\sigma_C^2 = (\sigma_A^2 + \alpha_0^2)t + (\sigma_B^2 + \beta_0^2)(1-t) + (\alpha_0 + \beta_0)\theta\phi\left[\frac{\alpha_0 - \beta_0}{\theta}\right] - c_0^2 \tag{18}$$

with $\phi[x]$ being the standard Gaussian probability density function (PDF).

From the above discussion, we know that the tightness probability t for computing C is the same as the edge criticality. Since chip slack C and edge slack A can be computed incrementally, we propose to compute t from A and C directly instead of computing it from A and B.

According to (16), we have

$$\beta_i = \frac{c_i - t\alpha_i}{1-t}. \tag{19}$$

Substituting (19) into (12), noting that $cov(A, C) = \sum_{i=1}^{N} \alpha_i c_i$ and after some simplification, we get

$$\frac{t\sigma_A + (1-t)\rho_{AB}\sigma_B}{\sigma_C} - t\frac{\alpha_r^2}{\sigma_A\sigma_C} = \rho_{AC}, \tag{20}$$

where ρ_{AC} is the correlation coefficient between A and C.

Based on (14) and (13), we know

$$\rho_{AB}\sigma_B = \frac{\sigma_A^2 + \sigma_B^2 - \theta^2}{2\sigma_A}, \tag{21}$$

$$\theta = \frac{\alpha_0 - \beta_0}{\Phi^{-1}(t)}, \text{ and} \tag{22}$$

$$\phi\left[\frac{\alpha_0 - \beta_0}{\theta}\right] = \phi\left[\Phi^{-1}(t)\right]. \tag{23}$$

Substituting them into (15) and (20), and after some simplification, we arrive at

$$\beta_0 = \frac{c_0 - t\alpha_0 - \alpha_0\frac{\phi\left[\Phi^{-1}(t)\right]}{\Phi^{-1}(t)}}{1 - t - \frac{\phi\left[\Phi^{-1}(t)\right]}{\Phi^{-1}(t)}}, \tag{24}$$

$$\sigma_B^2 = \frac{2(t\alpha_r^2 + \sigma_A\sigma_C\rho_{AC} - t\sigma_A^2)}{1-t} + \left(\frac{\alpha_0 - \beta_0}{\Phi^{-1}(t)}\right)^2 - \sigma_A^2. \tag{25}$$

After substituting (22), (23), (24) and (25) into (18), we see that the only unknown left in (18) is t. So we can solve the resulting equation for t by any standard root-finding technique. As this is a one-dimensional root-finding problem, and the range of t is between 0 and 1, computation of t is straightforward. Thus, knowing edge slack A and chip slack C, we can compute edge criticality t in constant time by solving a simple one-dimensional root-finding problem. Although it is difficult to prove theoretically that the above root-finding problem has only one-unique solution, in practice, we find that seems to be true based on our experiments.

The above reconstruction procedure is valid as long as t does not equal to 0 or 1. Special care need to be paid to handle these two cases. When $t = 0$, it means that A is fully dominated by B, thus B would be the same as C, and this can be detected by computing $P(C \geq A)$, which should lead to zero. When $t = 1$, it means that B is fully dominated by A, thus A would be the same as C, and this can be detected by checking similarity between A and C.

Node criticality can be computed in a similar way by expressing the slack of the node u as $A = AT_u - RAT_u$ and then applying the above procedure. Again this can be done in constant time and used for incremental updating of node slacks during circuit optimization.

Path slack can be efficiently computed by summing the delays of the path edges$A = \sum_{e \in path} d_e$. By applying the same procedure to path slacks, we can compute path criticality in constant time, too. Numerical results will be shown in Section VI to demonstrate the superior accuracy of the reconstruction method.

B. Incremental Yield Gradients

For statistical circuit optimization by transistor or gate sizing, it is important to know the sensitivity $\partial y/\partial w$ of the parametric chip yield to the transistor or cell width w. Assume that the chip slack has a Gaussian distribution with CDF $\Phi\{(C - c_0)/\sigma_C\}$, where c_0 and σ_C are the mean and standard deviation of chip slack. Then assuming that the chip satisfies timing requirements only if its slack is positive, the chip yield can be expressed as

$$y = 1 - \Phi\{(-c_0)/\sigma_C)\}. \tag{26}$$

Using vector notation for derivatives and chain rule operations, the sensitivity of yield to transistor width can be written as

$$\frac{\partial y}{\partial w} = \left(\frac{\partial y}{\partial c_0} \cdot \frac{\partial c_0}{\partial \overrightarrow{A}} + \frac{\partial y}{\partial \sigma_C} \cdot \frac{\partial \sigma_C}{\partial \overrightarrow{A}}\right) \cdot \frac{\partial \overrightarrow{A}}{\partial \overrightarrow{d_e}} \cdot \frac{\partial \overrightarrow{d_e}}{\partial w}, \tag{27}$$

where $\overrightarrow{A} = (\alpha_0, \alpha_1, \ldots, \alpha_N, \alpha_r)$ are the variational parameters of edge slack A of the timing edge e corresponding to the transistor of interest and $\overrightarrow{d_e}$ are the variational parameters of the delay d_e of that timing edge. For brevity we assume here that variation of transistor width w affects only one timing edge e. Otherwise this formula can easily be modified to take it into account all the affected edges. Also this formula can be modified to take into account the effect of input signal slew and load capacitance on gate delay.

Here the sensitivities $\partial \overrightarrow{d_e}/\partial w$ of the parameters of the delay canonical form d_e can be obtained from transistor-level simulation of the gate and interconnects corresponding to the timing edge e. Well-known direct or adjoint sensitivity techniques or even simple finite differencing can be used. The details of this computation are outside the scope of this paper.

The sensitivities $\partial \overrightarrow{A}/\partial \overrightarrow{d_e}$ of edge slack variational parameters to edge delay d_e can be computed by differentiating (1) and taking into account that neither AT_u nor RAT_v depends on the edge delay d_e.

The vector of derivatives (gradients) $\partial c_0/\partial \overrightarrow{A}$ and $\partial \sigma_C/\partial \overrightarrow{A}$ of chip slack mean and variance with respect to the variational parameters of edge slack A can be computed by differentiating equations (15) and (18). Let Φ and ϕ denote $\Phi(\frac{\alpha_0 - \beta_0}{\theta})$ and $\phi(\frac{\alpha_0 - \beta_0}{\theta})$. The results of this differentiation are shown below. The derivation itself can be found in [11], and is omitted due to lack of space.

$$\frac{\partial c_0}{\partial \alpha_0} = \Phi \tag{28}$$

$$\frac{\partial c_0}{\partial \alpha_i} = (\alpha_i - \beta_i)\frac{\phi}{\theta} \tag{29}$$

$$\frac{\partial c_0}{\partial \alpha_r} = \alpha_r \frac{\phi}{\theta} \tag{30}$$

$$\frac{\partial \sigma_c}{\partial \alpha_0} = \frac{1}{2\sigma_c}\left\{(2(\alpha_0 - c_0)\Phi + (\sigma_A^2 - \sigma_B^2)\frac{\phi}{\theta} + \theta\phi\right\} \tag{31}$$

$$\frac{\partial \sigma_c}{\partial \alpha_i} = \frac{1}{\sigma_c}\Big[\alpha_i \Phi - c_0(\alpha_i - \beta_i)\frac{\phi}{\theta} + (\alpha_i - \beta_i) \left\{\alpha_0 + \beta_0 + (\alpha_0 - \beta_0)\frac{\sigma_B^2 - \sigma_A^2}{\theta^2}\right\}\frac{\phi}{2\theta}\Big], \tag{32}$$

$$\frac{\partial \sigma_c}{\partial \alpha_r} = \frac{1}{\sigma_c}\Big[\alpha_r \Phi - c_0 \alpha_r \frac{\phi}{\theta} + +\alpha_r\left\{\alpha_0 + \beta_0 + (\alpha_0 - \beta_0)\frac{\sigma_B^2 - \sigma_A^2}{\theta^2}\right\}\frac{\phi}{2\theta}\Big]. \tag{33}$$

In order to use these equations, we need to know the variational parameters $\overrightarrow{B} = (\beta_0, \beta_1, \ldots, \beta_N, \beta_r)$ of the complement edge slack B. They can be obtained by the reconstruction method as shown earlier, and the relevant equations are collected below for convenience.

$$\beta_0 = \frac{c_0 - t\alpha_0 - \alpha_0 \frac{\phi[\Phi^{-1}(t)]}{\Phi^{-1}(t)}}{1 - t - \frac{\phi[\Phi^{-1}(t)]}{\Phi^{-1}(t)}} \tag{34}$$

$$\beta_i = \frac{c_i - t\alpha_i}{1 - t} \tag{35}$$

$$\beta_r = \left(\sigma_B^2 - \sum_{i=1}^{N} \beta_i^2\right)^{1/2} \tag{36}$$

where σ_B^2 is given by

$$\sigma_B^2 = \frac{2(t\alpha_r^2 + \sigma_A \sigma_C \rho_{AC} - t\sigma_A^2)}{1 - t} + \left(\frac{\alpha_0 - \beta_0}{\Phi^{-1}(t)}\right)^2 - \sigma_A^2. \tag{37}$$

The tightness probability $t = P(A \geq B)$ required by these formulas can be computed by the technique described in the previous section.

Thus, we showed that similarly to criticalities, yield gradients with respect to variational parameters of edge delay, transistor width or cell size can be computed in constant time. This creates the required basis for incremental updates during circuit optimization.

V. EFFICIENT CHIP SLACK UPDATE

Chip slack, i.e., AT at the sink node of the augmented timing graph, can be incrementally computed by SSTA, just the same as AT for any internal node. However, a straightforward implementation will not be efficient.

A distinguishing feature of the sink node is that it has an incoming edge connecting it to every flip-flop and primary output in the design as shown in Fig. 2. For a typical design, the number of flip-flops can be on par with the size of the design itself. Hence incrementally updating AT at the sink after one or more incremental changes to the circuit can be computationally expensive.

To solve this problem, we propose a tree-like structure to replace the single virtual sink node in the original augmented timing graph. The fan-in count of the nodes of the tree is maintained at a reasonable number n, so at most $\lceil O(log_n N)\rceil$ additional levels of the timing graph will be required, where N is the total number of timing tests. This concept is illustrated in Fig. 3(b), where a tree-like subgraph with $n = 3$ has been used to replace a large fan-in sink node in

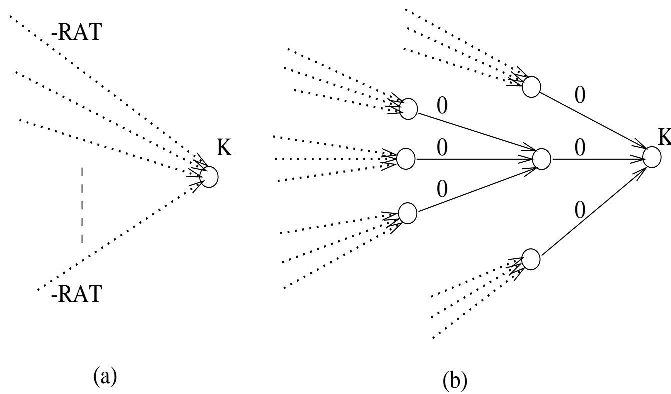

Fig. 3. Tree-like subgraph to connect the sink node.

Fig. 3(a). The tree edges closest to the primary outputs and data/clock pins of the flip-flops have delays equal to the negative of the RATs of the corresponding pins; all but the first level of tree edges have zero delay. When an incremental change is made in the circuit, even if the change ripples all the way through to the sink node of the timing graph, a small number of $\max$ operations will be required rather than a very wide $\max$ operation[2].

VI. Experimental Results

To illustrate our ideas, we have implemented the proposed two methods in C++ inside IBM's SSTA engine, EinsStat [2]. We denote the first method based on probability identity (4) as IDNTY, the second method based on reconstruction of complement slack as RECST, and the comparison base, the cutset-based criticality algorithm, as CUTSET. As shown in [6], the criticality computed by the CUTSET method is accurate compared to golden Monte Carlo simulation. We therefore only compare our approaches with CUTSET.

Five industrial ASIC designs based on 90 nm technology are used as test cases, and we denote them D1 to D5. The sizes of these designs are shown in the second row of Table I. The sources and amount of process variation are set according to foundry rules that are typical to this technology, including all front-end and back-end variations, such as Vth, NP-skew, NBTI, and metal variations.

TABLE I
Correlation of criticalities.

Designs	D1	D2	D3	D4	D5
Placeable objects	31K	135K	387K	1.5M	2.2M
corr(IDNTY,CUTSET)	0.9989	0.9965	0.9997	0.9786	0.9856
corr(RECST,CUTSET)	1.0	1.0	1.0	1.0	1.0

We first compare the criticality results from our approaches with those from CUTSET. Because criticality is used to rank gates for discrete circuit optimization and correlation coefficient is a convenient way to capture the similarity (or fidelity) of ranking between two sequences of data, we compute the correlation coefficients between our results and CUTSET's, and report them in Table I. We see that the criticalities computed by our methods are highly correlated with those from CUTSET, which validates the quality of our results which are incrementally computable unlike CUTSET's.

As we have discussed, the IDNTY method is generally applicable to all non-Gaussian and nonlinear delay models. But because of the approximate computation of max and tightness probability in our SSTA engine, IDNTY incurs some error, which explains why the results between IDNTY and CUTSET are not exactly the same. In contrast, our RECST method only applies to the linear Gaussian model, hence its computation is exact in our engine. The perfect correlation of 1.0 also suggests that results from RECST are indeed as accurate as those from CUTSET.

[2] Another benefit of our method is to improve accuracy by optimal clustering of timing test points and primary outputs, because Clark's formulas [17] lose accuracy when performing wide n-way maxes as a series of binary $\max$ operations [18].

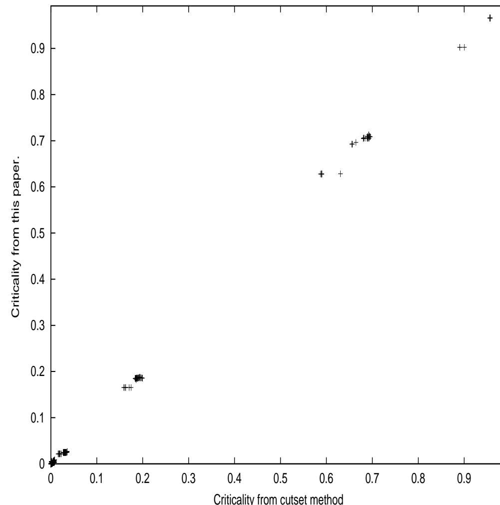

Fig. 4. Comparison of IDNTY criticality probabilities for D3.

Fig. 4 shows the comparison between our IDNTY method and the CUTSET method. We see that the criticality results from IDNTY are almost the same as the CUTSET method, as all criticality data fall on the 45-degree diagonal line. We further show the differences of criticalities computed by both methods in Fig. 5. Again, we can see that the differences are small, less than 4% across all edges. Similar plots are observed for all other test cases.

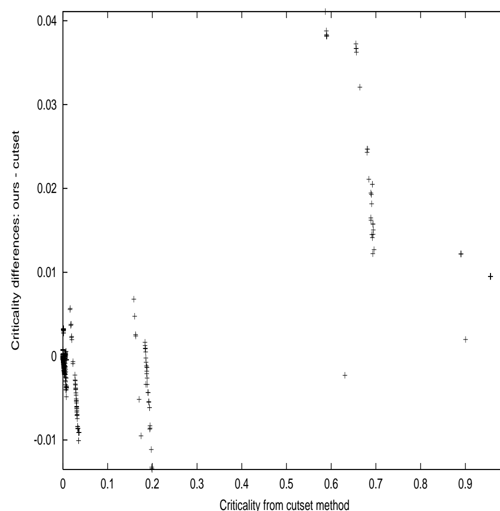

Fig. 5. Differences of IDNTY criticality probabilities for D3.

We compare the results between RECST and CUTSET in Table II. For each given design, we randomly chose a few hundred critical edges, and computed their criticalities by both methods. We compared the criticalities between RECST and CUTSET in terms of the maximum and average relative errors. As can be seen, the error is almost negligible, a small fraction of one percent in all cases. The

essence of our RECST method is the reconstruction of one of the max operands (the complement edge slack B) based on the result (chip slack C) and the other max operand (the edge slack A). We report the accuracy of the complement edge slack reconstructed by our RECST method in Table II as well. We clearly see that we are indeed able to reconstruct the complement edge slacks almost exactly in terms of both mean, sigma, and all sensitivities.

TABLE II
COMPARISON BETWEEN RECST AND CUTSET.

Design	D1	D2	D3	D4	D5
Relative error of criticality t					
Max.	2.0E-11%	7.7E-9%	1.0E-10%	3.5E-1%	1.0E-8%
Avg.	1.0E-11%	9.1E-10%	5.1E-11%	1.3E-3%	1.7E-9%
Relative error of complement edge slack's mean β_0					
Max.	1.3E-12%	3.4E-10%	1.0E-7%	2.0E-4%	2.9E-8%
Avg.	6.5E-15%	2.2E-11%	5.1E-8%	4.7E-6%	2.6E-9%
Relative error of complement edge slack's std. dev. σ_B					
Max.	2.7E-11%	5.6E-10%	2.6E-7%	8.4E-2%	6.6E-8%
Avg.	1.4E-11%	6.9E-11%	1.3E-7%	2.2E-3%	5.9E-9%
Relative error of complement edge slack's sensitivities β_i					
Max.	9.3E-12%	2.1E-9%	4.5E-7%	5.0E-1%	5.4E-7%
Avg.	3.9E-12%	8.0E-11%	5.1E-8%	1.6E-3%	6.0E-9%

Once we reconstruct complement edge slacks accurately, we can compute yield gradients analytically. Table III reports the relative error of yield gradients computed by our RECST method versus those computed through finite-differencing, which is done by perturbing each element of interest by a small fractional amount Δ. Ideally, the exact yield gradients should be obtained in the limit when Δ goes to zero. As we can see from Table III, as the perturbation amount decreases, the differences between our RECST method and the finite-differencing method reduce as well. From the trend, we conclude that results from finite-differencing would eventually converge to our analytic yield gradients when the perturbation Δ is infinitesimal.

TABLE III
YIELD GRADIENT COMPARISON BETWEEN RECST AND FINITE DIFFERENCING (FD).

FD Perturbation Δ		10%	1%	0.1%	0.01%
$\frac{\partial c_0}{\partial \alpha_j}$	Max.	5.2E-1	7.1E-2	7.2E-3	7.2E-4
	Avg.	1.5E-1	2.4E-2	2.5E-3	2.5E-4
$\frac{\partial \sigma_c}{\partial \alpha_j}$	Max.	3.9E-2	1.0E-2	1.1E-3	1.0E-4
	Avg.	1.8E-2	3.3E-3	3.2E-4	3.2E-5

Finally, we report run time and memory comparisons between our methods and the CUTSET method in Table IV for the entire designs. From Table IV, we see that for the same design, both IDNTY and RECST compute criticalities much faster than CUTSET, and the relative speed-up is more than two-orders of magnitude with much reduced memory consumption. Moreover, we should not loss sight that both of our approaches permit incremental computation while the CUTSET method does not. In other words, each subsequent query of criticality or yield gradients can be answered much faster than CUTSET. These superior run time and memory advantages of our approaches are particularly desirable for physical design optimization.

VII. CONCLUSIONS

Criticality and yield gradients are two crucial diagnostic metrics for statistical timing optimization in the presence of process variations. This paper presented two incremental algorithms for both criticality and yield gradient computation. Both techniques can easily be implemented in the context of an SSTA tool. When changes are made to an already-timed circuit, both metrics can be recomputed efficiently and incrementally. The method extends nicely to path criticality computation, a hitherto open problem. To further improve algorithmic efficiency, this paper also proposed a novel technique to update chip slack incrementally. Compared to the state-of-the-art cutset-based method, our new methods achieve more than two-orders of magnitude speedup with similar accuracy.

TABLE IV
RUN TIME AND MEMORY COMPARISONS.

Methods	CUTSET		IDNTY		RECST	
Designs	CPU (s)	Mem	CPU (s)	Mem	CPU (s)	Mem
D1	389.3	25.8M	1.3	0.38M	1.5	0.5M
D2	588.8	97.6M	8.0	0.9M	56.1	1.5M
D3	3880	361M	27.4	1.4M	221.9	2.7M
D4	7520	3.07G	123	5.3M	890	8.8M
D5	12847	4.4G	250	10.4M	1575	15.6M
Avg.	-	-	127×	287×	60×	176×

REFERENCES

[1] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. *Proc. ICCAD* , pages 621–625, November 2003.

[2] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. *Proc. DAC*, pages 331–336, June 2004.

[3] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. *Proc. ICCAD* , pages 900–907, November 2003.

[4] L. Zhang, Y. Hu, and C. C. Chen. Block based statistical timing analysis with extended canonical timing model. *Proc. ASPDAC*, pages 250–253, January 2005.

[5] M. R. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov. Gate sizing using incremental parameterized statistical timing analysis. *Proc. ICCAD*, pages 1029–1036, November 2005.

[6] J. Xiong, V. Zolotov, C. Visweswariah, and N. Venkateswaran. Criticality computation in parameterized statistical timing. *Proc. DAC*, pages 63–68, July 2006.

[7] R. P. Abato, A. D. Drumm, D. J. Hathaway, and L. P. P. P. van Ginneken. Incremental timing analysis. *U. S. Patent 5,508,937*, April 1996.

[8] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov. Circuit optimization using statistical timing analysis. *Proc. DAC*, pages 321–324, June 2005.

[9] X. Li, J. Le, M. Celik, and L. T. Pileggi. Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations. *Proc. ICCAD* , pages 844–851, November 2005.

[10] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester. Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation. *Proc. ICCAD* , pages 1023–1028, November 2005.

[11] V. Zolotov, J. Xiong, and C. Visweswariah. Computation of yield gradients from statistical timing analysis. *Proc. 2006 TAU*, pages 125–130, February 2006.

[12] J. Singh and S. Sapatnekar. Statistical timing analysis with correlated non-Gaussian parameters using independent component analysis. In *Proc. DAC*, pages 155–160, July 2006.

[13] L. Cheng, J. Xiong, and L. He. Non-linear statistical static timing analysis for non-Gaussian variation sources. In *Proc. DAC*, pages 250–255, June 2007.

[14] H. Chang, V. Zolotov, C. Visweswariah, and S. Narayan. Parameterized block-based statistical timing analysis with non-Gaussian and nonlinear parameters. *Proc. DAC*, pages 71–76, June 2005.

[15] L. Zhang, W. Chen, Y. Hu, J. A. Gubner, and C. C. Chen. Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model. In *Proc. DAC*, pages 83 – 88, June 2005.

[16] Y. Zhan, A. J. Strojwas, X. Li, and L. T. Pileggi. Correlation-aware statistical timing analysis with non-Gaussian delay distribution. In *Proc. DAC*, pages 77–82, June 2005.

[17] C. E. Clark. The greatest of a finite set of random variables. *Operations Research*, pages 145–162, March-April 1961.

[18] D. Sinha, H. Zhou, and N. V. Shenoy. Advances in computation of the maximum of a set of random variables. *Proc. ISQED*, pages 306–311, March 2006.