Statistical Timing Verification for Transparently Latched Circuits through Structural Graph Traversal

Xingliang Yuan and Jia Wang

Electrical and Computer Engineering Department
Illinois Institute of Technology
Chicago, IL 60616

Abstract— Level-sensitive transparent latches are widely used in high-performance sequential circuit designs. Under process variations, the timing of a transparently latched circuit will adapt random delays at runtime due to time borrowing. The central problem to determine the timing yield is to compute the probability of the presence of a positive cycle in the latest latch timing graph. Existing algorithms are either optimistic since cycles are omitted or require iterations that cannot be polynomially bounded. In this paper, we present the first algorithm to compute such probability based on block-based statistical timing analysis that, first, covers *all* cycles through a structural graph traversal, and second, terminates within a *polynomial* number of statistical "sum" and "max" operations. Experimental results confirm that the proposed approach is effective and efficient.

I. Introduction

Aggressive scaling down of feature sizes enables the manufacturing of VLSI circuits with billions of transistors. As devices and interconnects approach nanoscale, process variations, though previously ignorable, have become a critical issue in VLSI fabrication that designers must fact at design time since they will affect manufacturing yield and run-time reliability of the circuits. Among the design characteristics affected by process variations, circuit timing is addressed by many research works of statistical static timing analysis (SSTA), as reviewed in [1]. More specifically, block-based SSTA algorithms, e.g. [2, 3], model device and interconnect delays as probabilistic distributions instead of deterministic values, extend "sum" and "max" operations from deterministic values to probabilistic distributions, and apply those operations following the topological order in a directed acyclic graph representing the circuit, to compute the probabilistic distributions of the signal arrival times and then the timing yield of the circuit. It is clear that such approach is effective for combinational circuits and sequential circuits based on edge-triggered flip-flops as their timing graphs are generally acyclic.

On the other hand, level-triggered transparent latches are widely used in high performance circuits due to their low overhead in timing, area, and power, in comparison to edge-triggered flip-flops [4]. In those circuits, signals can propagate through a few latches without being synchronized, which enables time borrowing among multiple combinational stages. Though time borrowing allows the circuit to adapt delay variations at run-time, it complicates timing verification even for deterministic delays [5, 6, 7, 8].

To address the timing verification problem for general transparently latched circuits under process variations, two approaches [9, 10] were proposed to estimate the timing yield. In [9], a structural condition for deterministic delays is first identified stating that a clock schedule is valid iff there is no positive cycle in the latest latch timing graph and no negative cycle in the earliest latch timing graph. Then the authors were able to show that the key problem to compute timing yield is to compute the probability that a graph with edge weights given as random distributions has no positive cycle. They proposed an algorithm to extract acyclic subgraphs from the graph such that block-based SSTA techniques can be applied to estimate the timing yield. As there are cases where cycles are missed so the estimation is optimistic, multiple extractions must be performed at random to ensure coverage. However, it is not clear how the number of the extractions could be bounded to cover most of, if not all, the cycles. In [10], the authors proposed to directly extend the SMO algorithm [5, 6, 7] by incorporating the statistical "sum" and "max" operations. To achieve algorithmic convergence, a quantity called iteration mean is introduced. However, as the convergence depends both on the circuit structure and on the accuracy of the statistical operations, the number of the iterations cannot be bounded.

In this paper, we present the *SGT-PC* algorithm to compute the probability that a graph with random edge weights has no positive cycle in order to accurately calculate the timing yield of transparently latched circuits under process variations. The proposed algorithm is based on statistical "sum" and "max" operations, and Chen and Zhou's formulation [9]. The proposed algorithm can cover *all* cycles through a structural graph traversal, within a *polynomial* number ($O(|V|^2|E|)$) of statistical operations. Further more, we develop a decomposition technique based on strongly connected components (SCCs) [11] to improve the practical efficiency of the SGT-PC algorithm without sacrificing accuracy, and a heuristic approach to limit the region of graph traversal that allows designers to trade-off accuracy with running time. The experiments with delays under multivariate normal distribution [2, 3] show the proposed approach can generate more accurate results in less running time in comparison to [9]. Note that the overall accuracy to compute the timing yield depends on, first, the ability to cover all cycles, and second, the accuracy of statistical operations. We focus on the former in this paper but leave the latter to related SSTA research works, e.g. [12].

978-1-4244-5766-3/10/$26.00 ©2010 IEEE

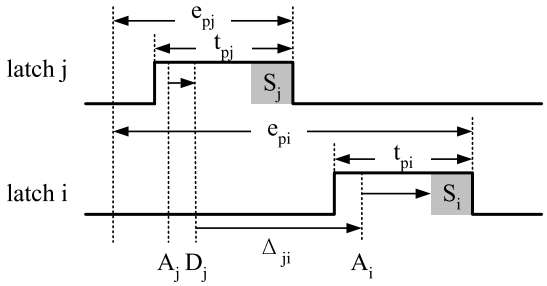

Fig. 1. Latch latest timing

The rest of paper is organized as follows. In Section II, the problem formulation of *Statistical Positive Cycle Detection* (SPCD) is introduced based on [9]. In Section III, the SGT-PC algorithm is presented to solve the SPCD problem. In Section IV, practical improvements for the SGT-PC algorithm are developed. After experimental results are shown in Section V, we conclude the paper in Section VI.

II. Problem Formulation

In [9], Chen and Zhou extended Szymanski and Shenoy's work [7] to establish structural conditions for clock schedules to be valid for transparently latched circuits. The conditions are based on a conservative SMO formulation [5, 6] and can be used for both deterministic and statistical timing verifications. Details follow.

Let L be the set of the transparent latches in the circuit. A clock schedule is the assignment of a clock signal p_i to each latch $i \in L$ such that all clock signals should be of the same clock period ϕ but can have different phases. Choosing an arbitrary time frame of length ϕ, a clock signal p can be denoted by its starting and ending time (s_p, e_p). Without loss of generality, assume $s_p < e_p$ and let $t_p = e_p - s_p$ be the width of p. For any pair of latches i and j, let Δ_{ij} be the longest combinational path delay from i to j and let δ_{ij} be the shortest one. Let S_i and H_i be the setup and hold time of latch i respectively. The clock schedule is valid iff there exist latest signal arrival and departure time (A_i, D_i), and earliest signal arrival and departure time (a_i, d_i), for each latch i, to satisfy the following two sets of constraints, while the first set is illustrated in Fig. 1.

Latest timing constraints: (1)

$$A_i - D_j \geq \Delta_{ji} - E_{p_j p_i},\ \forall j \rightarrow i,$$
$$D_i - A_i \geq 0,\ D_i \geq c - t_{p_i},\ -A_i \geq S_i - \phi,\ \forall i \in L.$$

Earliest timing constraints: (2)

$$a_i - d_j \leq \delta_{ji} - E_{p_j p_i},\ \forall j \rightarrow i,$$
$$d_i \leq \phi - t_{p_i},\ -a_i \leq -H_i,\ \forall i \in L.$$

Note that since (A_i, D_i) and (a_i, d_i) are in latch i's local time, i.e. starting at the falling edge of the clock signal, the local time translator E is used to translate from one latch's local time to the other's, which is defined as,

$$E_{p_j p_i} \triangleq \begin{cases} e_{p_j} - e_{p_i}, & \text{if } e_{p_j} > e_{p_i}, \\ \phi + e_{p_j} - e_{p_i}, & \text{otherwise.} \end{cases}$$

Clearly, Eq. (1) is a system of difference inequalities as there are at most two variables per inequality, one with coefficient 1 and the other with coefficient -1. Based on the correspondence between such a system and the longest path problem on a graph [11], Chen and Zhou [9] proposed to construct the latest latch timing graph corresponding to the latest timing constraints. To be more specific, for each constraint $x - y \geq z$ in Eq. (1), the vertices x and y are introduced to the graph for the variables, and an edge from x to y is introduced with weight z. For the constraints where x or y is missing, the dummy vertex O is introduced with the corresponding edges to represent the reference time 0. Similarly, the earliest latch timing graph can be constructed from the earliest timing constraints. Under process variations, the quantities at the right-hand-side in Eq. (1) and (2) become random variables. The timing yield of the circuit, which is defined as the probability that the clock schedule is valid, is shown [9] to be equal to to the probability that the latest latch timing graph has no positive cycle and the earliest latch timing graph has no negative cycle. Due to the symmetry between positive cycles and negative cycles (through negation of edge weights), the following *Statistical Positive Cycle Detection* (SPCD) problem is formulated.

Problem 1 (Statistical Positive Cycle Detection) *Let $G = (V, E)$ be a graph. For each edge $(i, j) \in E$, let $w(i, j)$ be the random edge weight. Assume that the joint distribution of w is known. Determine a random variable X and its distribution such that G has a positive cycle iff $X > 0$.*

The SPCD problem should be solved twice – once for the latest latch timing graph and once for the earliest latch timing graph with negated edge weights. Then the timing yield can be computed as the probability that both random variables are non-positive.

Chen and Zhou [9] proposed the *PCycle* algorithm to solve the SPCD problem assuming that the edge weights follow multivariate normal distribution and thus the statistical "sum" and "max" operations are available [2, 3]. Moreover, they found that the SPCD problem for the earliest latch timing graph can be directly solved through block-based SSTA techniques for acyclic graphs [2, 3] because of its special structure. However, as pointed out in [9], the PCycle algorithm may miss cycles and thus is optimistic, i.e., it is possible that the obtained $X \leq 0$ for some process corner but G does have a positive cycle, even assuming that the statistical operations are perfectly accurate. We will present in the next section our *SGT-PC* algorithm to solve the SPCD problem that can cover *all* cycles, with the same assumption as [9] for the edge weights distribution and the statistical operations. Note that both SGT-PC algorithm and the PCycle algorithm can be applied to edge weights beyond multivariate normal distribution, as long as the statistical "sum" and "max" operations with proper accuracy are available.

III. Structural Graph Traversal Based Statistic Positive Cycle Detection

A. Algorithmic Idea

For a cycle c in G, let $w(c)$ be the cycle weight, defined as the summation of the edge weights along c, i.e.,

$$w(c) \triangleq \sum_{(i,j) \in c} w(i, j).$$

For the SPCD problem, consider the approach to compute X as the maximum of all the cycle weights, i.e.,

$$X_{all} \triangleq \max_{\text{cycle } c} w(c).$$

Since a positive cycle can be constructed by traversing another positive cycle multiple times, it is straight-forward that $X_{all} = +\infty$ when G has a positive cycle. On the other hand, when G has no positive cycle, we should have $X_{all} \leq 0$. Therefore, it is possible that the mean of X_{all} is unbounded and thus it is quite difficult to approximate X_{all} directly with any distribution of finite mean, which is usually the case for the available block-based SSTA techniques, e.g. [2, 3].

From the formulation of the SPCD problem, it is clear that any random variable X satisfying that $X > 0$ iff $X_{all} > 0$ would be a solution. Consider a positive cycle c in G. Recall that a simple cycle is a cycle that traverses any vertex at most once. Suppose that c can be decomposed into simple cycle(s) $c_1, c_2, \ldots, c_l$. Then since

$$\sum_{k=1}^{l} w(c_k) = w(c) > 0,$$

there must exist some $1 \leq k \leq l$ such that $w(c_k) > 0$, i.e. G has a positive simple cycle. Therefore, after defining S to be the set of simple cycles in G, we have the following lemma,

Lemma 1 *Define*

$$X_S \triangleq \max_{c \in S} w(c).$$

Then $X_S > 0$ iff $X_{all} > 0$.

The above lemma implies that X_S is a solution of the SPCD problem. Moreover, since a simple cycle contains at most $|V|$ vertices, S is a finite set. Therefore, in comparison to X_{all}, X_S can be approximated more accurately with available statistical operations.

However, to compute X_S by enumerating simple cycles explicitly, e.g. through depth-first-search [11], is computationally prohibitive, since S may contain exponential number of cycles in terms of $|V|$. To circumvent such difficulty, our algorithmic idea is to identify a set of cycles S^* that includes S such that the maximum cycle weights of S^*, defined as,

$$X_{S^*} \triangleq \max_{c \in S^*} w(c),$$

can be computed without explicitly enumerating the cycles. The correctness of the idea is guaranteed by the following lemma.

Lemma 2 *If $S \subseteq S^*$, then $X_{S^*} > 0$ iff $X_S > 0$.*

B. The SGT-PC Algorithm

For a particular vertex $i \in V$, let L_i be the set of cycles in G that contain at most $|V|$ vertices and traverse the vertex i exactly once. As a simple cycle contains at most $|V|$ vertices, it is straight-forward that,

$$S \subseteq L \triangleq \bigcup_{i \in V} L_i.$$

Subroutine VertexDist

Inputs

$G = (V, E)$: the graph. i: a vertex in V.

Outputs

X_i: maximum cycle weight for cycles passing i exactly once.

1 **For** all $j \in V$:

2 $\quad W_{ij}^1 \leftarrow \begin{cases} \max_{(i,j)\in E} w(i,j), & \exists (i,j) \in E, \\ -\infty, & \nexists (i,j) \in E. \end{cases}$

3 **For** $k \leftarrow 2$ to $|V|$:

4 $\quad$ **For** all $j \in V$:

5 $\quad\quad W_{ij}^k \leftarrow \max_{(v \neq i) \wedge ((v,j)\in E)} W_{iv}^{k-1} + w(v,j).$

6 $X_i \leftarrow \max(W_{ii}^1, W_{ii}^2, \ldots, W_{ii}^{|V|}).$

Fig. 2. The VertexDist subroutine.

Moreover, after defining $X_i \triangleq \max_{c \in L_i} w(c)$, we have,

$$X_L = \max_{i \in V} X_i. \tag{3}$$

Therefore, X_L can be computed through applying the statistical "max" operation to all X_i's.

Consider a cycle c in L_i. Assume that c is not a self-loop, i.e. the cycle with exactly one edge from i to i. Then c must be consisted of a path p from i to some vertex j that traverses i only once at the head, and an edge from j to i. For $j \neq i$ and $1 \leq k \leq |V|$, define W_{ij}^k be the maximum path weight, i.e. the summation of edge weights along the path, among the paths from i to j that traverse i only once at the head and contain exactly k edges. For $1 \leq k \leq |V|$, define W_{ii}^k be the maximum cycle weight among the cycles that traverse i exactly once and contain exactly k edges. Assuming the statistical "sum" and "max" operations are available, we have,

$$W_{ii}^1 = \max_{(i,i)\in E} w(i,i), \tag{4}$$

$$W_{ii}^k = \max_{(j \neq i) \wedge ((j,i)\in E)} W_{ij}^{k-1} + w(j,i), \forall 2 \leq k \leq |V|, \tag{5}$$

$$X_i = \max_{1 \leq k \leq |V|} W_{ii}^k. \tag{6}$$

In addition, W_{ij}^k can be calculated recursively by dynamic programming as follows.

$$W_{ij}^1 = \max_{(i,j)\in E} w(i,j), \tag{7}$$

$$W_{ij}^k = \max_{(v \neq i) \wedge ((v,j)\in E)} W_{iv}^{k-1} + w(v,j), \forall 2 \leq k \leq |V|. \tag{8}$$

Note that the above random variables should take the value $-\infty$ if there is no operand for the "max" operations.

Based on Eq. (4) to (8), we design the VertexDist subroutine as shown in Fig. 2 to compute X_i. In this subroutine, we apply Eq. (4) and (7) in the loop on line 1, apply Eq. (5) and (8) in the loop on line 4, and compute X_i on line 6 from Eq. (6).

An example illustrating the VertexDist subroutine is shown in Fig. 3. For simplicity, deterministic edge weights are used instead of random ones. It should be clear that the VertexDist subroutine uses the statistical "sum" and "max" operations instead of the deterministic ones. The example graph is shown on the left, which has 4 vertices and 7 edges. The edge weights

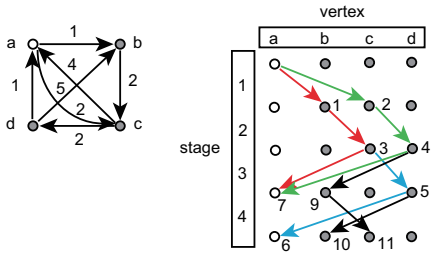

Fig. 3. Compute X_a by VertexDist.

Algorithm SGT-PC

Inputs

$G = (V, E)$: the graph.

Outputs

X_L: maximum cycle weight in L

1 **For** all $i \in V$:
2 $X_i \leftarrow \text{VertexDist}(G, i)$.
3 $X_L \leftarrow \max_{i \in V} X_i$.

Fig. 4. The SGT-PC algorithm.

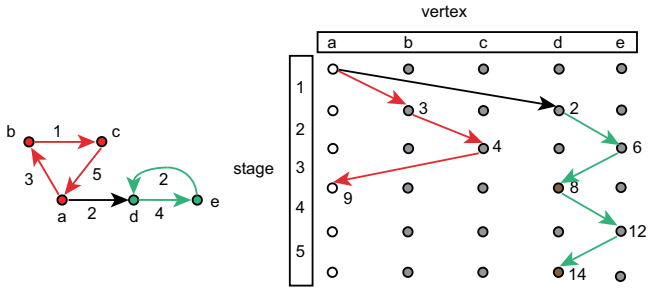

Fig. 5. A graph with a path never traveling back to a.

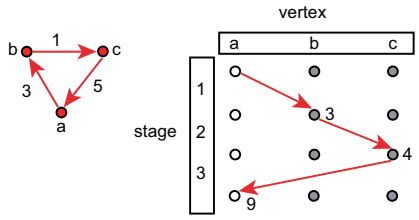

Fig. 6. Compute X_a in the SCC containing a.

are annotated on the edges. The subroutine traverses the graph from vertex a in order to compute X_a. The progress of the subroutine is shown on the right, where the stages in the loop on line 3 are expanded into rows. In the first stage, since (a, b) and (a, c) are the fanout edges from a, $W^1(a, b)$ and $W^1(a, c)$ are computed as 1 and 2 respectively, and are annotated in the figure. As there is no edge from a to a or d, $W^1(a, a)$ and $W^1(a, d)$ are set as $-\infty$, which are not shown for ease of presentation. This progress continues until $|V|$ stages have been explored. The X_i will take $W^3_{aa} = 7$, which is the weight of the cycle $a \to b \to c \to a$.

We design the SGT-PC algorithm as shown in Fig. 4 by combining the VertexDist subroutine with Eq. (3). The correctness of the algorithm is stated in the following theorem.

Theorem 1 *The SGT-PC algorithm generates a random variable X_L and its distribution such that G has a positive cycle iff $X_L > 0$.*

Since statistical operations are used in SGT-PC, the complexity of the SGT-PC algorithm depends on that of the statistical operations. Assume that it will take $O(R)$ space for each random variable and $O(T)$ time for each "sum" or "max" operation. Consider the VertexDist subroutine. For the loop on line 1, it takes at most $O(|E|)$ operations. For the loop on line 4, it can be implemented by traversing each edge exactly once. So it will take $O(|E|)$ operations. Line 6 requires $O(|V|)$ operations. Therefore, the time complexity for the VertexDist subroutine is $O(T|E||V|)$. On the other hand, although $|V|^2$ random variables W^k_{ij} are computed in the subroutine, at most $3|V|$ variables are required to be stored at any given time. To be more specific, in the loop on line 3, only W^k_{ij}'s and W^{k-1}_{ij}'s are required and the $W^{k'}_{ij}$'s for all $k' < k - 1$ and $j \neq i$ can be discarded. Therefore, the space complexity for the VertexDist subroutine is $O(R|V|)$. Based on above discussions, we have the following theorem.

Theorem 2 *Assume that each random variable requires $O(R)$ storage and each "sum" or "max" operation requires $O(T)$ time. Then the time complexity of the SGT-PC algorithm is $O(T|V|^2|E|)$ and the space complexity is $O(R|V|)$.*

IV. Practical Implementation Considerations

A. Strongly Connected Components Decomposition

An observation of the SGT-PC algorithm is that the same cycle may be counted multiple times when X_i's are computed. For example, consider the cycle $a \to b \to c \to a$ in Fig. 3. It is counted for 3 times in the progress of computing X_a, X_b, and X_c. A straight-forward method to avoid those extra computations is to remove a and all the edges incident on a from G once X_a is computed. Such a modification of the SGT-PC algorithm will still be correct as any simple cycle containing a is included in L_a already.

Further more, in the progress to compute X_i, if a path starting from i never travels back to i, the path weight will not contribute to X_i. Therefore, if such paths can be identified before applying the VertexDist subroutine, they can be excluded to speed-up the algorithm practically without affecting the correctness of the algorithm. For example, consider the graph shown in Fig. 5. Starting from a, the paths containing d will never travel back to a and thus the vertex d can be excluded when computing X_a to save practical running time.

Since a path will travel back to its starting point iff all the vertices along the path belong to a strongly connected component (SCC) [11], we propose to decompose the graph into SCCs before applying the VertexDist subroutine. Under such decomposition, paths that will not travel back to its starting point are eliminated from computation. Moreover, since paths are limited to each SCC, the number of required stages on line 3 of the VertexDist subroutine, is the size of the SCC, which is usually smaller than $|V|$. For the example shown in Fig. 5, we perform SCC decomposition and identify the SCC containing a before applying the VertexDist subroutine as shown in Fig. 6. Clearly, we save more than half of the statistical operations.

Algorithm SGT-PC$_{SCC}$

Inputs

$G = (V, E)$: the graph.

Outputs

X: maximum cycle weight

```
1  Q ← ∅.
2  SCC-Enqueue(G, Q).
3  While Q ≠ ∅:
4    G' ← Dequeue(Q).
5    i ← any vertex in G'.
6    X_i ← VertexDist(G', i).
7    SCC-Enqueue(G' − i, Q).
8  X ← max_{i∈V} X_i.
```

Fig. 7. The SGT-PC$_{SCC}$ algorithm.

We design the SGT-PC$_{SCC}$ algorithm as shown in Fig. 7 to incorporate the above ideas. In this algorithm, a queue Q of graphs is used to maintain the SCCs found in the progress of the algorithm. For a given graph G' and a queue Q, we assume that the SCC-Enqueue subroutine will decompose G' into SCCs and put the SCCs into Q. As SCC decomposition can be finished within $O(|V| + |E|)$ time and space [11], we have the following theorem for the correctness and the complexity of the SGT-PC$_{SCC}$ algorithm.

Theorem 3 *The SGT-PC$_{SCC}$ algorithm generates a random variable X and its distribution such that G has a positive cycle iff $X > 0$. The time complexity of the algorithm is $O(T|V|^2|E|)$ and the space complexity is $O(R|V|)$.*

Although the theoretical complexities of the SGT-PC and the SGT-PC$_{SCC}$ algorithm are the same, we observed in our experiments that the SCC decomposition contributes significantly to the practical efficiency of the SGT-PC$_{SCC}$ algorithm. Though the latest latch timing graph contains only one SCC at the beginning, once a few vertices are removed after a few iterations, the graph may be decomposed into many small SCCs, whose maximum cycle weights can be computed efficiently.

B. Limiting Number of Traversal Stages

To cover all simple cycles in the VertexDist subroutine, all cycles containing at most $|V|$ vertices are considered in the loop on line 3. We propose to explore the trade-off between solution accuracy and running time by introducing a bound N to the number of stages such that the loop on line 3 will execute at most $\min(N, |V|)$ times and thus some cycles could be missed from the computation.

The motivation of such restriction is as follows. First of all, if the missed cycle is not a simple cycle, then the solution accuracy will not be affected due to Lemma 2. Second, the SPCD problem is formulated to verify the latest latch timing graph. In such graph, a simple cycle containing more than N vertices usually corresponds to a possible time borrowing across $N/2$ latches. As designers tend to manage the complexity of latch timing by restricting time borrowing to only a few latches, it is unlikely that a timing violation will happen across $N/2$ latches and time borrowing will tolerate acceptable variations in long cycles In such case, it should be up to the designers to determine the bound N for the number of traversal stages.

V. Experiments

We derive the experimental circuits from the ISCAS89 benchmark. Each flip-flop is replaced by a pair of transparent latches defined as the *front* latch and the *back* latch. While the front latches are kept at the locations of the flip-flops, the back latches are moved into the combinational part of the circuit randomly while preserving the circuit functionality. A two-phase clock consisting of a clock and its inversion is assigned to the latches such that all the front latches are assigned the original clock and all the back latches are assigned the inverted clock. The gate delays are assumed to follow multivariate normal distribution and are determined as follows. First, we assign each gate a nominal delay which is equal to the number of its fanouts. Then, we assign a standard deviation to each gate that is within $20 - 30\%$ of its nominal delay. Assuming that each gate has a unit size, a wire-length driven global placement is performed by mPL6[13] to determine the rough positions of each gate in a 4×4 grid of the chip area. We assume that two gate delays are perfectly correlated if they are within a same block. Otherwise, the covariance of two gate delays is assigned to be inversely proportional to the distance of the two grid centers. The latest latch timing graph is then generated by applying block-based SSTA techniques [2, 3].

We implement our SGT-PC$_{SCC}$ algorithm with the statistical "sum" and "max" operations from [3] as gate delays are assumed to follow multivariate normal distribution. For comparison, we implement the PCycle algorithm [9]. Both statistical operations from [2] and [3] were experimented for PCycle but the ones from [2] generated more accurate results and thus are used. To verify the accuracy of both SGT-PC$_{SCC}$ and the PCycle algorithm, we implement an approach based on Monte Carlo simulation and Bellman-Ford algorithm [11] to evaluate the probability that the latest latch timing graph has no positive cycle. For each graph, we run 10000 iterations of Monte Carlo simulation and assume that the result is accurate. All codes are implemented in C++, compiled by GCC version 3.4, and run on a Linux PC with a 2.4GHz processor and 4.0GB memory.

The experimental results are compared in Table I. For circuits with a $*$ mark, we limited the number of traversal stages. Clearly, our SGT-PC$_{SCC}$ algorithm generates more accurate results in less running time in comparison to the PCycle algorithm for most circuits, while both algorithms use much less running time in comparison with Monte Carlo simulation. On average, the error of the SGT-PC$_{SCC}$ algorithm is 0.21%, while the error of the PCycle algorithm is 1.03%.

As the majority of the running time for both algorithms are spent on statistical operations, we present in Table II the number of the statistical operations used for each circuit. It can be seen that: for small circuits where the SGT-PC$_{SCC}$ algorithm can guarantee to cover all cycles, the number of operations executed by the PCycle algorithm are not necessarily smaller although cycles may be missed; for large circuits where both algorithms may miss cycle, the SGT-PC$_{SCC}$ algorithm applies the operations more effectively and achieves a better accuracy.

TABLE I
COMPARISON OF PCYCLE, SGT-PC$_{SCC}$, AND MONTE CARLO SIMULATION.

circuit			PCycle			SGT-PC$_{SCC}$			Monte Carlo	
name	$\|V\|$	$\|E\|$	yield%	time(s)	error%	yield%	time(s)	error%	yield%	time(s)
s27	7	21	97.37	0.02	0.11	97.37	0.01	0.11	97.26	0.34
s208.1	27	113	99.16	0.08	0.54	98.90	0.02	0.28	98.62	3.45
s382	49	249	99.76	0.25	2.53	97.37	0.03	0.14	97.23	11.68
s420.1	50	283	98.37	0.25	-0.21	98.82	0.03	0.24	98.58	13.43
s526	53	263	94.96	0.27	0.05	94.81	0.12	-0.10	94.91	19.93
s832	83	336	97.99	0.57	-0.37	98.06	0.15	-0.3	98.36	26.99
s1196	127	391	97.77	0.76	-1.63	99.39	0.43	-0.01	99.40	68.34
s1423	159	2315	96.38	2.39	1.55	94.66	0.64	-0.17	94.83	246.14
s5378*	514	2697	97.19	26.06	1.15	96.37	3.16	0.33	96.04	1228.06
s13207*	1365	5714	97.92	197.20	1.93	96.15	38.28	0.16	95.99	7515.15
s13207.1*	1337	5673	99.93	204.94	1.05	98.98	35.66	0.10	98.88	7613.59
s15850*	1275	17901	98.38	542.25	0.80	98.19	47.05	0.61	97.58	15136.97
s35932*	3145	10838	97.12	977.72	-2.08	99.50	466.29	0.30	99.20	35704.76
s38417*	3419	34971	96.91	3083.51	-0.69	97.62	494.23	0.02	97.60	85445.45
s38514*	4253	26208	99.70	2906.01	0.70	99.31	968.06	0.34	98.97	93268.92

TABLE II
COMPARISON ON THE COUNT OF "SUM" AND "MAX" OPERATIONS FOR PCYCLE AND SGT-PC$_{SCC}$.

circuit	PCycle		SGT-PC$_{SCC}$	
name	sum#	max#	sum#	max#
s27	2173	856	765	249
s208.1	26352	16146	7235	2076
s382	102916	67148	24396	7633
s420.1	106078	72754	21036	7538
s526	116123	79674	152221	75059
s832	314128	200318	207288	73183
s1196	459642	261392	659943	202957
s1423*	6823K	5478K	741K	339K
s5378*	13900K	9193K	1849K	512K
s13207*	88255K	52408K	540K	103K
s13207.1*	95489K	57157K	1467K	277K
s15850*	267978K	211626K	23694K	10540K
s35932*	274623K	167203K	119737K	30157K
s38417*	1571896K	1278600K	6976K	2350K
s38584*	1377417K	970265K	84463K	18272K

VI. CONCLUSION

In this paper, we presented the SGT-PC algorithm to solve the Statistical Positive Cycle Detection problem for statistical timing verification of transparently latched circuits. Based on block-based statistical timing analysis techniques, the SGT-PC algorithm was able to cover all cycles through a structural graph traversal within $O(|V|^2|E|)$ number of statistical "sum" and "max" operations. A decomposition technique based on strongly connected components was developed to improve the practical efficiency of the algorithm without sacrificing accuracy, and a heuristic approach to limit the region of graph traversal was proposed to allow designers to trade-off accuracy with running time. The proposed approach was confirmed by the experimental results in comparison to previous works and Monte Carlo simulation.

ACKNOWLEDGMENTS

This work is supported in part by the Educational and Research Initiative Fund from Illinois Institute of Technology.

REFERENCES

[1] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: from basic principles to state of the art," *IEEE TCAD*, 27(4):589–607, Apr. 2008.

[2] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE TCAD*, 24(9):1467–1482, Sep. 2005.

[3] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *DAC*, pp. 331–336, 2004.

[4] C. Ebeling and B. Lockyear, "On the performance of level-clocked circuits," in *Proc. Advanced Research VLSI*, pp. 342–356, 1995.

[5] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "$checkT_c$ and $minT_c$: Timing verification and optimal clocking of synchronous digital circuits," in *ICCAD*, pp. 552–555, 1990.

[6] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," *IEEE TCAD*, 11(3):322–332, Mar. 1992.

[7] T. G. Szymanski and N. Shenoy, "Verifying clock schedules," in *ICCAD*, pp. 124–131, 1992.

[8] N. Shenoy, "Timing issues in sequential circuits," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, 1993.

[9] R. Chen and H. Zhou, "Statistical timing verification for transparently latched circuits," *IEEE TCAD*, 25(9):1847–1855, Sep. 2006.

[10] L. Zhang, J. Tsai, W. Chen, Y. Hu, and C. C.-P. Chen, "Convergence-provable statistical timing analysis with level-sensitive latches and feedback loops," in *ASPDAC*, pp. 941–946, 2006.

[11] T. H. Cormen, C. E. Leiserson, R. H. Rivest, and C. Stein, *Introduction to Algorithms, 2nd ed*, The MIT Press, 2001.

[12] D. Sinha, H. Zhou, and N. Shenoy, "Advances in computation of the maximum of a set of Gaussian random variables," *IEEE TCAD*, 26(8):1522–1533, Aug. 2007.

[13] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," in *ISPD*, pp. 212–214, 2006.