Statistical Bellman-Ford Algorithm With An Application to Retiming

Mongkol Ekpanyapong Thaisiri Waterwai[†] Sung Kyu Lim

School of Electrical and Computer Engineering
Georgia Institute of Technology
{pop, limsk}@ece.gatech.edu

[†]Dept. of Industrial Engineering and Operations Research
University of California, Berkeley
thaisiri@uclink.berkeley.edu

Abstract— **Process variations in digital circuits make sequential circuit timing validation an extremely challenging task. In this paper, a Statistical Bellman-Ford (SBF) algorithm is proposed to compute the longest path length distribution for directed graphs with cycles. Our SBF algorithm efficiently computes the statistical longest path length distribution if there exist no positive cycles or detects one if the circuit is likely to have a positive cycle. An important application of SBF is Statistical Retiming-based Timing Analysis (SRTA), where SBF is used to check for the feasibility of a given target clock period distribution for retiming. Our gate and wire delay distribution model considers several high-impact intra-die process parameters and accurately captures the spatial and reconvergent path correlations. The Monte Carlo simulation is used to validate the accuracy of our SBF algorithm. To the best of our knowledge, this is the first paper that propose the statistic version of the longest path algorithm for sequential circuits.**

I. Introduction

Process variations in digital circuits make circuit timing validation an extremely challenging task. Variations on several high-impact intra-die process parameters such as effective gate length, wire width, and so forth, can easily invalidate the timing predictions made before the fabrication [1]. Therefore, statistical timing analysis tools that model gate and wire delay as probability distribution function became increasingly popular to tackle the timing validation under process variations [2], [3], [4]. However, most of the existing works focus on combinational circuits or sub-circuits (after FF removal) and fail to address sequential circuit timing validation directly. Partitioning circuit into sub-circuits and solving the problem on a sub-circuit by sub-circuit basis lead to a sub-optimal solution. By considering the sequential circuit, timing analysis can be done by using longest path algorithms that can handle graphs with negative cycles such as the Bellman-Ford algorithm. There are many CAD algorithms that adopt the Bellman-Ford algorithm including scheduling[5], clock scheduling[6], verification[7], and retiming [8]. A recent work on static timing analysis for sequential circuits [8] allows the users to model FFs and use them to predict the timing information *after* retiming. This work achieves a significant performance improvement by exploiting retiming-aware timing slack. Our goal in this paper is to develop the Statistical Bellman-Ford (SBF) algorithm. In addition, we show an application of SBF on global placement using retiming [8].

In this paper, we first develop a Statistical Bellman-Ford (SBF) algorithm to compute the longest path length distribution for directed graphs with negative cycles. We first prove that a statistical extension of the original Bellman-Ford algorithm correctly computes the longest path length distribution for the true distribution, but it requires infinite amount of time for the continuous distribution. Next, we show that two straightforward extensions of the Bellman-Ford algorithm for statistical analysis can not guarantee the correctness of the results. Lastly, we propose our SBF algorithm that closely approximates and efficiently computes the statistical longest path length distribution if there exists no positive cycles or detects one if the circuit is likely to have a positive cycle. Our SBF algorithm is integrated into SRTA, where SBF checks for the feasibility of a target clock period distribution for retiming. We show that the final critical path delay distribution after retiming is the statistical maximum among all primary outputs and all feedback vertices. The Monte Carlo simulation is used to validate the accuracy of our SRTA algorithm.

The remainder of the paper is organized as follows. Section II presents our statistical Bellman-Ford algorithm. Section III presents our statistical retiming-based algorithm and its application in retiming. We present the experimental results in Section IV and conclude in Section V.

II. Statistical Bellman-Ford Algorithm

We first provide the analysis of statistical longest path algorithm for the sequential circuit including its properties. Next, we show that the simple modified Bellman-Ford algorithms can not compute the statistical longest path correctly. Finally, we propose a modified version of the Bellman-Ford algorithm that closely approximates the statistical longest path length distribution for the sequential circuit.

A. Statistical Longest Path Analysis

We first introduce a stochastic version of the Bellman-Ford algorithm that correctly solves the stochastic longest path problem for true distribution. Before we do so, we first introduce some quantities in the probability theory that are required to develop algorithms. For more precise definitions of the quantities, see [9], [10]. Let Ω be the set of outcomes of a fabrication process. A subset of Ω is called an event. Let $\mathbf{P}$ be a function that assigns a probability to each event. A random variable $\mathbf{X} : \Omega \to \mathbb{R}^*$ maps each outcome $\varpi \in \Omega$ to a number in the extended real line $\mathbb{R}^* \triangleq [-\infty, \infty]$. The probability that a random variable $\mathbf{X}$ takes a value in a subset M of $\mathbb{R}^*$ is $\mathbf{P}[\varpi | \mathbf{X}(\varpi) \in M]$. Assume that the probability $\mathbf{P}$ determines the joint (and hence, marginal and conditional) distributions of all random variables of interest. Let $G = (V, E)$ be a directed graph with a source node s and a sink node t, and $w : E \times \Omega \to \mathbb{R}$ be an associated edge-length function, which is a random variable for each edge $(u, v) \in E$. We assume without loss of generality that there is no weight on nodes (since we can push the weights on nodes to their fan-in edges). Let K denote the number of directed *simple* (i.e., no cycles) $s-t$ paths in G, and $l_i : \Omega \to \mathbb{R}$ denote the length of the i^{th} path, $i = 1, \ldots, K$. Also, let $G(\varpi)$ be the graph G with length $w(u, v)(\varpi)$ on edge $(u, v) \in E$. If $\mathbf{X}$ is the longest path of G, it is defined as follows: for each $\varpi \in \Omega$, $\mathbf{X}(\varpi) = \max\{l_1(\varpi), \ldots, l_K(\varpi)\}$, if there is no positive cycle in $G(\varpi)$, and $\mathbf{X}(\varpi) = \infty$, otherwise. The distribution of $\mathbf{X}$ is determined by the probability measure $\mathbf{P}$ as mentioned above. We define the *Statistical Longest Path Problem* as that of finding the distribution of the longest $s-t$ path in $G = (V, E)$ with edge length function $w : E \times \Omega \to \mathbb{R}$.

We extend the Bellman-Ford (BF) algorithm to obtain the outcome-by-outcome Statistical Bellman-Ford algorithm (oSBF). An illustration is shown in Figure 1. The algorithm is similar to the original Bellman-Ford algorithm but it is called for each outcome untill all possible outcomes are computed. The algorithm starts by first initialize all variables in the initialization step. The value of $a[v]$ represents the arrival time of node v. At the beginning, the arrival time of all nodes is set to $-\infty$, except the source node that has the value zero. After that, the relaxation step is called. Similar to the Bellman-Ford algorithm, the algorithm will stop when there is no update. During the relaxation, for a given outcome, the algorithm checks for all edges in the graph whether the value of sink node of each edge is greater than the summation of the gate delay of source node and the wire delay of that edge or not (the delay contraint). The algorithm stops when there is no update in the graph that is the value of sink node is greater than or equal to the summation of the delay of source node and the wire delay on the edge. During the positive cycles detection, the algorithm checks for each edge whether, is there any edge that violate the delay constraint. If there is an violation, the algorithm return false, otherwise the algorithm returns true with delay $a[t]$ of the sink node.

As opposed to updating a certain set of *constants* (such as arrival times $a[i]$) in the BF, at each step of the oSBF, we are required to update certain random variables that are *functions* on Ω by updating their values for each outcome $\varpi \in \Omega$. As a result, the complexity of the oSBF is high when the numbers of possible outcomes are large. In fact, when random variables are continuous such as uniform random variables, Ω has uncountably many elements, and the oSBF cannot be carried out in practice (require infinite runtime). However, its properties, which we prove here, are useful for our approximation algorithm, presented in the next section. Note that Monte Carlo simulation can be considered as an approximated version of the oSBF in which certain elements of Ω are sampled according to probability $\mathbf{P}$. The following theorem proves the correctness of the oSBF algorithm.

```
oStatistical Bellman-Ford($G, w, s, t$)

Initialization Step
for (each $v \in V$)
  $a[v](\varpi) \leftarrow -\infty, \forall \varpi \in \Omega$;
  $a[s](\varpi) \leftarrow 0, \forall \varpi \in \Omega$;
$Stop(\varpi) \leftarrow$ NO, $\forall \varpi \in \Omega$;
$g(\varpi) \leftarrow -1, \forall \varpi \in \Omega$;
$iter \leftarrow 1$;

Relaxation Step
while ($Stop(\varpi)$ = NO for some $\varpi \in \Omega$ and $iter < |V|$)
  $iter \leftarrow iter + 1$;
  $\widetilde{\Omega} \leftarrow \{\varpi \in \Omega | Stop(\varpi) =$ NO$\}$;
  $Stop(\varpi) \leftarrow$ YES, $\forall \varpi \in \widetilde{\Omega}$;
  for (each $v \in V$)
      for (each $\varpi \in \widetilde{\Omega}$)
          for (each edge $(u, v) \in E$)
              if     $a[v](\varpi) < a[u](\varpi) + w(u, v)(\varpi)$;
              then  $a[v](\varpi) \leftarrow a[u](\varpi) + w(u, v)(\varpi)$;
                     $Stop(\varpi) \leftarrow$ NO;

Positive Cycles Detection Step
for (each $(u, v) \in E$)
  for (each $\varpi \in \Omega$)
      if ($a[v](\varpi) < a[u](\varpi) + w(u, v)(\varpi)$)
          $g(\varpi) \leftarrow +1$;

Output Step
if ($\mathbf{P}[\varpi | g(\varpi) = +1] > 0$)
  then return FALSE;
  else return TRUE and $a[t]$;
```

Fig. 1. A description of outcome-by-outcome Statistical Bellman-Ford (oSBF) algorithm

Theorem 1: If $\mathbf{P}[\varpi | G(\varpi)$ has a positive cycle$]$ = 0, then $a[i]$ from the oSBF has the same distribution as that of the random variable representing the longest path from s to i, $i \in V$. Otherwise, the oSBF returns FALSE.

Proof: For each outcome $\varpi \in \Omega$, the oSBF is equivalent to the BF, which correctly calculates the lengths of the longest $s-i$ paths $a[i](\varpi), i \in V$ or correctly identifies a positive cycle in $G(\varpi)$. Hence, when the oSBF terminates, the function (random variable) $a[i]$ and the longest $s-i$ path are equal on $\Omega_1 \triangleq \{\varpi \in \Omega | G(\varpi)$ has no positive cycles$\}$ and are different on $\Omega_2 \triangleq \{\varpi \in \Omega | G(\varpi)$ has a positive cycle$\}$. If $\mathbf{P}[\Omega_2] = 0$, then they are equal with probability one, and hence, have the same distribution. Otherwise, there is a positive probability of having a positive cycle. Therefore, the oSBF returns FALSE. ■

We define backward edges and nodes as follows:

Definition 1: For a given order of nodes P, $(u, v) \in E$ is a *forward edge* if u precedes v in P, and a *backward edge* otherwise. In the latter case, node u is called a *backward node*.

Lemma 1: If a node order $L = \{s, v_1, \ldots, v_n, t\}$ is used in the relaxation step of the oSBF, after j relaxation iterations, $a[i]$ from oSBF is the random variable representing the longest (possibly not simple) $s-i$ path in G that contains $j-1$ or fewer (possibly repeated) backward edges.[1]

[1]Similar proof was shown in [11]

Proof: Let B denote the set of all backward edges associated with the order L. Then the subgraph $G_B \triangleq (V, E \setminus B)$ is a directed acyclic graph (DAG). Now we recall the update step for node i in the relaxation step:

$$a[i] := \max_{u \in FI(i)} \Big[a[u] + w(u,i) \Big], \tag{1}$$

where $FI(i) \triangleq \{u \in V | (u,i) \in E\}$ is the set of *fan-ins* of node i. At the first iteration of the relaxation step, when $a[i]$ is updated, $a[u] = -\infty$ for all *backward fan-ins* $u \in FI_b(i) \triangleq \{u \in FI(i) | (u,i) \in B\}$ because from Definition 1, nodes $u \in FI_b(i)$ come after node i in the order, and hence, have not been updated. Thus, at the first iteration of the relaxation step, it is sufficient to perform relaxation on G_B. Since G_B is a DAG, and L is a topological order of G_B, $a[i]$ represents the longest $s - i$ path that contains zero backward edge after the first relaxation step.

Now suppose that the result of Lemma 1 is true up to some $j \geq 1$ (Induction Hypothesis 1: IH 1). At iteration $j + 1$ of the relaxation step, we will show by induction that after $a[i]$ is updated using (1), it represents the longest $s - i$ path containing at most j backward edges. Consider the update for node v_1. Since s is the only node that precedes v_1 in L, all other nodes $u \in FI(v_1)$ are all updated after v_1. By IH 1, $a[u]$ represents the longest $s - u$ path with at most $j - 1$ backward edges for all $u \in FI(v_1)$. From (1), the updated $a[v_1]$ is the longest $s - v_1$ path with at most j backward edges because any $s - v_1$ path with $c > 0$ backward edges has the last edge being a backward edge, and removing such an edge results in a path with $c - 1$ backward edges.

Suppose that $a[v_i], i = 1, \ldots, r$ are now the longest $s - v_i$ paths with at most j backward edges for some $r \geq 1$ (Induction Hypothesis 2: IH 2). Similarly, from (1), the updated $a[v_{r+1}]$ is the longest $s - v_{r+1}$ path with at most j backward edges because any $s - v_{r+1}$ path with $c > 0$ backward edges either has the last edge being a backward edge and removing such an edge results in a path with $c - 1$ edges (this case corresponds to $u \in FI_b(v_{r+1})$, whose $a[u]$ are, from IH 1, the longest $s - u$ paths with at most $j - 1$ backward edges), or has the last edge being a forward edge and removing such edge results in a path with c backward edges (this case corresponds to $u \in FI_f(v_{r+1}) \triangleq \{u \in FI(v_{r+1}) | (u, v_{r+1}) \notin B\}$ whose $a[u]$ are, from IH 2, the longest $s - u$ path with at most j backward edges). This completes the proof. ∎

The following theorem improves the bound on the number of iterations of the relaxation step when there is no positive cycle. In the oSBF, the algorithm automatically terminates once the number of relaxation iterations reaches this bound (by the condition $Stop(\varpi)$ = YES for all $\varpi \in \Omega$). However, this is not true when the distribution of each $a[i]$ is approximately updated. Hence, the bound from this theorem will be useful for our approximation algorithm.

Theorem 2: If $\mathbf{P}[\varpi | G(\varpi)$ has a positive cycle] = 0, and a node order $L = \{s, v_1, \ldots, v_n, t\}$ is used in the relaxation step of the oSBF; then after $k+1$ iterations, $a[i]$ from oSBF has the same distribution as that of the random variable representing the longest path from s to i, $i \in V$, where k is the maximum number of connected backward nodes that can be in a simple $s - t$ path in G.

Proof: Since G has no positive cycles with probability 1, the longest $s - t$ path is a simple path with probability 1. According to the definition of k, the longest path has at most k backward nodes and hence, at most k backward edges. The theorem follows from Lemma 1. ∎

The result from Theorem 2 can be applied to approximation methods that are similar to the oSBF, except at each iteration $a[i], i \in V$ are *approximately* updated. More specifically, if $a[i]$ is a good approximation to the actual $a[i]$ obtained from the oSBF, then after $k + 1$ iterations, $a[i]$, obtained from the approximation method, is also a good approximation to the actual longest $s - i$ path. Hence, when there is no positive cycle with probability one, we can stop the approximation algorithm after $k + 1$ iterations.

B. Limitation of the Bellman-Ford Extensions

To the best of our knowledge, all proposed analytical models for statistical timing analysis suffer from the error introduced by the maximum function. This is because the output of the maximum function results in a new form of distribution. Unlike the true distribution, Bellman-Ford can return incorrect results because of the error from approximated distributions (such as the normal distribution approximation for the delay distribution). This is because the original Bellman-Ford algorithm is not designed to tolerate such error from stochastic computation. More precisely, it is typically assumed that the joint distribution of arrival times is fully characterized by vectors of parameters $\theta_i, i \in V$, which belong to a certain set Θ, which is closed under addition[2]. For example, when each node is assumed to be independently normally distributed, a two-dimensional vector $[\mu_i, \sigma_i^2]' \in \mathbb{R} \times [0, \infty)$ can be used to describe the mean and variance of the arrival time of node i. Let $f_{\max} : \Theta \times \Theta \rightarrow \Theta$ denote the maximum function that approximates the distribution of the maximum by a distribution characterized by a vector in Θ. In this section, two examples are used to demonstrate the drawback of simple extensions of Bellman-Ford algorithms.

The first extension is to use the Bellman-Ford algorithm to compute longest path length distribution as it is in a statistical timing analysis. Based on the Bellman-Ford algorithm, after all vertices are visited and there is still an updated edge, the algorithm will report a positive cycle. Because of the error from the approximation, it is possible that the algorithm can keep update the graph after $|V|$ iterations even when there is no positive cycle. Given two distributions of the same type, e.g., Gaussian distribution, the maximum function of two distributions is likely to exhibit the new distribution which is different from the input distributions. To make the timing analysis simple, the approximated maximum function is used instead that is assuming that the output of maximum distribution results in the same distribution as the input. Note that the good approximated maximum function should result in the

[2]A set A is closed under addition if for any $a, b \in A$, we have $a + b \in A$. This assumption leads to efficient propagation procedure which, however, can be relaxed.

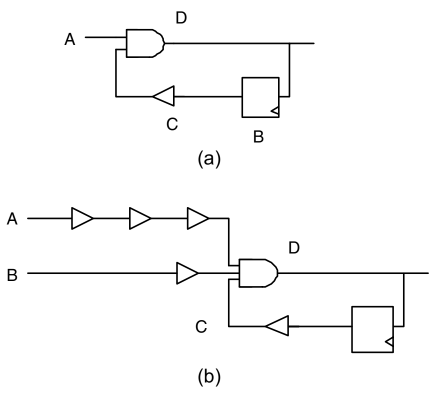

Fig. 2. Illustration of Bellman-Ford update

new distribution that is greater than both input distributions. If the graph has at least a cycle, it is possible that the error from the approximated maximum distribution can result in the repeatly update of the graph over the cycle even though there is no new information propagated from that path or positive cycle. An illustration is shown in Figure 2(a). At the first iteration, an input arrives at node A with value a. Flip-flop B, buffer C, and gate D have delay b, c, and d, respectively. The output value of node D is $D^{(1)} = f_{\max}(a, C^{(0)}) + d = a + d$, where $C^{(0)}$ denotes the vector describing the distribution of the arrival time of gate C at iteration 0. After the signal propagates through flip-flop B and gate C, the new value of node D becomes $D^{(2)} = f_{\max}(a, a + d + b + c) + d$. Let $\triangle^{(2)} = D^{(2)} - D^{(1)}$ denote the change in the value of node D. Now if $d+b+c$ is originally negative with probability one, but its distribution is approximated by that of a negative-mean random variable with a small probability of being positive (for example, a normal random variable with mean -10 and variance 9), then we cannot guarantee that $\triangle^{(2)}$ will be a zero vector. Alternatively, even if the true distribution is initially used, the error from the maximum function could result in the same situation. After one more iteration of the Bellman-Ford, $D^{(3)} = f_{\max}(a, D^{(2)} + b + c) + d = D^{(2)} + \triangle^{(3)}$. A similar argument shows that $\triangle^{(3)}$ may not be a zero vector either. We observe from related experiments that $\triangle^{(i)}, i = 1, 2, \ldots$ are small but might not become zero vectors after the $|V|$ iterations; consequently the algorithm reports a positive cycle.

The second extension is to introduce error bound on longest path length distribution updates, which is named eSBF (error-bounded Statistical Bellman-Ford) algorithm.[3] Specifically, when the change in the distribution (for example the norm of $\triangle^{(i)}$) is less than such a bound, we consider it as no update. Although, imposing a positive error bound δ can help the Bellman-Ford algorithm terminate when the graph has no positive cycle, it could cause the Bellman-Ford to stop too early when δ is too large. Consequently, from Lemma 1, some paths are not considered if it stops before $k+1$ iterations. Figure 2(b) shows an example in which the delay from path A is ignored as follows: Assume that the arrival time value of node D at iteration i is D^i. The total delay on path A,B, C, and gate D are a, b, c, and d respectively. If the change of the arrival time at D resulting from delay propagated through A, which is $\triangle^{(i+1)} = f_{\max}(f_{\max}(D^{(i)} + c, b), a) + d - D^{(i)}$, is considered to be small with respect to the error bound δ, and the arrival time of node A has no further update, the algorithm could terminate without updating D. The information from path A is hence not propagated to the calculation of some other arrival times. As a consequence, the distribution of some $s-t$ paths that contain path A is not accounted for. Depending on the circuit structure, the total error due to this early termination could result in large error in the arrival times.

[3]We use this algorithm in comparison with other Bellman-Ford extensions.

```
Statistical Bellman-Ford($G, w, s, t, \mathbf{P}a$)

Reachability Check Step
DFS($s$); // find all backward edges
$backward\_node \leftarrow 1$;
for (each $v \in V$)
  if (backward edge connected to $v$)
    $backward\_node \leftarrow backward\_node + 1$;
    $list\_backward\_node \leftarrow v$;
$max\_k \leftarrow 0$;
for (each $v \in list\_backward\_node$)
  $k \leftarrow$ DFS'($v$); // backward nodes connected with $v$
  if ($max\_k < k$)
    $k \leftarrow max\_k$;

Initialization Step
for (each $v \in V$)
  $a[v] \leftarrow -\infty$;
$a[s] \leftarrow 0$;

Relaxation Step
for ($iter \leftarrow 1$ to $k+1$)
  for (each $v \in V$)
    $a[v] \leftarrow max_{u \in FI(v)} \Big[ a[u] + w(u,v) \Big]$;

Checking Positive Cycles Step
$\mathbf{P}(cycle) \leftarrow check\_pos\_cycle()$;

Output Step
if ($\mathbf{P}(cycle) \leq \mathbf{P}a$)
  then return FALSE;
  else return TRUE and $a[t]$;
```

Fig. 3. k-Statistical Bellman-Ford algorithm (kSBF) used in our SRTA (statistical retiming-based timing analysis)

C. Statistical Bellman-Ford Algorithm

Our last extension of Bellman-Ford algorithm, named k-Statistical Bellman-Ford (kSBF), is shown in Figure 3. This is an algorithm that closely approximates and efficiently computes the longest path length distribution of directed graphs with negative cycles. We thus use kSBF in our SRTA (statistical retiming-based timing analysis) introduced in the next section. First, a depth first search algorithm is called to identify all backward edges and sort the nodes in a topological order. For each backward node, we call the depth first search DFS' by setting this backward node as a source node. DFS' returns the maximum number of connected backward nodes reachable by a simple path from the given source. The maximum number of connected backward nodes of the graph (=k) is the largest number obtained by the DFS' algorithm. Note that this reachability algorithm needs to be performed only once. If all backward nodes are likely to be connected, the reachability step is not required, and instead, the total number of backward nodes can be used.

After the maximum number of connected backward nodes of the graph is found, we initialize the arrival times of all nodes. Next, the relaxation step is called. For stochastic longest path, $k+1$ iterations are required (from Theorem 2). After the relaxation is done, all simple paths from source to sink are considered according to Theorem 2. Then the stochastic positive cycle detection algorithm is used. We implement a stochastic positive cycle detection algorithm proposed in [12]. The positive cycle detection algorithm starts by first finding all backward edges. Then, it randomly pick a backward edge. Then, it creates a new graph G' by assigning sink node of the backward edge to be a new source node of G' and a source node of that backward edge to be a new sink node of G'. Then, depth first search is performed on G' to find the new set of backward edges and then remove this new set of backward edges from G'. After that, the maximum delay of G' is computed from source node to sink node of the graph (G'). The algorithm is randomly performed for M iterations, when M is an input parameter. Finally, the algorithm computes the probability of having positive cycle, the probability of the maximum of the delay distribution of all new M sink node greater than zero. If the probability of having no positive cycle is less than an acceptable probability, it returns FALSE, otherwise returns TRUE.

III. Statistical Retiming Analysis

The reason that global placement is employed because the process variations can affect both gate and wire. With the increasingly important of wire delay, any optimization/modelling technique should target both gate and wire delay. In addition, spatial correlation information will be available only after placement. Note that Statistical Retiming based Timing Analysis (SRTA) is used to compute the timing solution of final placement only.

A. Modelling Delay Distribution

In this paper, the delay distribution model is based on the first order delay model from [4]. We assume that each gate and wire has the Gaussian distribution. Elmore delay model is used for wire delay computation based on the following equation (similar to [2]):

$$\begin{aligned} d_{int} &= d^0_{int} + \sum_{i\in\Gamma_g}[\frac{\partial d}{\partial L^i_g}]\triangle L^i_g + \sum_{i\in\Gamma_g}[\frac{\partial d}{\partial W^i_g}]\triangle W^i_g \\ &\quad + \sum_{i\in\Gamma_{int}}[\frac{\partial d}{\partial T^i_{int}}]\triangle T^i_{int} \end{aligned}$$

where d^0_{int} is expected value of wire delay. Γ_g and Γ_{int} are the set of grids where all the receiver reside and the interconnect tree traverses, respectively. $\triangle L^i_g$, $\triangle W^i_g$, and $\triangle T^i_{int}$ are random variables representing the variation over the expected value of transistor length, transistor width, and metal thickness respectively. The differentiations are derived based on transistor and wire delay model from [13], [14].

Principal component analysis technique (PCA), similar to [2], is used to derived the first-order form for arrival time delay distribution. The basic idea of PCA is to classify input coefficient into orthogonal terms so that each coefficient term is uncorrelated. Reconvergent correlation can be efficiently handled by PCA. We use a grid hierarchical model for spatial correlation [2]. If two gates are located near each others, they are more correlated than putting them far apart.

There are four operations involved during statistical sequential arrival time computation: maximum, minimum, addition, and subtraction operations. The addition and subtraction of two Gaussian distributions result in another Gaussian distribution. The coefficient of each term in the first order model can be added and subtracted directly. Maximum and minimum functions require the tightness probability calculation [4], which is derived from [15]. Based on this model and the assumption that the maximum and/or minimum of two Gaussian distributions result in a new Gaussian distribution, the coefficient results can be expressed as the summation of product between input distributions and tightness probabilities.

In this paper, wire delay is computed based on Elmore delay model. Since the actual wire distance is not known until routing has been done, the approximated analytical model similar to [16] is used instead. We assume 10% variations in each process parameter terms.

B. Bounds on Target Clock Period

Here, we provide a theoretical result on the bounds of the target clock period, ϕ, which will be useful in the binary search procedure. Recall that the target clock period is set to the smallest value for which the graph $G=(V,E)$ has no positive cycle, and the arrival time of the sink node, $a[t]$, is no larger than ϕ. Let the delay of the i^{th} directed simple $s-t$ path in G be represented by $l_i = \psi_i - \kappa_i\phi$, where ψ_i denotes the sum of gate and wire delays along path i, and κ_i denotes the number of flip-flops in path i. Let C denote the number of directed cycles in G, and $\zeta_j = \xi_j - \sigma_j\phi$ denote the total delay of the j^{th} directed cycle, where ξ_j denotes the sum of gate and wire delays along cycle j. σ_j denotes the number of flip-flops in cycle j. Now ϕ is the smallest number that satisfies

$$\phi \geq a[t] = \max_{i=1,\ldots,K}\{\psi_i - \kappa_i\phi\} \qquad (2)$$

$$\zeta_j = \xi_j - \sigma_j\phi \leq 0 \qquad j = 1\ldots,C. \qquad (3)$$

Equivalently, the target clock period is given by

$$\phi = \max\left[\max_{i=1\ldots,K}\left\{\frac{\psi_i}{\kappa_i+1}\right\},\max_{j=1\ldots,C}\left\{\frac{\xi_j}{\sigma_j}\right\}\right] \qquad (4)$$

Recall that each gate and wire delay is a random variable, and hence, ψ_i and ξ_j, which are sums of gate and wire delays, are random variables. Let $\phi^l_d, \phi^m_d, \phi^u_d$ denote the values of ϕ obtained from (4) when all gate and wire delays are replaced by their lower bounds (best case), means (average case), and upper bounds (worst case), respectively. It is obvious that ϕ, which is a random variable, is in $[\phi^l_d, \phi^u_d]$ with probability one. Moreover, as we show in the theorem below, the mean of ϕ is bounded below by ϕ^m_d.

Theorem 3: Let $\mathbf{E}[\phi]$ be the mean (expected value) of ϕ. Then $\phi^l_d \leq \phi \leq \phi^u_d$ with probability one, and $\phi^m_d \leq \mathbf{E}[\phi]$.

Proof: Only the mean case needs a proof. Equation (4) implies

$$\phi \geq \frac{\psi_i}{\kappa_i + 1}, \quad i = 1, \ldots, K \qquad \phi \geq \frac{\xi_j}{\sigma_j}, \quad j = 1, \ldots, C$$

Since the ψ_i and ξ_j are the sum of gate and wire delays, and κ_i and σ_j are constants, the expected values of the right-hand-side terms of the inequalities above can be obtained by replacing all gate and wire delays by their means. As a result, taking the expectation on both sides of the inequalities shows that the mean of ϕ is greater than each right hand side term under a deterministic average case. This implies that the mean of ϕ is greater than the maximum of all such terms (ϕ_d^m). ■
Note that the bounds $\phi_d^l, \phi_d^m, \phi_d^u$ can be obtained by solving deterministic longest path problems.

IV. Experimental Results

Our algorithms are implemented in C++/STL, compiled with gcc v3.2.2, and run on a Pentium IV 2.4 GHz machine. The benchmark set consists of six big circuits from ISCAS89 and five big circuits from ITC99 suites.

Table I shows a comparison of the results obtained by using Monte Carlo simulation, the modified Bellman-Ford algorithm with the error bound (eSBF), and the modified Bellman-Ford with $k+1$ iterations (kSBF) on 8x8 dimension. Monte Carlo simulations are performed using 10,000 samples. We report the expectation (mean) and standard deviation (sigma) of the retiming delay distribution. We note that both eSBF and kSBF provide close results to Monte Carlo simulation results, especially in terms of the mean value. kSBF provides more accurate results than eSBF because, as pointed out in section II, eSBF can ignore some paths during its computation. Note that it is possible that eSBF can outperforms kSBF since kSBF may require more iterations than necessary. Because of the error arising from the analytical model, the higher the number of iterations, the more the error that is accumulated. However, most of the cases, kSBF is more accurate that eSBF. Both eSBF and kSBF substantially outperform Monte Carlo simulations in terms of runtime. The runtime reported is average runtime. Note that before using the bounds on target clock period, average runtime of kSBF is 9.4 hours. The bound can help reduce runtime substantially. Thus, we conclude kSBF, however, requires longer runtime than eSBF because of the higher the number of iterations required compared to eSBF. Figure 4 shows the comparison among Monte Carlo simulations, eSBF, and kSBF on s5378 benchmark. The solid, dotted, and dashed lines represent Monte Carlo simulation, eSBF, and kSBF, respectively. Results show that kSBF can provide a distribution similar to that of Monte Carlo simulations. In this case, it shows that kSBF is better since eSBF stops early and can ignore some paths.

V. Conclusions

In this paper, a Statistical Bellman-Ford (SBF) algorithm is proposed to compute the longest path length distribution for directed graphs with cycles. We use SBF in our Statistical Retiming-based Timing Analysis (SRTA), where SBF is used to check for the feasibility of a given target clock period distribution for retiming. Our Monte Carlo simulation validates the accuracy of our SRTA algorithm.

TABLE I
The comparison between deterministic Monte Carlo simulation, eSBF, and kSBF on 8x8 dimension

ckt	monte		eSBF		kSBF	
	mean	std.dev.	mean	std.dev.	mean	std.dev.
s5378	179.65	6.27	163.29	5.49	179.47	6.51
s9234	229.24	16.18	164.58	7.14	225.53	10.51
s13207	304.93	13.27	279.47	8.72	305.76	13.05
s15850	363.16	15.83	317.32	7.57	363.37	15.72
s38417	187.11	4.53	184.95	5.28	189.00	8.41
s38584	437.02	19.41	399.62	10.45	436.75	19.33
b14o	161.19	10.70	117.42	5.37	160.43	5.31
b15o	247.00	10.00	212.72	18.45	247.88	10.23
b20o	259.68	9.03	229.16	6.12	267.12	9.33
b21o	267.57	6.18	240.01	4.08	267.98	9.24
b22o	286.63	18.92	300.49	25.17	320.62	15.26
runtime	21 days		2.7 hours		3.7 hours	

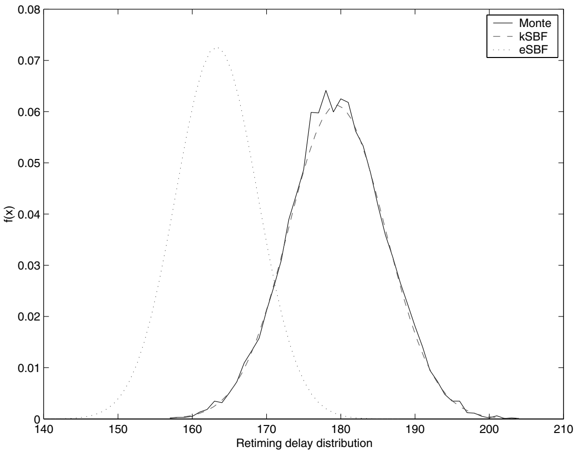

Fig. 4. The Distribution Comparison among Monte Carlo simulation (solid), eSBF (dotted), kSBF(dashed)

References

[1] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die paraemter fluctuations on ..." in *Proc. ISSCC*, 2001.
[2] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using ..." in *Proc. ICCAD*, 2003.
[3] A. Agarwal, V. Zolotov, and D. Blaauw, "Statistical timing analysis using bounds and selective enumeration," in *IEEE TCAD*, 2003.
[4] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based ..." in *Proc. DAC.*, 2004.
[5] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.
[6] T. G. Szymanski, "Computing optimal clock schedules," in *Proc. DAC.*, 1992.
[7] Y. S. F-R. Boyer, El M. Aboulhamd, "An efficient verification method for a class of multi-phase sequential circuits," in *ICECS*, 2000.
[8] J. Cong and S. K. Lim, "Retiming-based timing analysis with an application to ..." *IEEE TCAD*, vol. 23, no. 12, 2004.
[9] R. Durrett, *Probability: Theory and Examples*. Duxbury Press, 1995.
[10] P. Billingsley, *Probability and Measure*. John Wiley & Sons, 1995.
[11] Y.-Z. Liao and C. K. Wong, "An algorithm to compact a vlsi symbolic layout with mixed constraints," *IEEE TCAD*, 1983.
[12] R. Chen and H. Zhou, "Clock schedule verification under process variations," in *Proc. ICCAD*, 2004.
[13] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*. Prentice Hall Electronics, 2003.
[14] SIA, "National Techonology Roadmap for Semiconductors," 2003.
[15] C. Clark, "The greatest of a finite set of random variables," in *Operations Research*, 1961.
[16] C. Ababei and K. Bazargan, "Statistical timing driven partitioning for vlsi circuits," in *Proc. DATE*, 2002.