An Efficient Computation of Statistically Critical Sequential Paths Under Retiming

Mongkol Ekpanyapong†, Xin Zhao, and Sung Kyu Lim
†Intel Corporation, Folsom, California, USA
Georgia Institute of Technology, Atlanta, Georgia, USA
mongkol.ekpanyapong@intel.com, xzhao@ece.gatech.edu, limsk@ece.gatech.edu

***Abstract*— In this paper we present the Statistical Retiming-based Timing Analysis (SRTA) algorithm. The goal is to compute the timing slack distribution for the nodes in the timing graph and identify the statistically critical paths under retiming, which are the paths with a high probability of becoming timing-critical after retiming. SRTA enables the designers to perform circuit optimization on these paths to reduce the probability of them becoming timing bottleneck if the circuit is retimed as a post-process. We provide a comparison among static timing analysis (= STA), statistical timing analysis (= SSTA), retiming-based timing analysis (= RTA), and our statistical retiming-based timing analysis (SRTA). Our results show that the placement optimization based on SRTA achieves the best performance results.**

I. Introduction

Statistical timing analysis has become crucial to characterize signal transmission in the era of nano-scale device and interconnect. Compared to a large volume of works on statistical timing analysis for combinational circuits, there exist few works on how to deal with sequential circuits in the presence of process variations [1], [2], [3]. In this paper we present the Statistical Retiming-based Timing Analysis (SRTA) algorithm. The goal is to compute the timing slack distribution for the nodes in the timing graph and identify the statistically critical paths under retiming, which are the paths with a high probability of becoming timing-critical after retiming. SRTA enables the designers to perform circuit optimization on these paths to reduce the probability of them becoming timing bottleneck if the circuit is retimed as a post-process. We show that the final critical path delay distribution after retiming is the statistical maximum among all primary outputs and all feedback vertices. For this purpose, we introduce a new metric called Minimum Feasible Clock Period Distribution (MFCPD) to correctly capture the minimum possible delay distribution the subsequent retiming can achieve under process variations.

We integrate the SRTA algorithm into our mincut-based global placer to optimize statistical longest paths in sequential circuits. We perform the SRTA algorithm to compute statistical critical paths that consider retiming. Our mincut placer then tries to place these paths into a single partition. We provide a comparison among static timing analysis (= STA), statistical timing analysis (= SSTA), retiming-based timing analysis (= RTA), and our statistical retiming-based timing analysis (SRTA). Our results show that the placement optimization based on SRTA achieves the best performance results.

The remainder of the paper is organized as follows. Section II reviews the related works. Section III presents our Statistical Retiming-based Timing Analysis (SRTA) algorithm. We present the experimental results in Section IV and conclude in Section V.

II. Preliminaries

This section presents an overview of two existing works that our algorithm is based on, namely, Statistical Bellman-Ford (SBF) algorithm [3] and Retiming-based Timing Analysis (RTA) [4].

A. Statistical Bellman-Ford Algorithm

The Statistical Bellman-Ford (SBF) algorithm is recently presented in [3] to compute the longest path distribution under process variations. SBF 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. Unlike the deterministic Bellman-Ford algorithm that iterates longest path length update until no more update is possible, SBF performs exactly K iteration, where K is the the maximum number of backward edges along any cycle. The authors showed that in the presence of probability distribution functions, K iterations is enough to consider al simple paths in the timing graph and obtain highly accurate longest path distribution.

In SBF, depth-first search (DFS) is first called to identify all backward edges and to sort the nodes in a topological order (when cycles are ignored). For each backward node, we perform another 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. After the K is found, we initialize the arrival times of all nodes. Next, the relaxation step is called. In this case, the arrival time, node delay, and edge delay values are random variables. Thus, the statistical min/max and arithmetic operations are used to compute the new arrival time distribution. Once the computation of delay distribution is complete, we need to determine if the probability that a positive cycle exists is above a given threshold. The problem of detecting statistical positive cycles is complex since it involves

1-4244-0630-7/07/$20.00 ©2007 IEEE.

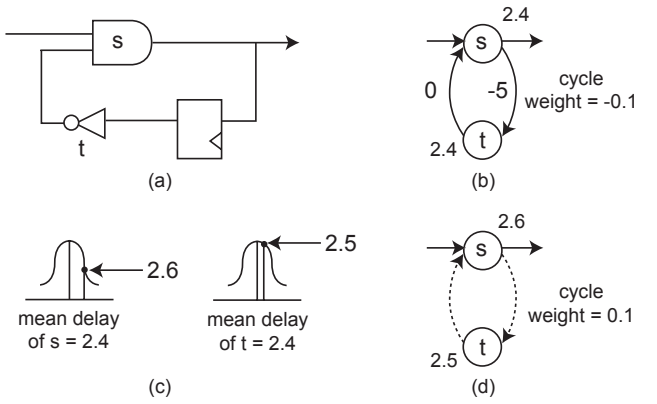

Fig. 1. Illustration of positive cycle formation due to process variation: (a) example circuit, (b) its retiming graph with a negative cycle, when target clock period $\phi = 5$. RTA assigns -5 as the weight of edge that contains a FF, (c) there exists a non-zero probability that the actual delay of gate s and t is above their mean, (d) positive cycle formed due to the process variation. Thus, the $a[s]$ value will never stop updating and Bellman-Ford will never converge.

the enumeration of all existing cycles. Thus, SBF uses a heuristic proposed in [5]. This heuristic considers only the cycles encountered during an initial run of DFS and uses them to approximate the probability of positive cycle existence.

B. Deterministic Retiming-based Timing Analysis

The Retiming-based Timing Analysis (RTA) [4] is proposed to calculate the timing slack *after* min-delay retiming. The basic idea is to compute the arrival and required time assuming that the FFs are optimally positioned in terms of performance, i.e., min-delay retiming is performed. The benefit of RTA is that this "retiming-based timing slack" can be exploited for more rigorous timing optimization during partitioning and placement [4]. In addition, RTA generates retiming as a byproduct via its Bellman-Ford approach, thereby eliminating the need for the time/memory-intensive ILP approach [6].

In RTA, the sequential circuit is modeled by a retiming graph [6], where FFs become the weights of directed edges connecting two neighboring gates. Due to the feedback loops involving FFs in the given sequential circuits, retiming graphs are usually cyclic. In addition, RTA uses another edge weight that combines FF-weight and a user-specified target clock period ϕ to compute the timing information, which may become positive or negative depending on ϕ. Thus, Bellman-Ford algorithm is used to compute longest paths (= arrival/required time) for the cyclic graph with negative cycles. In case the ϕ causes positive cycles to form, RTA determines that ϕ is not feasible. Finally, binary search is performed to compute minimum feasible ϕ using RTA as a feasibility checker.

Figure 1 shows how RTA fails to compute correct ϕ under process variation. RTA declares a target ϕ feasible if the resulting retiming graph does not contain a positive cycle. However, the probability of containing a positive cycle is still non-zero if the gate delay values are random variable as shown in Figure 1. Thus, a major challenge in the statistical extension of RTA is to consider the probability distribution functions (PDF) of the related delay values to accurately compute the PDF of minimum feasible clock distribution.

III. Statistical Retiming-based Timing Analysis

A. Statistical Timing Model

We use retiming graph [6] for statistical retiming-based timing analysis (SRTA). A retiming graph $G = (V, E, W)$ consists of a vertex set V that represents gates, a directed edge set E that represents signal directions in the given sequential netlist, and edge weight set W that represents the number of flip-flops (FFs) between the two end-vertices of each edge. G contains a source vertex v_{src} that connects to all PI vertices and a sink node v_{sink} that connects from all PO vertices. A retiming is a labeling of the vertices $r : V \rightarrow Z$, where Z is the set of integers. The weight of an edge $e = (u, v)$ after retiming is denoted by $w^r(e)$ and is given by $w(e(u, v)) + r(v) - r(u)$. The retiming label $r(v)$ for a vertex $v \in V$ represents the number of FFs moved from its output towards its inputs. A circuit is retimed to a delay ϕ by a retiming r if the following conditions are satisfied; (i) $w^r(e) \geq 0$, (ii) $w^r(p) \geq 1$ for each path p such that $d(p) > \phi$, where $w^r(p) = \sum_{e \in p} w^r(e)$. In this case, ϕ is called *feasible target delay*.

We use the following canonical first-order form to represent gate delay, wire delay, arrival time, require time, and slack distribution:

$$m + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta R \tag{1}$$

where m is the mean value. X_i denotes the n variation sources we consider, and ΔX_i represents the variation from its mean value caused by the variation source i. a_i is the sensitivity to the variation source i. ΔR is the variation of an independent random variable R from its mean value, and a_{n+1} is the sensitivity to R. We assume that the random variables X_i and R are Gaussian distribution $N(0, 1)$. We consider the following four sources of variation for the gate/wire delay distribution: transistor length (L_g), transistor width (W_g), wire width (W_i), and wire thickness (T_i).[1] We follow the suggestion in [7] to model the intra-die spatial correlation among the random variables. We divide the die into an $m \times n$ tile and assume perfect correlation among the devices and wires in the same tile. In addition, the correlation is high among the devices and wires from nearby tiles, and the correlation decreases as the distance among the tiles increases. We assume that correlation exists only among the same type of variation source. Lastly, we perform principal component analysis (PCA) as suggested in [7] to classify the coefficients into orthogonal terms so that each coefficient term is uncorrelated. Reconvergent correlation is also handled by PCA.

Our gate delay distribution is modeled as follows:

$$d(v) = d_m(v) + a_1 \Delta L_g^t(v) + a_2 \Delta W_g^t(v) \tag{2}$$

[1] It is straightforward to extend our formulation to consider other intra/inter-die variation sources.

where $d_m(v)$ is the mean delay of gate v. Assuming v is located in tile t, $\Delta L_g^t(v)$ and $\Delta W_g^t(v)$ are the variation of gate delay caused by the gate length and gate width variation in tile t, respectively. a_1 and a_2 are the sensitivity constants. Our wire delay distribution is modeled as follows:

$$d(e) = d_m(e) + \sum_{k \in T(e)} \left[b_1 \Delta W_i^k(e) + b_2 \Delta T_i^k(e) \right] \quad (3)$$

where $T(e)$ denotes the set of tiles that wire e traverses. b_1 and b_2 are the sensitivity constants.[2] $d_m(e)$ is the mean delay of wire e, $\Delta W_i^k(e)$ and $\Delta T_i^k(e)$ are the variation of wire delay caused by the wire width and wire thickness variation in tile k, respectively.

We define the statistical sequential arrival time (SSAT) of a node v in the retiming graph as follows;

$$l(v) = \max\{l(u) - \phi \cdot w(e) + d(e) + d(v) | e(u,v) \in E\} \quad (4)$$

$w(e)$ denotes the number of FFs along the edge e, and ϕ is the target clock period. $d(v)$ and $d(e)$ are the delay distribution variables shown in Equation (2) and (3).[3] $l(v)$ is computed via statistical addition and maximum operations and expressed in the canonical form shown in Equation (1). The intuition behind SSAT is that it represents the arrival time distribution of a node v assuming that the source-to-v path is optimally retimed to ϕ. In a similar way, we define the statistical sequential required time (SSRT) of a node v in the retiming graph as follows;

$$q(v) = \min\{q(u) + \phi \cdot w(e) - d(e) - d(v) | e(v,u) \in E\} \quad (5)$$

The statistical sequential slack (SSSK) of v, denoted $s(v)$, is given by $q(v) - l(v)$. The intuition behind SSSK is that it represents the timing slack distribution of a node v assuming that the input sequential circuit is optimally retimed to ϕ. We adopt the tightness probability calculation proposed in [8] to perform Gaussian approximation after the statistical maximum/minimum operation of two Gaussian distributions.

Once the SSSK values are computed, we define the "statistical ϵ-network" as the subset of nodes in the retiming graph and the edges connecting them, where the mean value of the SSSK is smaller than ϵ. Then, any path that shares the nodes and edges with the statistical ϵ-network is timing critical. Note that a higher ϵ value means more timing critical paths to consider during circuit optimization.[4]

B. Statistical RTA Algorithm

Note that the retiming graph G introduced in Section III-A is cyclic because of the FFs in the given sequential circuit. In addition, the computation of SSAT and SSRT may involve negative or positive weighted cycles depending on the random variables $d(e)$ and $d(v)$ as well as the constants $w(e)$ and ϕ used in Equation (4) and (5). This calls for Bellman-Ford approach to compute longest path distributions under negative cycles. This calls for statistical Bellman-Ford (SBF) discussed in Section II-A to handle randome variables. In the meantime, SRTA performs min-delay retiming (= retiming G so that the clock period becomes ϕ) while performing the statistical timing analysis. Lastly, our SRTA also determines if the target clock period ϕ is feasible or not, i.e., whether G can be retimed to ϕ with sufficiently high probability.

Statistical Retiming-based Timing Analysis
input: retiming graph R, target delay ϕ
output: $l(v)$ pdfs, $q(v)$ pdfs, and $r(v)$ for all $v \in V$

```
1.  compute backward nodes and calculate K;
2.  for (each vertex v ∈ V)
3.      l(v) = −∞, q(v) = ∞, r(v) = 0;
4.  l(v_src) = 0, q(v_sink) = φ;
5.  for (i = 1 to K + 1)
6.      for (each vertex v ∈ V)
7.          t_a = statistical sequential arrival time of v;
8.          t_r = statistical sequential required time of v;
9.          if (t_a > l(v))
10.             l(v) = t_a;
11.         if (t_r < q(v))
12.             q(v) = t_r;
13. compute MFCPD;
14. P(cycle) ← check_pos_cycle();
15. if (P(cycle) ≤ P_acceptable)
16.     return (FALSE);
17. return (TRUE);
```

Fig. 2. Description of statistical retiming-based timing analysis (SRTA) algorithm that computes the statistical sequential arrival time distribution $l(v)$, statistical sequential required time distribution $q(v)$, and retiming $r(v)$ for all $v \in V$. SRTA also determines the feasibility of the target delay ϕ.

SSAT and statistical retiming are closely related. In fact, the computation of SSAT and statistical retiming can be performed at the same time. Consider a path p that starts from a PI u and ends at vertex v. If we want to retime p to satisfy the time constraint ϕ, there must be at least $\lceil l(p)/\phi \rceil - 1$ FFs on p. Since there exists $w(p)$ FFs on p, we can set the retiming value $r(v)$ as $\lceil l(p)/\phi \rceil - 1 - w(p)$. Thus, $r(v) = \lceil l(p)/\phi \rceil - 1 - w(p)$. After rewriting, we get $r(v) = \lceil l(v)/\phi \rceil - 1$. Thus, our SRTA uses a feasible target delay ϕ to compute SSAT, SSRT, and retiming all at the same time. In SRTA, SSAT for all PIs are set to zero while all others are set to $-\infty$. SSRT for all POs are set to ϕ while all others are set to ∞. Then, we can iteratively update SSAT and SSRT until they converge to their maximum and minimum values, respectively.

Figure 2 shows the description of our SRTA algorithm. We first compute K, the maximum number of connected backward nodes as discussed in Section II-A. The purpose is to perform $K + 1$ iterations of SSAT and SSRT updates during SRTA (line 5). Next, the initialization of SSAT $l(v)$, SSRT $q(v)$, and retiming $r(v)$ for each vertex is done (line 2-4). During

[2]In this article, $a_1 = 0.099$, $a_2 = -0.099$, $b_1 = 0.019$, and $b_2 = -0.133$. These sensitivity values are computed based on the assumption that each process parameter is varied by 10%.

[3]In case an edge contains FFs, i.e., $w(e) > 0$, we extend the edge delay distribution Equation (3) to consider the FFs on the edge. In this case, the delay change from the width and length variation of each FF is added to the overall edge delay distribution.

[4]Our empirical choice of ϵ is 5% of the maximum mean SSSK among all nodes.

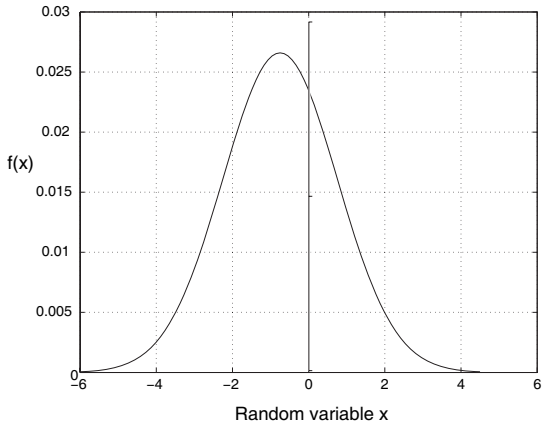

Fig. 3. Positive cycle

each iteration we visit each node and compute new SSAT and SSRT (line 7-8) using statistical min/max and arithmetic operations. If the new SSAT is larger than the existing SSAT, we update the existing SSAT (line 9-10). We update SSRT in a similar way (line 11-12). In addition, we update the Minimum Feasible Clock Period Distribution (MFCPD) (to be discussed in Section III-C) (line 13). The theoretical runtime of SRTA algorithm is $O(n^2)$ since $K = O(|V|)$ in the worst case. K, however, is rarely close to $|V|$ in VLSI circuits typically as shown in Table II in Section IV, making the practical runtime of SRTA to be linear.

In deterministic RTA, Bellman-Ford terminates immediately when the sequential arrival time of the sink node exceeds the target clock period. This condition is met when there exists a positive cycle in the retiming graph. If so, RTA determines that the given ϕ is not feasible. This termination condition still holds in the statistical case that if the expectation of the summation of gate and wire delay over the cycle is positive, the arrival time of the sink can exceed the target clock period. This condition can be used for the algorithm to terminate early. However, one might have to be aware that if the expectation of the summation of gate and wire delay over a cycle is negative, it does not necessarily mean that there exists no positive cycle. An illustration is shown in Figure 3. A cycle could be negative in terms of the mean value, but there is still a high probability that a positive cycle exists. Therefore, our SRTA performs $K+1$ iterations regardless of the delay distribution changes along the cycles to fully account for all simple paths as discussed in Section II-A. The last step in SRTA (line 14-17) is to explicitly detect positive cycles using the method discussed in Section II-A.

C. Target Delay Distribution

The deterministic RTA is performed under a given target clock period ϕ. In case ϕ is feasible, the weight of all cycles in the retiming graph G become negative, and thus the Bellman-Ford based RTA terminates and computes the sequential timing slack values successfully. In addition, the circuit is guaranteed to be retimed to ϕ, i.e., a subsequent retiming is guaranteed to reduce the clock period to ϕ. In case the min-delay retiming is desired, RTA performs binary search to find the minimum possible ϕ. We note that the following relation holds:

$$\phi = \max\{max\_cycle, SAT(v_{sink})\} \tag{6}$$

where max_cycle denotes the maximum delay among all cycles in G, and $SAT(v_{sink})$ is the sequential arrival time at the sink node. This means that the most critical path we obtain after a subsequent retiming may include a cycle or not depending on the circuit structure. This relation suggests another way of computing the minimum ϕ, where we compute the max_cycle and $SAT(v_{sink})$ instead of performing binary search. The computation of $SAT(v_{sink})$ is straightforward once max_cycle is known—a single run of RTA is enough since we just use the ϕ from max_cycle. However, the computation of max_cycle requires us to examine *all* cycles in the circuit. In this case, the authors in [9] suggest that the Howard's algorithm [10] be used for this purpose. However, the runtime overhead for the binary search-based approach is minimal. In addition, we do not need a separate step to compute max_cycle since the minimum ϕ and its $SAT(v_{sink})$ are directly computed.

A similar argument applies to the statistical case. We define the following new random variable:

Definition 1: The Minimum Feasible Clock Period Distribution (MFCPD) of a given sequential circuit is the minimum possible delay distribution the subsequent retiming can achieve, where the gate, FF, and interconnect delay values are random variables.

Then, the following relation holds:

$$MFCPD = \max\{max\_cycle\_pdf, l(v_{sink})\} \tag{7}$$

where max_cycle_pdf denotes the delay distribution of the longest cycle, and $l(v_{sink})$ is the SSAT of the sink node as defined in Equation (4). It is important to note the difference between MFCPD and ϕ, where MFCPD is a random variable and ϕ is a constant. In SRTA, we specify a constant target clock period ϕ and compute its corresponding MFCPD. The feasibility checking for ϕ is not done by comparing to $l(v_{sink})$ since SRTA performs $K+1$ iteration regardless of the convergence of the SSAT/SSRT values. Instead, a separate step to detect positive cycle is performed as explained in Section II-A.

D. Comparison Among Various Timing Analysis

Table I shows a comparison among static timing analysis (STA), statistical static timing analysis (SSTA), retiming-based timing analysis (RTA), and our statistical retiming-based timing analysis (SRTA). STA has been widely used during timing-driven optimization and validation mainly for its simplicity and efficiency. The main goal is to identify timing-critical nets by computing the timing slack values of the nodes in an acyclic directed graph that represents a sequential circuit. The timing graph is acyclic since the FFs are removed from the circuit so that topological ordering is well defined. SSAT is a statistical extension of STA, where the delay values of

TABLE I

COMPARISON AMONG STATIC TIMING ANALYSIS (STA), STATISTICAL STATIC TIMING ANALYSIS (SSTA), RETIMING-BASED TIMING ANALYSIS (RTA), AND OUR STATISTICAL RETIMING-BASED TIMING ANALYSIS (SRTA).

	STA	SSTA	RTA	SRTA
goal	computation of deterministic timing slack values in combinational circuits	computation of statistical timing slack distribution in combinational circuits	computation of deterministic timing slack values after retiming in sequential circuits	computation of statistical timing slack distribution after retiming in sequential circuits
delay values	deterministic	statistical distribution	deterministic	statistical distribution
retiming	no	no	yes	yes
circuit graph	directed acyclic graph, FF removed	directed acyclic graph, FF removed	cyclic retiming graph, FF becomes edge weight	cyclic retiming graph, FF becomes edge weight
basic algorithm	topological sort	topological sort, statistical min/max and arithmetic operation	Bellman-Ford	Bellman-Ford, statistical min/max and arithmetic operation
approach	visit nodes in forward (backward) topological order to compute arrival (require) time.	visit nodes in forward (backward) topological order to compute and propagate statistical arrival (require) time distribution.	compute longest path for cyclic graph with negative edge weights to compute timing slack after retiming.	compute "statistical longest path distribution" and "slack distribution after retiming" with statistical Bellman-Ford algorithm
complexity	$O(n)$	$O(n)$	$O(n^2)$, $O(k \cdot n)$ in practice	$O(n^2)$, $O(k \cdot n)$ in practice
advantage	simple and fastest	handle statistical analysis	model FFs and predict delay after retiming	perform statistical retiming and report delay distribution after retiming
disadvantage	can't handle retiming nor statistical variations	slow and no retiming consideration	slow and can't handle statistical variations	slow

the nodes and edges in the acyclic circuit graph are given as probability distribution function (pdf). The goal is to compute the timing slack pdfs for the nodes and identify "statistically critical paths", which are the paths with a high probability of becoming timing-critical. A huge volume of works on SSTA has been proposed recently, and its application on circuit optimization is currently begin investigated as discussed in Section I.

RTA has been proposed to compute the timing slack values after retiming. Several works [11], [4] have demonstrated the benefit of performing layout optimization using these "timing slacks after retiming" compared to the traditional timing slack values without any retiming. FFs are modeled as edge weights as discussed in Section III-A and introduce cycles in the circuit graph. Depending on the target clock period for retiming, the cycles may become positive or negative weight, and Bellman-Ford algorithm is used to test the feasibility of the given target clock period and compute timing slack values. SRTA is a statistical extension of RTA, where delay distributions are computed using statistical Bellman-Ford algorithm. The goal is to compute the timing slack pdfs for the nodes and identify "statistically critical paths under retiming", which are the paths with a high probability of becoming timing-critical after retiming. Circuit optimization on these paths will reduce the probability of them becoming timing bottleneck if the circuit is retimed as a post-process.

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 [12] and five big circuits from ITC99 [13] suites. We do not use the ISPD98 benchmark since it does not contain signal direction information. We assume 10% variations in each process parameter terms as discussed in Section III-A. In this paper, wire delay is computed based on Elmore delay model. Since the actual wire distance is not known until routing, an analytical model is used to estimate the wirelength [14].

TABLE II

BENCHMARK CIRCUIT CHARACTERISTICS. $K+1$ DENOTES THE MAXIMUM NUMBER OF BACKWARD NODES ALONG ANY CYCLE. B-NODES DENOTES THE TOTAL NUMBER OF BACKWARD NODES.

ckt	gate	PI	PO	FF	$K+1$	b-node
s5378	2828	36	49	163	76	95
s9234	5597	36	39	211	239	354
s13207	8027	31	121	669	510	637
s15850	9786	14	87	597	495	699
s38417	22397	28	106	1636	1444	1660
s38584	19407	12	278	1452	1860	2054
b14o	5401	32	299	245	451	616
b15o	7092	37	519	449	988	1408
b20o	11979	32	512	490	1486	2197
b21o	12156	32	512	490	1511	2209
b22o	17351	32	725	703	1870	2770

Table II shows the characteristics of the benchmark circuits we used. We report $K+1$, the maximum number of backward nodes along any cycle. This is also the number of the iterations used in Statistical Bellman-Ford Algorithm (SBF) discussed in Section II-A. We note that this value correlates with the size of the circuits. Unlike the deterministic Bellman-Ford where the number of iteration depends on whether there is any update on the delay values begin computed, SBF enforces $K+1$ iteration regardless of the changes on the delay value distribution. This is intended to make sure all simple paths are considered during the delay distribution updates as discussed in Section II-A.

Table III shows how the minimum feasible clock period distribution (MFCPD) is computed for each circuit. The

TABLE III

MINIMUM FEASIBLE CLOCK PERIOD DISTRIBUTION (MFCPD) RESULTS. THE MAXIMUM BETWEEN MAX-CYCLE AND SINK-SSAT IS SHOWN IN BOLD TYPE.

ckt	final MFCPD	max cycle	sink SSAT
s5378	199.00	62.75	**199.00**
s9234	257.03	225.08	**257.03**
s13207	344.87	281.36	**344.87**
s15850	410.53	276.76	**410.53**
s38417	214.23	**214.23**	152.89
s38584	494.73	394.07	**494.73**
b14o	176.36	137.44	**176.36**
b15o	278.53	**278.53**	57.56
b20o	295.11	**295.11**	117.14
b21o	295.70	**295.70**	275.72
b22o	366.36	318.8	**366.36**

TABLE IV

TIMING ANALYSIS RESULTS FOR STATIC TIMING ANALYSIS (= STA), STATISTICAL TIMING ANALYSIS (= SSTA), RETIMING-BASED TIMING ANALYSIS (= RTA), AND OUR STATISTICAL RETIMING-BASED TIMING ANALYSIS (SRTA).

	without retiming		with retiming	
ckt	STA	SSTA	RTA	SRTA
s5378	232.00	218.96	212.00	199.00
s9234	407.00	379.05	287.00	257.03
s13207	426.00	403.43	364.00	344.87
s15850	575.00	532.59	435.00	410.53
s38417	390.00	363.29	217.00	214.23
s38584	576.00	554.21	515.00	494.73
b14o	375.00	351.86	195.00	176.36
b15o	352.00	327.57	292.00	278.53
b20o	516.00	484.09	318.00	295.11
b21o	518.00	479.66	320.00	295.70
b22o	626.00	605.99	375.00	366.36
avg.	1.00	0.94	0.71	0.67

retime-delay column shows the deterministic delay value after retiming. The max-cycle column shows the worst value (= mean plus 3 sigma) of the delay distribution of the longest cycle, and sink-SSAT is the worst value of the statistical sequential arrival time distribution of the sink node. According to Equation (7), MFCPD is the maximum between max-cycle and sink-SSAT. We observe that cycles are involved with the most critical paths in half the benchmarks (such as s38417, b15o, b20o, b21o), whereas the other half contain acyclic critical paths. Thus, we conclude that it is important to look at both the cycle delay distribution and sink node delay distribution to compute accurate MFCPD.

Table IV shows a comparison among STA, SSTA, RTA, and SRT. We integrate STA/SSTA/RTA/SRTA into our mincut-based global placer to optimize longest paths in sequential circuits. Our placer performs multi-level bipartitioning recursively until the desired number of partitions is obtained. In this case, we perform STA/SSTA/RTA/SRTA to compute the $epsilon$-network discussed in Section III-A to identify the timing-critical paths. Our placer then tries to place these paths into a single partition. The goal is to maximize the performance based on the 8×8 global placement results. Given an 8×8 global placement result, we report the worst case deterministic delay values for STA and RTA and the mean plus 3 sigma values for SSTA and SRTA. First, we note that retiming reduces the delay results significantly (about 30% on average) in both deterministic and statistical cases. This highlights the advantage of retiming-based timing analysis algorithms (= RTA and SRTA). Second, we note that the placement optimization base on statistical timers (SSTA and SRTA) achieves consistently better results than deterministic timers (STA and RTA).

V. CONCLUSIONS

We presented an efficient algorithm named Statistical Retiming-based Timing Analysis (SRTA) to compute the statistically critical paths under retiming. The goal is to find the paths with a high probability of becoming timing-critical after retiming. SRTA uses Statistical Bellman-Ford algorithm to check for the feasibility of a given target clock period distribution under retiming. Our related experiments show that the placement optimization based on SRTA achieves better performance results compared to other well-known timing analyzers.

REFERENCES

[1] L. Zhang, Y. Hu, and C. Chen, "Statistical Timing Analysis in Sequential Circuit for On Chip Global Interconnect Pipelining," in *Proc. ACM Design Automation Conf.*, 2004.

[2] P. M, C. Chu, and H. Zhou, "Timing yield estimation using statistical static timing analysis," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2005.

[3] M. Ekpanyapong, T. Watewai, and S. K. Lim, "Statistical Bellman-Ford Algorithm With An Application to Statistical Retiming," in *Proc. Asia and South Pacific Design Automation Conf.*, 2006.

[4] J. Cong and S. K. Lim, "Retiming-based timing analysis with an application to mincut-based global placement," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 12, pp. 1684–1692, 2004.

[5] R. Chen and H. Zhou, "Clock schedule verification under process variations," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2004.

[6] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, pp. 5–35, 1991.

[7] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2003.

[8] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. ACM Design Automation Conf.*, 2004.

[9] A. P. Hurst, P. Chong, and A. Kuehlmann, "Physical placement driven by sequential timing analysis," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2004.

[10] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. McGettrick, and J.-P. Quadrat, "Numerical computation of spectral elements in max-plus algebra," in *in Proceedings of the IFAC Conference on System Structure and Control*, 1998.

[11] P. Pan, A. K. Karandikar, and C. L. Liu, "Optimal clock period clustering for sequential circuits with retiming," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 6, pp. 489–498, 1998.

[12] ISCAS89, "The ISCAS 1989 benchmark suite." [Online]. Available: http://www.cbl.ncsu.edu

[13] ITC99, "The ITC 1999 benchmark suite." [Online]. Available: http://www.cad.polito.it/tools/9.html

[14] P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl, "Prediction of Net-Length Distribution for Global Interconnects in a Heterogeneous System-on-a-Chip," *IEEE Trans. on VLSI Systems*, vol. 8, no. 6, pp. 649–659, December 2000.