

A Novel Criticality Computation Method in Statistical Timing Analysis

Feng Wang, Yuan Xie
The Pennsylvania State University,
University Park, PA, USA
{fenwang, yuanxie}@cse.psu.edu

Hai Ju
IBM China Research Laboratory,
China
juhai@cn.ibm.com

Abstract—The impact of process variations increases as technology scales to nanometer region. Under large process variations, the path and arc/node criticality [18] provide effective metrics in guiding circuit optimization. To facilitate the criticality computation considering the correlation, we define the critical region for the path and arc/node in a timing graph, and propose an efficient method to compute the criticality for paths and arcs/nodes simultaneously by a single breadth-first graph traversal during the backward propagation. Instead of choosing a set of paths for analysis prematurely, we develop a new property of the path criticality to prune those paths with low criticality at very earlier stages, so that our path criticality computation method has linear complexity with respect of the timing edges in a timing graph. To improve the computation accuracy, cutset and path criticality properties are exploited to calibrate the computation results. The experimental results on ISCAS benchmark circuits show that our criticality computation method can achieve high accuracy with fast speed.

I. INTRODUCTION

As technology scales to nanometer region, it becomes a great challenge to control the process variation [4]. The variations in the device and interconnect parameters continuously increase as the feature size approaches to the optical resolution limit [13] [3]. The increasing fluctuations in manufacturing process parameters make it extremely difficult for designers to predict the circuit performance and verify the timing. Traditional corner based static timing analysis becomes too pessimistic if worst case timing analysis is performed. Furthermore, analyzing all possible process corners is computationally prohibitive since the computation complexity increases dramatically with the increase of the number of process corners. Many researches have been done in statistical timing analysis to address these problems [18] [5] [15] [1] [2] [12] [8] [17] [14] [11] [9] [7].

Statistical timing analysis approaches can be categorized into two major types: path-based approaches and block-based approaches [18]. Path-based approaches perform timing analysis with a depth first traversal of a timing graph, by choosing a set of paths with high probability being critical for analysis. The correlation due to the path sharing and global sources of variation can be easily captured using path-based approaches [10] [18]. However, the set of the paths selected for statistical timing analysis is hard to determine. Existing approaches choose the set of paths based on the results of static timing analysis, which might miss some important paths. In comparison, the block-based approaches perform timing analysis by traversing the timing graph in a

levelized "breadth-first" manner [18]. In such approaches, the parameterized delay model approximates the timing quantities as normal distributions. With this approximation, the block-based approaches can effectively handle the correlation due to the path sharing and the variation of global parameters [10] [18].

After the statistical timing analysis, the statistical delay information of the circuits is available. The slack is not a deterministic value and many paths could be critical in some process subspaces. Thus, slack and critical path metrics used in the deterministic static timing analysis become less effective in guiding the circuit optimization. Visweswariah et al. [18] first defined the concepts of the *criticality* of the path and the arc/node. The criticality of the path is defined as the probability of the path being critical. The criticality of the node/arc is defined as the probability of the node/arc on the critical path.

Visweswariah et al. proposed a very simple fast method to compute the criticality for the path and the node/arc [18]. However, this approach is valid under the assumption of independence among the paths, which is not true due to the path convergence and the variation of global parameters [10] [20] [19]. Li et al. [10] and Zhan et al. [20] first pointed out the limitations of that approach. Li et al. [10] proposed a sensitivity based criticality computation method for arcs in the timing graph. Path sensitivity and arc sensitivity are defined as the mean value of the maximal circuit delay over the individual path delay and arc delay, respectively. The criticality of node/arc is computed based on sensitivity propagation using chain rule through a timing graph. Xiong et al. [19] computed the criticality as the probability of the edge slack being larger than its complement edge slack. Binary partition tree was used to speed up the criticality computation. Zhan et al. [20] proposed two techniques to compute the path criticality based on *max* operations and Monte Carlo integration. In his method, paths are chosen from a preselected path set for criticality computation. However, similar to the path-based statistical timing analysis, it is not clear how to determine the preselected path set. If the paths are chosen based on the static timing analysis, important paths could be missed.

In this paper, we propose a new approach to compute the *path* and *arc/node* criticality *simultaneously* by a single breadth-first graph traversal based on the concept of the *critical region* (will be defined in Section III). Instead of selecting the paths using static timing information as [20], we develop

a new *path criticality property* to prune the paths with low criticality at very earlier stages. Pruning paths based on the statistical information significantly reduces the computation costs without sacrificing the accuracy. Existing *cutset and path criticality properties* are exploited to improve the accuracy of the criticality computation.

The rest of the paper is organized as follows: Section II reviews the background for the statistical timing analysis method and introduces the concept of criticality for the node/arc and the path; Section III presents a concept of critical region and the criticality computation methods for the path and the arc/node; Section IV shows experimental results on the ISCAS benchmark circuits. Finally, the conclusions are provided in Section V.

II. BACKGROUND

In this section, we first give a brief introduction on the first order parameterized statistical timing analysis (SSTA). We then introduce the concept of statistical criticality. Finally we present the methods to determine the criticality of the node/arc without the simple independence assumption.

A. Canonical Model

In first order SSTA, the timing quantity (such as delay, arrival/required time and slew) is approximated as a linear function of random variables. In Visweswariah's paper [18], the first order approximation of the timing quantity is expressed as a canonical form:

$$a_0 + \sum_{i=1}^n a_i \Delta Y_i + a_{n+1} \Delta R_a \quad (1)$$

where a_0 is the nominal delay computed at the nominal values of the process parameters. ΔY_i represents the correlated variation, and ΔR_a represents the independent random variation. ΔY_i and R_a are independent and normally distributed random variables with zero mean and unit variance. a_i and a_{n+1} are the sensitivities to their corresponding sources of the variation.

B. Timing Graph

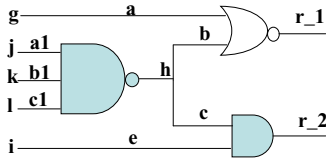


Fig. 1. Circuit diagram

In statistical timing analysis, the circuit is modeled as a timing graph $G=(V,E)$. Fig. 1 and Fig. 2 depict a circuit diagram and the corresponding timing graph. The node, $v_i \in V$, in the timing graph represent the gate, primary input, primary output or the interconnects. The edge, $e(i,j) = (v_i, v_j) \in E$, represents the arc between the node v_i and v_j . The weight associated with the arc $e(i,j)$ in a graph represents the delay of that arc. Virtual source node and virtual sink

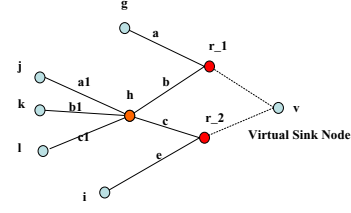


Fig. 2. The circuit timing graph.

node are connected to the primary inputs and primary outputs, respectively.

C. Atomic Operations

In statistical timing analysis, the distribution of the timing quantity is computed using two atomic functions *max* and *sum*. Assume that we have three timing quantities, A , B , and C in canonical forms (equation 2-4). The result of sum operation $sum(A,B)$ or max operation $max(A,B)$ for the timing quantities is denoted as C .

$$A = a_0 + \sum_{i=1}^n a_i \Delta Y_i + a_{n+1} \Delta R_a \quad (2)$$

$$B = b_0 + \sum_{i=1}^n b_i \Delta Y_i + b_{n+1} \Delta R_b \quad (3)$$

$$C = c_0 + \sum_{i=1}^n c_i \Delta Y_i + c_{n+1} \Delta R_c \quad (4)$$

- *Sum operation.*

The sum operation is simple. $C = sum(A,B)$ can be computed as $c_i = a_i + b_i$, where $i \in [0, n]$ and $c_{n+1} = \sqrt{a_{n+1}^2 + b_{n+1}^2}$.

- *Max operation.*

The *max* operation is quite complex. The maximum C of A and B is approximated as a normally distributed random variable with the same canonical form as shown in equation (4). Tightness probability and moment matching techniques [6] can be used to determine the corresponding sensitivities to the process parameters. Tightness probability is defined as the probability of a random variable being larger than another. Clark's paper [6] provide an analytical equation to compute the tightness probability.

We first give the definition of the terms, which are used in the calculation of the $max(A,B)$.

$$\phi(x) = \exp\left(-\frac{x^2}{2}\right) \quad (5)$$

$$\Phi(x) = \int_{-\infty}^x \exp\left(-\frac{x^2}{2}\right) \quad (6)$$

$$\theta = (\sigma_a^2 + \sigma_b^2 - 2\rho\sigma_a\sigma_b) \quad (7)$$

$$\alpha = \frac{a_0 - b_0}{\theta} \quad (8)$$

$$v_2 = (a_0^2 + \sigma_a^2)\Phi(\alpha) + (b_0^2 + \sigma_b^2)(1 - \Phi(\alpha)) + \theta\phi(\alpha) \quad (9)$$

We then show the calculation of the sensitivity values of C . c_0 is used to match the first order moment.

$$c_0 = a_0\Phi(\alpha) + b_0(1 - \Phi(\alpha)) + \theta\phi(\alpha) \quad (10)$$

The sensitivity to the correlated random variables can be obtained as a weighted sum of the a_i and b_i regarding to the tightness probability.

$$c_i = a_i\Phi(\alpha) + b_i(1 - \Phi(\alpha)) \quad \forall i \in 1, 2, \dots, n \quad (11)$$

Finally, the sensitivity to the independent random variable is calculated to match the variance, v_2 , which is given by Clark [6].

$$c_{n+1} = (v_2 - \sum_{i=0}^n (c_i^2))^{\frac{1}{2}} \quad (12)$$

With the atomic operations defined, the timing quantities of the arrival time and the required time can be computed in *forward* and *backward* operations respectively. Our criticality computation is based on the results of the arrival time in SSTA, but it is not restricted to the first order SSTA. Note that our criticality computation method is *NOT* limited to first order SSTA, and using a *higher order* SSTA can improve its accuracy.

D. Concept of the Criticality

Visweswariah [18] proposed the following concepts of the criticality of the path and the criticality of arc/node.

- The criticality of the path is defined as the probability of the path being critical.
- The criticality of the node/arc is defined as the probability of the node/arc on the critical path.

These concepts can be used to guide the designer to identify the critical arc/node to perform the optimization. The gates with high criticality on the critical path can be sized up to improve the performance, while the arc/node with lower criticality can be sized down to save the power. In performance optimization, it is also very important for the circuit designer to identify the most important paths, which have highest probability being critical [20].

E. Criticality of Arc/Node

In this section, we show the arc/node criticality computation method, which is *NOT* restricted to the *independence assumptions* [18]. In Section III, we will show the criticality computation for paths, which is based on the critical region computation for the node/arc.

For arc (h, r_i) in a timing graph, its criticality is expressed as:

$$Prob((AT(h) + d(h, r_i) > \max_{g \neq h}(AT(g) + d(g, r_i))) | r_i \text{ critical}) \times Prob(r_i \text{ critical}) \quad (13)$$

where:

- g represent all the other fan-in nodes of node r_i ;

- $d(h, r_i)$ is the delay of the arc from node h to node r_i ;
- r_i ($i \in [1, m]$) represents the set of the fan-out nodes of node h ;
- $Prob(r_i \text{ critical})$ represents the probability of r_i being critical.

According to the lemma in [18], for node h in a timing graph, the criticality of node h can be calculated as the summation of the criticality of arcs originating from node h .

$$\sum_{i=1}^m \{Prob(arc(h, r_i) \text{ is critical})\} \quad (14)$$

III. CRITICALITY COMPUTATION METHOD

In the previous section, the criticality is expressed as the form of $P(A|B)P(B)$. According to probabilities theory [16], $P(A|B)P(B)$ is equal to $P(A \cap B)$. With the help of Venn diagrams, $P(A \cap B)$ can be represented as the intersection of the two sets A and B [16]. To facilitate the criticality computation considering the correlations, we introduce the concept of the *critical region* for nodes/arcs and paths. We define the critical region of a node/arc as the process subspace where the node/arc is on the critical path. The critical region of a path is defined as the process subspace where the path becomes critical. The critical region is computed when we perform the backward operation on a timing graph. *With the critical regions determined, the criticality can be calculated using tightness function [18] and the equations in Section II.* We classify the nodes in a timing graph into two types: 1) the nodes with a single fan-out 2) the nodes with multiple fan-outs. We first show the critical region computation for these two types of nodes and their corresponding arcs. We then introduce a Lemma to determine the critical regions of paths in a timing graph and presents a path criticality property. Finally, we shows our criticality computation method based on the concept of the critical region.

A. Critical Region Computation for Arc/Node

From Section II, the criticality of the arc can be expressed as equation (13). Then, the critical region of the arc is the intersection of the critical region of its fan-out node and the region where the arrival time (AT) of that arc determines that of the fan-out node. So the critical region of arc (h, r_i) is the intersection of the following two regions:

- The region where the arrival time of the arc (h, r_i) determines the arrival time of fan-out node r_i , i.e., $(AT(h) + d(h, r_i) > \max_{g \neq h}(AT(g) + d(g, r_i)))$. We rewrite this condition as $(AT(h) + d(h, r_i) - \max_{g \neq h}(AT(g) + d(g, r_i))) > 0$ and it is denoted as $F(h, r_i) > 0$
- The region where fan-out node r_i is critical, and it is denoted as $F(r_i) > 0$

So the critical region for arc (h, r_i) can be expressed as $\min(F(r_i), F(h, r_i)) > 0$. With the critical region available, we use the *tightness function* [18] to compute the criticality of arc (h, r_i) as the probability of $\min(F(r_i), F(h, r_i)) > 0$.

From equation (14), the criticality of the node can be expressed as the summation of those of its fan-out arcs. Thus,

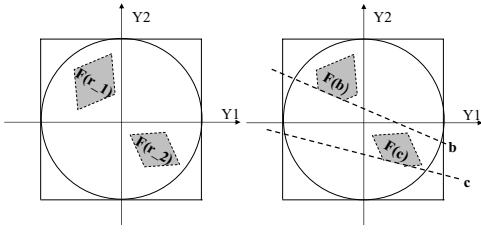


Fig. 3. The critical regions.

the combination of the critical regions of the arcs originating from that node is the critical region of the node. For example, in Fig. 2, assuming the critical regions for the fan-out nodes r_1 and r_2 are known, we show how to determine the critical region for node h . In Fig. 3, the dark areas, $F(r_1)$ and $F(r_2)$, represent the critical region of node r_1 and node r_2 respectively. Assuming that the process subspace where arc (h, r_1) determines the delay of the node r_1 is denoted as the region above dashed line b , the intersection of these two regions is the critical region of arc (h, r_1) . We denote this intersection as $F(b)$. Similarly we obtain the critical region $F(c)$ for arc (h, r_2) . For node h , its critical region is simply the combination of these two regions, $F(b)$ and $F(c)$.

Thus, for the internal node with multiple fan-outs, we first compute the critical regions of the arcs from that node to its corresponding fan-outs individually and then calculate the criticality over those regions for all its fan-out arcs. We then compute the criticality of the node by simply summing up the criticality of each arc originating from that node. For the internal node with a single fan-out, its the *critical region/criticality* is the same as that of the arc originating from the node to its fan-out.

B. Critical Region Computation for Path

To compute the path criticality, we first identify the critical regions of the paths, and Lemma 1 helps determine these critical regions. Then, we show that the critical regions of the paths are computed after a breadth first traversal of a leveled timing graph. Finally we develop a new path criticality property to prune the paths with low criticality to improve the efficiency of path criticality computation.

Lemma 1: A path's critical region is the intersection of all the critical regions of the arcs along the path.

Proof: We prove the statement with *proof by contradiction*. We can assume that 1) there exists at least one subspace in the process space other than the intersection where the path is critical or 2) there exists at least one subspace within the intersection where the path is *NOT* critical. An arc is critical if the arc is on a critical path. All arcs along the path are critical when that path is critical, so that any subspace where a path is critical is part of the intersection of critical regions of all arcs along that path. Thus it contradicts the assumption 1). Since all the arcs on the path are critical, the slack of the path is equal to the slack of the arc. Thus the path has minimum slack, which means the path must be critical. It contradicts the assumption 2). Therefore, the path's critical region is simply

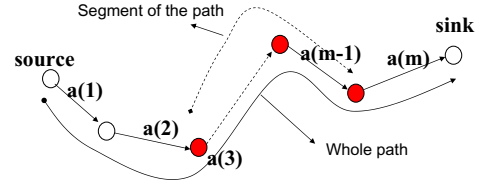


Fig. 4. The example of the segment of the path in a timing graph.

the intersection of the critical regions of the arcs along that path.

To compute the critical regions of the paths, we perform the critical region analysis for nodes/arcs in a breadth first traversal of a leveled timing graph. After this BFS traversal, the critical region of each PI contains the combinations of the critical regions of all the paths originating from the PI in the timing graph.

We show the correctness of the above statement with a stronger statement as the follows. We perform a breadth-first search for a leveled timing graph and compute the critical region for node/arc from the highest level, m , to level 1, which consists of all the *PIs*. The nodes in this graph are leveled according to its distance to the primary inputs. We prove that: *at each level, the critical region of the node is the combination of all the critical regions of the paths originating from that node to the primary outputs*. We prove this statement with *proof by induction*.

Proof: Initially, at level m , the critical regions of the arcs from the primary outputs to the virtual sink node is computed. The statement is certainly true for level m .

Assume that the above statement is true for all level k , $n \leq k \leq m$. We prove that the statement is also true for level $n-1$. We classify the nodes into two types: 1) the nodes without any fan-outs 2) the nodes with one fan-out or multiple fan-outs. For the first type of the node, similar to the node at level m , we compute the criticality for the arc from the fan-out to the virtual sink node. Thus the statement is true. For the node with one or more fan-outs, since we assume that the statement is true for level k , where $n < k < m$, any fan-out of the node at level $n-1$ contains the critical regions of all the paths from that fan-out node to the primary outputs. We denote these regions as the set c_i , where $1 \leq i \leq p$, p is number of the path from the fan-out to the primary outputs. As shown in Section III-A, the critical region of the node's fan-out edge is computed as the intersection of the critical region of its fan-out node and the region (denoted c_{arc}) where the AT of that arc determines the AT of the fan-out node. Effectively, the critical region of that fan-out edge is the combination of the intersections of c_i and c_{arc} for all the i , where $1 \leq i \leq p$. From Lemma 1, the intersection of c_i and c_{arc} is the critical region of the path from the node to the virtual sink node via its fan-out, because the critical region of the node is the combination of the critical regions of the arcs along the path. Any path originating from the node is simply the path, from its fan-out to the virtual sink node, concatenated with the arc from the

node to its fan-out. Thus, at level $n - 1$, the statement is true.

In conclusion, since the base case is true and the inductive step is true, the statement is true for all the levels. Therefore, the critical region of a *PI* is the combination of the critical regions of all the paths originating from that *PI* to the virtual sink node. Since any path in the timing graph is from the *PI* to the virtual sink node, its critical region is determined when we perform criticality computation in a BFS manner. In addition, no path has been computed twice, since there is no single path going through any two fan-out arcs of that node.

To speed up the path criticality computation, we develop a new path criticality property. Assume that a path consists of a set of arcs: a_i , where i is from 1 to m , the segment of the path can be defined as a set of arcs: a_j , where j is from k to p , and $1 \leq k < p \leq m$. Fig. 4 shows an example of a segment of the path. With this definition, we have **property 1** as follows.

Property 1: The criticality of the path is not larger than the criticality of any segment of that path.

Proof: From lemma 1, the criticality of the path is equal to $Prob((a_1 \text{ critical}) \cap (a_2 \text{ critical}) \cap \dots \cap (a_m \text{ critical}))$ and the criticality of the segment of the path criticality is calculated as $Prob((a_k \text{ critical}) \cap (a_{k+1} \text{ critical}) \cap \dots \cap (a_p \text{ critical}))$. So the critical region of the path is the subspace of that of the segment. From the probabilities theory [16], the statement is true.

C. Computation Algorithm

In this section we first show the algorithm to compute the criticality of the paths and nodes in the timing graph. We then show a heuristic to improve the speed of the criticality computation. Finally we present a heuristic to improve the accuracy of the computation.

```
Criticality (netlist){
1. Compute the critical regions of the primary outputs (POs);
2. Compute the critical regions of the nodes/arcs;
3. Prune the nodes/arc with the criticality less than the criticality threshold;
4. Repeat 2 and 3 until the primary inputs are visited;
5. Compute the path criticality at the primary inputs;
6. }
```

Fig. 5. The pseudo code of the criticality computation

Fig. 5 shows the criticality computation algorithm. It takes the gate net-list as its input and compute the criticality for the arc/node and path simultaneously. The criticality computation involves a BFS traversal from the POs to the PIs. The critical regions of the nodes/arcs are determined from the critical regions of its fan-out arcs and nodes in the BFS traversal. After the traversal reaches the PIs, each PI contains the critical regions of all the paths starting from that PI. The path criticality is then computed over its critical region.

A brute-force path criticality computation approach leads to large computational overheads (its computation complexity is linear with respect to the number of the paths in a timing graph). To speed up the computation, we use a heuristic to improve the performance of our algorithm. **Property 1 enables us to prune the path/node/arc with a small criticality value at**

very earlier stages of path criticality computation. Since a large portion of the paths in a timing graph has low criticality value, the computational complexity can be greatly reduced with the heuristic. As shown in Table II in Section IV, our path criticality computation method has a *linear computational complexity with respect of the timing edges*. Although the reduction of the computation cost depends on the designs, the experimental results demonstrate the effectiveness of our path pruning technique across the ISCAS benchmark circuits. In addition, we avoid the path selection problem in Zhan's approach [20]. We use the statistical timing information, instead of static timing information, to remove paths that are not important to the circuit designer.

The linear approximation of the gaussian distribution in *max* or *min* operation is a major source of the computation error in the criticality computation [18]. As the critical region computation proceeds to the level close to the primary inputs, the error due to the approximation accumulates. However, we extend the properties developed by Visweswariah et al. [18] and integrate them to calibrate the results.

Property 2: The sum of the criticality of the unpruned paths in a timing graph is 1.0-the sum of the criticality of the pruned paths.

Property 3: The sum of the unpruned edge criticality of any cutset in a timing graph that separates the source from the sink node is 1.0-the sum of the criticality of the pruned edges of that cutset.

From the properties in [18], the sum of the criticality of all the paths in a timing graph has to be 1.0. We record the sum of the criticality of the pruned paths as *prunedpath*. The sum of the unpruned path criticality denoted as *unprunedpath* can be computed after the BFS traversal. We normalize the criticality value of each path by multiplying a scaling factor $\frac{1}{pathtotal}$, where $pathtotal = prunedpath + unprunedpath$. Similarly, we compute the sum of the arc criticality of any cutset as *cutsettotal*, we normalize the criticality value of each arc belonging to that cutset with a factor $\frac{1}{cutsettotal}$.

IV. ANALYSIS RESULTS

In this section, we present the analysis results and show our method can accurately compute the criticality with fast speed.

We implement our criticality computation method in C++ and integrate it into our statistical timing analysis tools. We conduct the criticality analysis on ISCAS 85 benchmark circuits to show the efficiency and accuracy of our method.

To demonstrate the accuracy of our method, we compare the simulation results against Monte Carlo simulation with 10,000 samples. We perform the statistical timing analysis and collect the statistical information of the critical path/node/arc for each sample. Table I shows the results of our method against the Monte Carlo techniques. In the second and third columns, we show the results of maximal and average criticality errors of the arcs with our methods. We also show the results of Li et al.'s [10] method in the fourth and fifth columns. In our method, the maximal and average criticality errors for arcs are less than 1.17% and 0.05% respectively. The maximal error

TABLE I
ACCURACY OF OUR CRITICALITY COMPUTATION METHODS

Circuit	critical node/arc		critical node/arc [10]		critical path
	max	avg	max	avg	max
c880	0.12%	0.010 %	1.3%	0.9 %	0.010 %
c1908	0.31%	0.006 %	3.4%	0.4 %	0.260 %
c2670	1.17%	0.043 %	2.6%	0.3 %	0.820 %
c5315	0.44%	0.016%	2.8%	1.8 %	0.340%
c6288	0.19%	0.047%	1.9%	0.6 %	0.100 %
c7552	0.43%	0.042 %	3.5%	1.1 %	0.400 %

TABLE II
RUN TIME OF OUR CRITICALITY COMPUTATION METHODS

Circuit	Size	SSTA (sec)	Criticality Computation (sec)	Overhead
c880	0.50k	0.0052	0.0041	0.78
c1908	0.60k	0.010	0.007	0.7
c2670	0.78k	0.013	0.012	0.99
c5315	1.7k	0.026	0.016	0.62
c6288	3.8k	0.049	0.027	0.55
c7552	2.2k	0.042	0.015	0.38

of the path criticality against Monte Carlo techniques is less than 0.82% as shown in sixth column.

Table II shows the run time of our method against the basic statistical timing analysis. The circuit size in terms of the number of gates is given in column two. The run time of basic statistical timing analysis, the run time of criticality computation, and the relative overhead of criticality computation over statistical timing analysis are reported in column three, four and five, respectively. From column five in Table II, we can see that the run time of the criticality computation for both paths and nodes/arcs is less than that of the corresponding SSTA. The run time overhead of our criticality computation over the basic statistical timing analysis tends to decrease as the size of the circuit increases. These results indicate that our path pruning technique reduces the computational complexity of path criticality computation to linear complexity with respect of the timing edges.

Compared to the previous work [10] [19] [20], our criticality computation method computes criticality for both paths and nodes/arcs. Our method has the same run time complexity as that of existing methods solely for the arc criticality computation [10] [19]. The results on the same benchmarks demonstrate that our method is more accurate in computing arc criticality compared to Li's approach [10]. In Zhan's approach [20], the path criticality computation is performed on a pre-selected set of paths, which might lead to missing some important paths. Our method avoid this problem by pruning the paths based on the statistical information.

V. CONCLUSIONS

In this paper, we define the critical region for paths and nodes/arcs in a timing graph. With this definition, we develop an efficient method to compute the criticality for paths and the arcs/nodes simultaneously. A new property of the path criticality is used to prune the low criticality node/arc at the very early stages of computation to avoid the selection of the paths based on the static timing information. Cutset and path criticality prosperities are used to improve the accuracy in the

criticality computation. Simulation results show our criticality computation method is very accurate and fast.

VI. ACKNOWLEDGEMENT

The authors would like to express their appreciation to Keith A. Bowman of Intel for his helpful advice. This work was supported in parts by grants from MARCO/DARPA-GSRC.

REFERENCES

- [1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 900–907, November 2003.
- [2] A. Agarwal, V. Zolotov, and D. Blaauw. Statistical timing analysis using bounds and selective enumeration. *IEEE Trans. CAD*, (9):1243–1260, September 2003.
- [3] V. Axelrad and J. Kibarian. Statistical aspects of modern IC designs. *Proceedings of the 28th European Solid-State Device Research Conference*, pages 309–321, September 1998.
- [4] K. A. Bowman, S. G. Duvall, and J. D. Meindl. Impact of Die-to-Die and Within Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration. *Journal of Solid-State Circuits*, pages 183–190, February 2002.
- [5] H. Chang and S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 621–625, November 2003.
- [6] C. Clark. The greatest of a finite set of random variables. *Operations Research*, pages 145–162, 1961.
- [7] Liang Deng and Martin D. F. Wong. An exact algorithm for the statistical shortest path problem. *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 965–970, 2006.
- [8] A. Devgan and C. Kashyap. Block-based static timing analysis with uncertainty. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 607–614, November 2003.
- [9] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. *IEEE/ACM DAC*, pages 932–937, 2003.
- [10] X. Li, J. Le, M. Celik, and L. Pileggi. Defining statistical sensitivity for timing optimization of logic circuits with largescale process and environmental variations. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 422–429, November 2005.
- [11] J. Liou, K. Chen, S. Kundu, and A. Krstic. Fast statistical timing analysis by probabilistic event propagation. *IEEE/ACM DAC*, pages 661–666, 2001.
- [12] F. N. Najm. On the need for statistical timing analysis. *IEEE/ACM DAC*, pages 764–765, 2005.
- [13] S. Nassif. Design for variability in DSM technologies. *IEEE International Symposium on Quality of Electronic Design*, pages 451–454, March 2000.
- [14] A. Gattiker, S. Nassif, R. Dinakar, and C. Long. Timing yield estimation from static timing analysis. *IEEE International Symposium on Quality Electronic Design*, pages 437–442, 2001.
- [15] M. Orshansky and K. Keutzer. A general probabilistic framework for worst case timing analysis. *IEEE/ACM DAC*, pages 556–561, 2002.
- [16] A. Papoulis and S. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 2001.
- [17] D. Sinha and H. Zhou. A Unified Framework for Statistical Timing Analysis with Coupling and Multiple Input Switching. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, November 2005.
- [18] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. *Design Automation Conference (DAC)*, pages 331–336, June 2004.
- [19] J. Xiong, V. Zolotov, C. Visweswariah, and N. Venkateswaran. Criticality Computation in Parameterized Statistical Timing. *ACM/IEEE international workshop on timing issues in the specification and synthesis of digital systems (TAU)*, pages 119–124, February 2006.
- [20] Y. Zhan, A. J. Strojwas, M. Sharma, and D. Newmark. Statistical Critical Path Analysis Considering Correlations. *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, November 2005.