Clustering Based Pruning for Statistical Criticality Computation under Process Variations

Hushrav D Mogal*, Haifeng Qian†, Sachin S Sapatnekar* and Kia Bazargan*
*Department of Electrical and Computer Engineering, University of Minnesota,
Minneapolis, MN 55455, {mhush, sachin, kia}@umn.edu
†IBM Research, Yorktown Heights, NY, qianhaifeng@us.ibm.com

***Abstract*— We present a new linear time technique to compute criticality information in a timing graph by dividing it into "zones". Errors in using tightness probabilities for criticality computation are dealt with using a new clustering based pruning algorithm which greatly reduces the size of circuit-level cutsets. Our clustering algorithm gives a $150X$ speedup compared to a pairwise pruning strategy in addition to ordering edges in a cutset to reduce errors due to Clark's MAX formulation. The clustering based pruning strategy coupled with a localized sampling technique reduces errors to within 5% of Monte Carlo simulations with large speedups in runtime.**

I. Introduction and Previous Work

With scaling of technology, process parameter variations render the circuit delay as unpredictable [6], making sign-off ineffective in assuring against chip failure. Recent works concerning Statistical Static Timing Analysis (SSTA) in [1], [9] deal with this issue by treating the delay of gates and interconnects as random variables with Gaussian distributions, and predict the mean and variance of circuit delay to within a few percents.

With process variations in effect, no one path dominates the delay of the circuit [5]. Work in [9] proposes the concept of edge(node) criticality, which is the probability that an edge(node) lies on the critical path in a manufactured chip. Works like [2], [5], [10] compute the criticality probability of edges in a timing graph, using a canonical first order delay model.

In [9], the authors perform a reverse traversal of the timing graph assuming independence of edge criticalities despite structural and spatial correlations in the circuit. Subsequently, the work in [5] used sensitivity matrices to compute the criticalities of nodes and edges in the timing graph, with complexity potentially cubic in the number of principal components of the circuit due to the matrix multiplications involved. Work in [2] perturbs gate delays to compute its effect on the circuit output delay, using cutsets to simplify computation. The quadratic time complexity of this approach is reduced to linear in [10], where the criticality of edges in a cutset is computed using a balanced binary partition tree.

Our contributions are twofold. We propose a linear time algorithm to compute the criticality probability of edges (nodes) in a timing graph using the notion of cutsets. Edges crossing multiple cutsets are dealt with using a novel zone based approach, similar to [11], where old computations are reused to the greatest possible extent. We develop a clustering based pruning algorithm to effectively eliminate a large number of non-competing edges in cutsets of several thousand edges. The proposed clustering scheme helps order the MAX operations in a set, a source of significant error as shown in [8]. Localized sampling on the pruned cutset further reduces errors in edge criticalities to within 5% of Monte Carlo simulations, with large speedups in run-time when compared to a pairwise pruning strategy.

II. Background

A. Correlation Model and SSTA

We use the spatial correlation model in [1] to model intra-die parameter correlations. Briefly, the chip is divided into a uniform grid in which gates in a grid square have perfect correlation and gates in far-away grid squares have weak correlations. A covariance matrix of size equal to the number of grid squares is obtained for each modeled parameter. We model the length and width of gates, and the thickness, width and inter-layer dielectric thickness for interconnects. Process parameters are assumed Gaussian as in [1], [9] and using the principal component analysis (PCA) technique each correlated parameter is expressed as a sum of independent normal random variables, or principal components (PCs). Delays of edges in the timing graph are expressed in terms of the PCs and SSTA is then performed by a forward propagation on the timing graph. For more details, readers are referred to [1].

B. Definitions

Definition II.1 (Timing Graph). A timing graph $G(V, E)$ of a circuit is a directed acyclic graph with V nodes representing gate terminals and E edges representing connections between them. Primary inputs and outputs are connected, respectively, to a virtual source node, v_s, and a virtual sink node, v_t. The delay of an edge in G is represented by a probability density function (pdf).

This work was supported in part by the SRC under award 2007-TJ-1572.

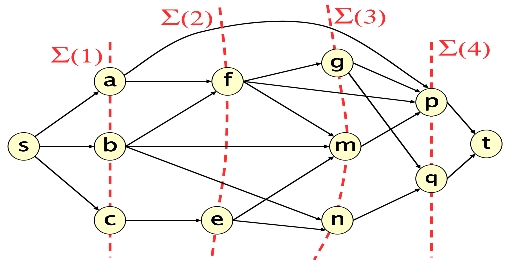

Fig. 1. Example timing graph G (depth L) illustrating cutsets $\Sigma(1) - \Sigma(4)$.

Definition II.2 (Cutset). A cutset Σ is a set of edges/nodes in G such that **every** v_s to v_t path passes through one and only one member of Σ.

Definition II.3 (Arrival Time (Required Time)). The arrival time AT (required time RT) at an edge/node in G is the maximum delay from any primary input (output) to the edge/node. Like delays, these are also $pdfs$.

Definition II.4 (Path Delay). The path delay of an edge/node (hereon, also referred to as edge delay) i in G is defined as $e_i = AT_i + RT_i$ and is a pdf. Each path delay e_i is represented in a canonical form in terms of independent principal components (PCs) p_j, as

$$e_i \quad = \quad \mu_i + \sum_{j=1}^{j=k} a_{ij} \cdot p_j \qquad (1)$$

Here a_{ij} is the coefficient of PC j for edge i and k is the total number of PCs. The mean of the edge delay is given by μ_i.

Definition II.5 (Complementary Path Delay). Given a cutset, Σ, in a timing graph G, the complementary path delay of an edge/node $k \in \Sigma \subset G$, $e_k{}'$, is defined as the pdf of $MAX(\forall e_i : e_i \in \Sigma, e_i \neq e_k)$.

Definition II.6 (Local Criticality). The local criticality (also referred to as tightness probability [9]) τ_{ij} of edge e_i with respect to e_j, with means μ_i, μ_j and standard deviations σ_i, σ_j respectively, and correlation ρ_{ij}, is given by

$$\tau_{ij} \quad = \quad \Phi\left(\frac{\mu_i - \mu_j}{\theta}\right) \qquad (2)$$

where $\theta = \sqrt{\sigma_i^2 + \sigma_j^2 - 2 \cdot \rho_{ij} \cdot \sigma_i \cdot \sigma_j}$. The **degree of domination** of edge i over j is represented by τ_{ij}. Φ is the distribution (cdf) of a unit normal random variable $\mathcal{N}(0, 1)$. It can be shown that $\tau_{ji} = 1 - \tau_{ij}$.

Definition II.7 (Global Criticality). The global criticality T_i (also referred to as criticality hereon) of edge e_i in cutset Σ is the probability that it has maximum delay among all the edges in the cutset, i.e.,

$$= \quad \Pr(e_i \geq e_i{}') \quad \text{(see Def. II.5)} \qquad (3)$$

T_i is also referred to as the **criticality probability** of e_i. It follows that the global criticality of an edge e_i in Σ cannot be greater than its local criticality with respect to any other edge in Σ, i.e.,

$$T_i \quad \leq \quad \tau_{ij} \quad \{ \forall \ e_j \ \in \ \Sigma, \quad e_j \neq e_i \ \} \qquad (4)$$

Definition II.8 ($\mathbf{MAX_\theta}$). The maximum MAX_θ of two normal random variables e_i and e_j in canonical form using Clark's formulation [3] is given by $c = MAX_\theta(i, j)$, where ϕ is the density (pdf) function of $\mathcal{N}(0, 1)$ and

$$\begin{aligned} \mu_c &= \tau_{ij} \cdot \mu_i + \tau_{ji} \cdot \mu_j + \theta \cdot \phi\left(\frac{\mu_i - \mu_j}{\theta}\right) \\ \sigma_c^2 &= \tau_{ij} \cdot (\sigma_i^2 + \mu_i^2) + \tau_{ji} \cdot (\sigma_j^2 + \mu_j^2) \\ &\quad + (\mu_i + \mu_j) \cdot \theta \cdot \phi\left(\frac{\mu_i - \mu_j}{\theta}\right) - \mu_c^2 \\ c_k &= \tau_{ij} \cdot a_{ik} + \tau_{ji} \cdot a_{jk} \end{aligned} \qquad (5)$$

III. Statistical Criticality Computation

This section describes a quadratic time complexity statistical criticality (SC) algorithm followed by details of our linear time approach.

A. Cutset Computation

Fig. 1 illustrates cutset computation on a timing graph G. We topologically traverse G, ordering nodes as per their levels, l, denoted $\Sigma_n(l)$, and edges crossing l denoted $\Sigma_e(l)$. For instance, $\Sigma_n(2) = \{n_e, n_f\}$ and $\Sigma_e(2) = \{e_{ap}, e_{bm}, e_{bn}\}$. $\Sigma(l) = \Sigma_n(l) \cup \Sigma_e(l)$ forms a cutset by Def. II.2 since every v_s to v_t path in G must pass through at least one member of $\Sigma(l)$ and its elements are disjoint. Our aim is to compute the criticalities of all edges in G, using Def. II.7. Towards this end, the topological level-enumerated cutsets are necessary and sufficient since they cover all the nodes and edges in G. The number of cutsets equals the number of levels L in G. To compute the criticality of all edges in G, we substitute nodes in Σ_n with their fanout edges. For instance, cutset 2, $\Sigma(2) = \{e_{em}, e_{en}, e_{fg}, e_{fp}, e_{fm}, e_{ap}, e_{bm}, e_{bn}\}$.

B. BSC: Basic SC Algorithm

The simplistic approach **BSC** shown in Algorithm 1 computes the **global criticalities** of all edges in timing graph G. Step 1 performs a forward and reverse SSTA to compute path delays ($AT + RT$) of all edges (nodes) in G, followed by a topological ordering of G into levels to compute its cutsets in Σ. Steps 3-8 compute the criticality of each edge in Σ.

Algorithm 1 BSC $(G(V, E))$

```
1: Perform forward and reverse SSTA on $G$
2: Topologically order $G$ and compute its cutsets $\Sigma$
3: for all cutsets $\Sigma \in G$ do
4:     for all edges $e_i \in \Sigma$ do
5:         $e_i{}' = MAX(\forall e_j \in \Sigma,\ e_j \neq e_i)$ {see Def. II.5}
6:         $T_i = \tau_{ii'}$ {see Def. II.6}
7:     end for
8: end for
```

Definition III.1 (mc-edge). An mc-edge is an edge with end level at least one greater than its start level. With our level enumerated cutsets, these are edges which cross over at least one cutset. In Fig. 1, e_{ap}, e_{bm} and $e_{bn} \in \Sigma_e(2)$ are a set of mc-edges crossing level 2. However, edges like e_{af} and e_{bf} are not mc-edges since they start at level 2 and end at level 3 with no cross over.

Due to mc-edges, each cutset Σ can potentially contain $O(E)$ edges. Moreover, since Step 5 computes the complementary path delay of an edge in time linear in the size of Σ, over all cutsets in G, Algorithm 1 has a complexity of $O(LE^2)$. Sections III-C and III-D discuss methods to reduce the time complexity of the basic approach.

C. Linear Time Book-Keeping

Definition III.2 (Ordered Lists). Given an arbitrary set $\Sigma = \{e_1, e_2, \ldots e_n\}$ of n random variables, we define ordered lists Υ_F and Υ_R as follows

$$\Upsilon_F(i) = MAX(e_1, \ldots e_i) \tag{6}$$

$$\Upsilon_R(i) = MAX(e_i, \ldots e_n) \tag{7}$$

The global criticality of an edge $e_i \in \Sigma$ (Def. II.7) is

$$\begin{aligned} T_i &= \Pr(e_i > MAX(e_1, \ldots e_{i-1}, e_{i+1}, \ldots e_n)) \\ &= \Pr(e_i > MAX(\Upsilon_F(i-1), \Upsilon_R(i+1))) \end{aligned} \tag{8}$$

Computing Υ_F and Υ_R takes $2n$ MAX operations. Eq. 8 takes two MAX operations for a total of $4n$ MAX operations over all edges in Σ. Our ordered lists can thus compute criticalities of cutset edges in linear time.

D. Zone Computation

Using Υ_F and Υ_R, Steps 5-6 of Algorithm 1 take time $O(E)$ as compared to $O(E^2)$. However, since Algorithm 1 recomputes criticalities of mc-edges (Def. III.1) in every cutset Σ, over L cutsets we could take $O(LE)$ time, still a considerable slowdown. Moreover, the complementary path delay (Def. II.5) for each edge $e_i \in \Sigma$ (excluding mc-edges) at level l has in common all the mc-edges crossing l. The only information therefore needed, is the collective MAX of these mc-edges. Zones help us compute this MAX.

Definition III.3 (Zone). A zone Z_i is a maximal set of mc-edges with the end-level of any edge higher than the start-level of all edges in Z_i, i.e., edges enter a zone before any edge exits it.

We divide G into zones Z_i, which contain mutually exclusive sets of mc-edges and compute the MAX of these edges at any level by performing a MAX over all the zones. To do this however, we also need to keep track of the mc-edges which contribute to the MAX of a zone (called Z_{iMAX}) at each level of G. This is performed using pointers z_{if} and z_{ir} which maintain the history of mc-edges entering (recorded in Z_{iF}) and leaving (recorded in Z_{iR}) each zone Z_i respectively. The authors in [10] use an array based structure to keep track of mc-edges, with no notion of a maximal set.

E. ZSC: Zone Based SC Algorithm

Our **zone** based criticality computation technique, **ZSC**, is shown in Algorithm 2. Step 2 computes zones in G in time linear in the size of Σ_e. We then forward traverse G from v_s to v_t. Steps 4-5 update the forward and reverse history pointers of each zone Z_i, to compute Z_{iMAX} in constant time. In Step 6 we compute the MAX over all the currently active zones. Since we have on the order of $O(L)$ number of zones, over all levels of G this step takes $O(L^2)$ time. Finally using the book-keeping ordered lists from Section III-C, we compute **global criticalities** of edges in Σ in linear time. The overall runtime of our **ZSC** algorithm is therefore $O(E + L^2)$, which for a reasonably sized circuit is $O(E)$.

Algorithm 2 ZSC $(G(V, E))$

```
1: Steps 1-2 of Algorithm 1
2: Compute zones of all edges in $\Sigma_e$ {see Section III-A}
3: for $l = 1$ to $L$ do
4:     $\forall e_j$ with end-level $l$, $z_{jr} = z_{jr} + 1$; $Z_{jMAX} = Z_{jR}(z_{jr})$
5:     $\forall e_i$ with start-level $l$, $z_{if} = z_{if} + 1$; $Z_{iMAX} = Z_{iF}(z_{if})$
6:     $\forall k \in Z$, $Z_{MAX} = MAX(Z_{MAX}, Z_{kMAX})$
7:     $\Sigma = \{$fanouts of nodes in $\Sigma_n(l)\} \cup Z_{MAX}$
8:     Create $\Upsilon_F$ and $\Upsilon_R$. Compute $T_i$ $\forall$ $e_i \in \Sigma$ {see Section III-C}
9: end for
```

IV. Errors in **ZSC** and the **"ABC"** Problem

Algorithm 2 was run on the ISCAS89 benchmarks to compute the global criticalities (see Def. II.7) of all edges in the timing graph G. We compared our implementation with a Monte Carlo (MC) simulation and noted the absolute maximum difference in criticalities of edges (denoted δ hereon). The δ was larger than 50% (for example, an edge reported by MC as 80% critical was reported by Algorithm 2 as 30% critical). As an illustration of these errors, consider a cutset Σ with random variables a, b and c, each with independent PCs p_1 and p_2 (where p_i is an $\mathcal{N}(0,1)$ Gaussian), shown below,

$$\begin{aligned} a &= 4.000 + 0.5000 \cdot p_1 + 0.5000 \cdot p_2 \\ b &= 3.999 + 0.4999 \cdot p_1 + 0.5001 \cdot p_2 \\ c &= 3.800 + 0.6001 \cdot p_1 + 0.3999 \cdot p_2 \end{aligned} \tag{9}$$

Observe that a and b are nearly identical highly correlated random variables, and for any sample value of the $p_i{}'s$, $a \geq b$ (high correlation coupled with the difference in means ensures $\Pr(b > a) \approx 0.0$). We ran a MC simulation with 100000 samples to determine the global criticalities of a, b and c. Table I shows a comparison with Clark's formulation MAX_θ (see Def II.8). We obtained errors of 57% and 30% in the global criticality of a and b respectively, as shown in the last row of Table I. For better illustration purposes, Fig 2 depicts the scenario of Eq. 9, but using just one principal component (PC), p. We make the following observations.

TABLE I
COMPARISON OF MC AND MAX_θ FOR THE **ABC** PROBLEM

Method	T_a	T_b	T_c
MC	0.923	0.000	0.077
Clark	0.356	0.297	0.079
Error δ	56.7%	29.7%	0.2%

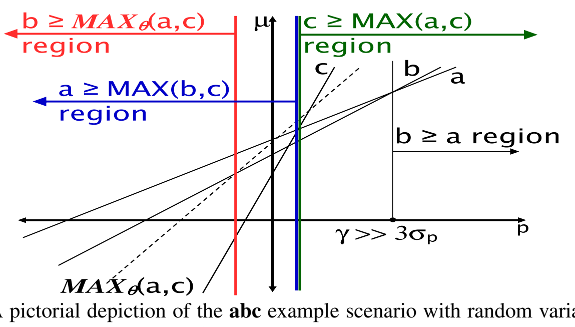

Fig. 2. A pictorial depiction of the **abc** example scenario with random variables a, b and c with one PC, p. The figure is not to scale.

(1) The local criticality of b with respect to a, i.e., $\tau_{ba} \approx 0$. This is indicated by a large value of $\gamma \gg 3\sigma_p$ (the region where $b \geq a$). Moreover, Clark's tightness probability formulation from Eq. 2 also gives $\tau_{ba} \approx 0$.
(2) Global criticality of b, $T_b \approx 0$. This is evident in Fig. 2 where regions $a \geq MAX(b,c)$ and $c \geq MAX(a,b)$ cover the entire probability space.

Observations 1 and 2 are consistent with Eq. 4. Now consider global criticality of b, using the cutset approach. We first compute complementary path delay $b' = MAX(a, c)$. It follows (Def. II.7) that $T_b = \Pr(b \geq MAX(a, c))$. With Clark's formulation MAX_θ, for the MAX, we get $T_b = \Pr(b \geq MAX_\theta(a, c)) = 0.297$.

Intuitively, for this scenario Clark's formulation is accurate with respect to local criticality t_{ba} of b, but it overestimates global criticality T_b, and is inconsistent with Eq. 4.

Definition IV.1 (**Local Errors**). With respect to Clark's formulation, edge e_i in cutset Σ is said to have **local errors** iff there exists some edge $e_j \in \Sigma$ for which its local criticality is smaller than its global criticality, i.e.,

$$\{\ \exists\ e_j\ \in\ \Sigma,\quad e_j \neq e_i\ \}\quad :\quad \tau_{ij}\quad <\quad T_i \tag{10}$$

In other words, Eq. 4 does not hold. The difference, $T_i - \tau_{ij}$ is called the **local error** of e_i. By definition, local errors always overestimate the criticality of an edge in Σ. In our toy example, b exhibits local errors with a magnitude of 0.297. Local errors were found to propagate in the **ZSC** algorithm, where variables (edges) like b that should never have been critical, were found to have a significant global criticality.

To further see the inconsistencies of the tightness probability approach using Clark's formulation we consider the event $a \geq MAX_\theta(b, c)$, with a represented as $a = a_0 + \sum a_i p_i = a_0 + a_p$. Variables b and c are represented similarly. We have,

$$\begin{aligned} \text{Event } \{a \quad &\geq \quad MAX_\theta(b, c)\} \quad \equiv \\ \tau_{bc}(a - b) + \tau_{cb}(a - c) \quad &\geq \quad \alpha + s(\tau_{bc} b_p + \tau_{cb} c_p) \end{aligned} \tag{11}$$

Here α and s are factors used to equate the first two moments (mean and standard deviation) of the non-Gaussian MAX with Clark's Gaussian MAX_θ approximation. Values of $\alpha \approx 0$ and $s \approx 0$ could result in a true event even for cases in which $b \leq a \leq c$ or $c \leq a \leq b$, overestimating the global criticality T_a of a. Values of $\alpha > 0$ could result in a false event even in the case that $a \geq b$ and $a \geq c$, underestimating T_a.

Definition IV.2 (**Global Errors**). With respect to Clark's formulation, edge e_i in a cutset Σ is said to have **global errors**, iff its computed criticality T_i differs from its true criticality and the edge does not exhibit local errors, i.e.,

$$T_i \quad \leq \quad \tau_{ij} \quad \{\ \forall\ e_j\ \in\ \Sigma,\quad e_j \neq e_i\ \} \tag{12}$$

and T_i is in error.
By definition, global errors can either overestimate or underestimate the criticality. For the toy example, T_a is underestimated by 0.567. Note that the value of T_a is consistent with Eq. 4, since both $\tau_{ab} = 1.0$ and $\tau_{ac} = 0.921$ are greater than $T_a = 0.356$

In summary, although local and global errors result from the linear approximation of Clark's MAX_θ operation, local errors are an artifact of the manner in which we compute global criticalities of edges in a cutset whereas global errors are more fundamental, overestimating or underestimating the true criticality of an edge in a cutset.

V. Clustering Based Pruning and Ordering

Definition V.1 (**Dominant and Non-dominant Edges**). An edge e_i in set Σ is said to be **dominant** iff its local criticality with respect to all other edges in Σ is above a threshold ε, i.e.,

$$\tau_{ij} \quad > \quad \varepsilon \quad \{\ \forall\ e_j\ \in\ \Sigma,\quad e_j \neq e_i\ \} \tag{13}$$

Otherwise, e_i is said to be **non-dominant** in Σ, i.e.,

$$\{\ \exists\ e_j\ \in\ \Sigma,\quad e_j \neq e_i\ \}\quad :\quad \tau_{ij}\quad \leq\quad \varepsilon \tag{14}$$

Definition V.2 (**Mutually-dominant Edges**). A set of edges Σ are said to be **mutually dominant** iff each edge in Σ is dominant, i.e.,

$$\tau_{ij} \quad > \quad \varepsilon \quad \{\ \forall\ e_i,\ e_j\ \in\ \Sigma,\quad e_j \neq e_i\ \} \tag{15}$$

As seen in the previous section, non-dominant edges (like b in Fig. 2) in a cutset exhibit local errors. Moreover, they also contribute to global errors of other edges in the cutset (like a in Fig. 2). To avoid the bulk of these errors we propose to prune the cutset, eliminating its non-dominant edges from competing and injecting errors in global criticality computations.

Pruning is justified by Eq. 4, wherein eliminating edge e_i with local criticality lower than a sufficiently small threshold value ε does not hurt global criticality computations because $T_i \leq \varepsilon$. The benefits are accentuated in cutsets with dominant edges with large global criticalities, since the sum of global criticalities across a cutset equals 1.0 (implying that many edges have very small local criticalities).

It must be pointed out, that not every edge with global criticality below ε can be eliminated by pruning because its local criticality might not be smaller than ε. Such edges cause global errors using the linear MAX_θ operation.

A. *nC2 Cutset Pruning*

A straightforward approach to pruning a cutset would be to perform pairwise comparisons of edges, eliminating those that have a minimal local criticality less than a predefined threshold ε. The main drawback of this approach is its quadratic time complexity of $O(n^2)$, due to $nC2$ local criticality computations, where n is the number of edges in the cutset. This can be prohibitive for large circuits.

B. *Clustering Based Cutset Pruning and Ordering*

To overcome the quadratic runtime complexity overhead of the nC2 approach we present a new clustering-based pruning heuristic based on the K-center clustering algorithm of [4]. Algorithm 3 describes the procedure. The basic idea is to prune a cutset into a set of mutually dominant edges by removing its non-dominant edges. Variables used in the algorithm are,

κ : Each cluster σ contains a center κ
$d_{i\kappa}$: Distance of an object i from its cluster center κ is its local criticality $\tau_{i\kappa}$
r_σ : Radius of cluster σ is distance of the object farthest from κ, i.e., $r_\sigma = MAX(d_{i\kappa})\ \forall i \in \sigma$
R_σ : Border R of cluster σ is object with the maximum distance from κ, i.e., $R_\sigma = j\ :\ d_{j\kappa} = r_\sigma$

We first prune cutset Σ with respect to seed χ, chosen as the object with maximum mean in Σ. Steps 5-8 of Algorithm 3 compute new clusters from existing ones (Algorithm 4) until no cluster has size exceeding S. Finally, cutset Σ is replaced by the remaining un-pruned objects.

Algorithm 3 K = KCenterPrune (Σ, ε, S) // Σ = cutset; ε = pruning threshold; S = max. cluster size; K = # clusters

```
1: $\Omega = NULL$; $\sigma = NULL$; $K = 0$ {$\Omega$ is the set of clusters, $\sigma$ is the 1st cluster}
2: Assign seed $\chi$ = object with max. $\mu \in \Sigma$ as the center of $\sigma$
3: Prune $\Sigma$ with respect to $\chi$; mark $\chi$ = pruned if $\exists\ j \in \Sigma\ :\ (1 - \tau_{j\chi}) \leq \varepsilon$
4: Compute $r_\sigma$ and $R_\sigma$; $K = K + 1$; $\Omega(K) = \sigma$ {insert cluster $\sigma$ into $\Omega$}
5: while (max. size of a cluster in $\Omega > S$) do
6:     $\delta$ = CreateNewCluster ($\Omega$)
7:     $K = K + 1$; $\Sigma(K) = \delta$ {insert new cluster $\delta$ into $\Omega$}
8: end while
9: insert all un-pruned objects of $\Omega$ in $\Sigma$ and return $K$
```

Intuitively, χ in Algorithm 4 is the object upon which its center has the lowest degree of domination (Def II.6) and hence a good candidate to facilitate pruning of other edges in the cutset. Step 3 uses χ to prune objects j (with local criticality with respect to χ less than ε) from their respective clusters. Alternatively, if χ has a higher degree of domination over j compared to its current center κ, j is removed from its cluster and inserted into new cluster δ (Step 3). Intuitively, a greater degree of domination between two edges results in smaller global errors in MAX_θ. We return δ after adjusting it radius and border and those of all currently existing clusters in Ω (Step 4). The algorithm has the following properties.

Algorithm 4 δ = CreateNewCluster (Ω) // Ω = set of clusters; δ = new cluster

```
1: Assign $\chi$ = border of cluster with max. radius as center $\kappa$ of new cluster $\delta$
2: Prune $\Omega$ with respect to $\chi$; mark $\chi$ = pruned if $\exists\ j \in \Omega\ :\ (1 - \tau_{j\chi}) \leq \varepsilon$
3: if $\exists\ j \in \Omega\ :\ \tau_{j\chi} < d_{j\kappa}$, remove $j$ from its current cluster and insert into $\delta$
4: compute radius and border for $\delta$ and all existing clusters in $\Omega$; return $\delta$
```

Property V.1. At any iteration, all objects in Ω (excluding cluster centers marked pruned) are dominant with respect to all existing cluster centers.

Proof: To avoid pruning in Line 3 of Algorithm 3, objects must be dominant with respect to the seed (1st cluster center). Moreover, every object j is compared with every newly added cluster center (over the entire run of the algorithm, all cluster centers) in Line 2 of Algorithm 4. Clearly, j must be dominant with respect to these centers to avoid being pruned. Moreover, Line 3 of Algorithm 3 and Line 2 of Algorithm 4 compare every cluster center with every object for dominance. Although not removed from Ω, centers are marked pruned if they are non-dominant with respect to other cluster objects. ■

Property V.2. With $S = 1$, KCenterPrune(Σ, ε, 1) returns a set of mutually dominant edges (see Def.V.2) in Σ.

Proof: With $S = 1$ each cluster in Ω contains only one object, its cluster center. From Property V.1 above we know that that these are either marked pruned or are dominant with respect to other cluster centers. It follows from step 9 of Algorithm 3 that Σ contains mutually dominant objects. ■

Property V.3. For any cluster $\sigma \in \Omega$, its center χ has a **higher degree of domination** over its members than any other cluster center κ, i.e.,

$$\tau_{\chi j} \quad > \quad \tau_{\kappa j} \quad \{\ \forall\ j\ \in\ \sigma,\ \kappa\ \in\ \Sigma,\quad \kappa \neq \chi\} \tag{16}$$

TABLE II

CRITICALITY RUN-TIMES AND ERRORS FOR VARIOUS BENCHMARKS; $\varepsilon = 5\%$ AND $N_{ls} = 1000$

Metric	Pruning Scheme	Benchmark								
		s27	s1196	s5378	s9234	s13207	s15850	s38417	s35932	s38584
max. $\delta\%$	ZSC	**3.84**	**2.25**	**47.99**	**45.79**	**38.11**	**58.49**	**40.82**	**24.99**	**43.56**
	nC2	3.84	2.99	3.13	21.97	2.65	6.43	37.82	16.68	18.65
	CPSC	3.84	2.99	3.13	23.65	5.30	5.49	37.82	16.12	18.65
	CPSC+TGR+LS	**3.84**	**2.99**	**3.13**	**2.92**	**5.21**	**2.37**	**2.17**	**2.79**	**15.82**
run time (sec)	ZSC	**0.00**	**0.03**	**0.16**	**0.30**	**0.45**	**0.54**	**2.51**	**2.71**	**3.06**
	nC2	0.00	0.02	0.39	0.71	2.31	2.19	43.53	55.39	51.76
	CPSC	0.00	0.01	0.02	0.05	0.07	0.09	0.27	0.34	0.25
	CPSC+TGR+LS	**0.00**	**0.00**	**0.03**	**0.08**	**0.09**	**0.11**	**0.34**	**0.37**	**0.34**
η	ZSC	7	202	593	644	1329	1688	2821	7340	5680
	nC2	**2**	**1**	**2**	**11**	**4**	**4**	**5**	**49**	**11**
	CPSC	2	1	2	11	4	4	5	49	11
	CPSC+TGR+LS	**2**	**1**	**2**	**11**	**4**	**4**	**5**	**49**	**11**

Proof: This is evident from Step 3 of Algorithm 4. Each object in Ω is compared with the new cluster center χ. The condition $\tau_{j\chi} < d_{j\kappa}$ is equivalent to $\tau_{\chi j} > \tau_{\kappa j}$, i.e., the new cluster center χ has a higher degree of domination over object j than its current cluster center κ. ∎

Property V.4. For a cutset Σ of size n and K clusters returned, KCenter-Prune takes $O(nK)$ time.

Proof: A single run of Algorithm 4 compares every object in Σ with center χ of the new cluster δ, taking $O(n)$ time. Since each iteration in Algorithm 3 returns a new cluster, with K clusters returned, the overall runtime is $O(nK)$. ∎

C. *CPSC: Clustering Based SC Algorithm*

Using Algorithm 3 for SC computation is straightforward. Step 2 in Algorithm 5 derives a set of mutually dominant edges from cutset Σ, facilitated using Property V.2 and orders Σ, facilitated by Property V.3. Such an ordering cannot be obtained with the nC2 pruning strategy of Section V-A. Property V.4 ensures that with a small number of dominant edges K in a cutset, the algorithm runs in linear time.

D. *Localized Sampling and Timing Graph Reduction*

To tackle edges that lead to global errors (Def. IV.2), we perform a quick localized Monte Carlo sampling of the edges in cutset Σ after pruning. We generate N_{ls} samples of the k independent identically distributed Gaussians, k being the number of PCs (Def. II.4). Each sample is used in instantiating e_i ($\forall e_i \in \Sigma$), from which we compute the edge in Σ with the maximum delay. By keeping count of the number of samples for which an edge in Σ has maximum delay, we can compute its global criticality probability.

Algorithm 5 CPSC $(G(V,E), \varepsilon)$

```
1: Algorithm 2 Steps 1-7
2: K = KCenterPrune ($\Sigma, \varepsilon$, 1); KCenterPrune ($\Sigma, \varepsilon, S$)
3: Algorithm 2 Step 8
```

The runtime of LS increases with the number of samples N_{ls} and the depth L of the timing graph G. Reducing N_{ls} reduces the accuracy of criticality computation but L has no effect on the accuracy. Moreover, the criticality of a node in G is equivalent to the sum of its fanin edge or fanout edge criticalities. Exploiting this property, we perform a timing graph reduction (TGR) procedure on nodes with a single fanin or fanout. A straightforward and practical example of this reduction is an inverter chain

Table III shows the effect of TGR on the depth (number of levels) L and maximum cutset size η on the three largest benchmark circuits. Column 2 shows the size of the circuit. As their names imply, columns w/ red. and w/o red. are results with and without TGR respectively, applied to G.

TABLE III

EFFECT OF TGR ON CIRCUIT DEPTH L AND MAXIMUM CUTSET SIZE η

Benchmark	# of Gates	L		η	
		w/o red.	w/ red	w/o red.	w/ red
s38417	22179	51	13	2821	6638
s35932	16065	33	10	5473	10742
s38584	19253	60	19	5680	10374

VI. RESULTS

Our algorithms were implemented in C++ on top of an SSTA engine and exercised on the ISCAS89 benchmarks, with parameter values corresponding to the 100nm technology node [7]. Experiments were conducted on a Linux PC with a 3.0-GHz CPU and 2GB RAM. We compared four schemes with Monte Carlo simulations using 10000 samples, shown in Table II.

The first scheme is the ZSC approach in Algorithm 2. Scheme nC2 implements the pairwise pruning strategy of Section V-A with a pruning threshold ε of 5%. CPSC implements Algorithm 5 using our clustered pruning and ordering technique. Finally, CPSC+LS+TGR performs clustered pruning on the reduced timing graph (TGR) and computes criticalities using the LS procedure with $N_{ls} = 1000$ samples. Row max. δ reports the maximum difference between the edge criticality computed using any of the above mentioned schemes and the Monte Carlo simulations, run-time reports the running time in seconds and η reports the maximum number of edges in any cutset of the timing graph after pruning . We exclude the times for SSTA and generating samples in the LS scheme.

From Table II, ZSC which computes criticalities using Clark's MAX_θ formulation results in large errors (the largest being 58%). Since ZSC fundamentally does not differ from [10], we expect to see errors of similar magnitude. CPSC with cutset pruning and ordering does better than ZSC in accuracy and runtime. For circuits exhibiting large global errors, ordering is insufficient to reduce these errors. Rows in bold compare ZSC with CPSC+TGR+LCS. The combined approach greatly reduces errors with negligible increase in runtime compared to CPSC. For the 3 large benchmarks we obtain about an order of magnitude difference run-times of ZSC and the combined approach. Most circuits have errors below 5%, except for s38584. On investigation, it was found that for large fanout structures, path delays themselves (computed in terms of the PCs) contained large errors and hence the LS procedure does not completely eliminate global errors. In terms of the efficacy of our pruning strategy, as expected we vastly outperform the nC2 procedure in run-time (about two orders of magnitude for the larger benchmarks). Moreover, each circuit contained an identical number of edges remaining in the cutsets using the nC2 and CPSC pruning strategies.

REFERENCES

[1] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Computer Society, 2003, p. 621.

[2] 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," in *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Computer Society, 2005, pp. 1023–1028.

[3] C. E. Clark, "The greatest of a finite set of random variables," *Operations Research*, vol. 9, no. 2, pp. 145–162, Mar-Apr 1961.

[4] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.

[5] 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," in *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Computer Society, 2005, pp. 844–851.

[6] S. R. Nassif, "Design for variability in DSM technologies," in *Proceedings of the 1st International Symposium on Quality of Electronic Design*. IEEE Computer Society, 2000, p. 451.

[7] Predictive technology model (PTM). [Online]. Available: http://www.eas.asu.edu/~ptm/

[8] D. Sinha, H. Zhou, and N. V. Shenoy, "Advances in computation of the maximum of a set of random variables," in *Proceedings of the 7th International Symposium on Quality Electronic Design*, Mar. 27–29, 2006.

[9] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proceedings of the 41st Annual Design Automation Conference*. ACM Press, 2004, pp. 331–336.

[10] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, "Criticality computation in parameterized statistical timing," in *Proceedings of the 43rd Design Automation Conference*. ACM Press, 2006, pp. 63–68.

[11] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 1, pp. 25–35, Jan. 1982.