

Statistical Timing Based Optimization Using Gate Sizing

Aseem Agarwal, Kaviraj Chopra, David Blaauw, *Vladimir Zolotov

University of Michigan, Ann Arbor, MI
*IBM T.J.Watson Research Center, Yorktown heights, NY

Abstract

The increased dominance of intra-die process variations has motivated the field of Statistical Static Timing Analysis (SSTA) and has raised the need for SSTA-based circuit optimization. In this paper, we propose a new sensitivity based, statistical gate sizing method. Since brute-force computation of the change in circuit delay distribution to gate size change is computationally expensive, we propose an efficient and exact pruning algorithm. The pruning algorithm is based on a novel theory of perturbation bounds which are shown to decrease as they propagate through the circuit. This allows pruning of gate sensitivities without complete propagation of their perturbations. We apply our proposed optimization algorithm to ISCAS benchmark circuits and demonstrate the accuracy and efficiency of the proposed method. Our results show an improvement of up to 10.5% in the 99-percentile circuit delay for the same circuit area, using the proposed statistical optimizer and an average run time improvement of up to 20x compared to the brute-force approach.

1 Introduction

Static Timing Analysis (STA) has been the mainstay of performance verification for the past two decades. Traditionally, process variation has been addressed in STA using corner-based analysis where all gates are assumed to operate at a worst-, typical- or best-case condition and within-die variability is not modeled. However, in the nanometer regime, within-die variation has become a substantial portion of the overall variability and corner-based STA suffers from significant inaccuracy. This has given rise to a new field of statistical timing analysis known as SSTA.

In SSTA, the circuit delay is considered a random variable and the objective of SSTA is to compute its probability distribution. So-called *block-based* SSTA approaches [1-6] are analogous to STA in that they propagate arrival times through the circuit. As the arrival times traverse gates, the delay of the gate is added to the arrival time and a maximum arrival time is selected when multiple arrival times converge at a gate. In SSTA, the arrival times also become random variables, and hence, the addition and maximum operations of STA are replaced by convolution and a statistical maximum, respectively. Like STA, they require a single pass of the circuit to compute the circuit delay distribution. From the CDF (cumulative distribution function) of the circuit delay, the user is then able to obtain the percentage of fabricated dies which meets a certain delay requirement, or conversely, the expected performance for a particular yield. In turn, gate or transistor sizing approaches should consider such metrics for their objective function and should perform their optimization in a statistically aware manner.

SSTA-based optimization can significantly improve the yield of a design compared to deterministic optimization. This is due to the fact that deterministic optimization tends to create a so-called “wall” of critical and nearly critical paths, as shown in Figure 1a, since there is no incentive to improve path delays that are not critical. All critical paths can affect the circuit delay due to delay variability, and hence, a balanced circuit with many near-critical paths is highly susceptible to process variations. This is illustrated in Figure 1(a) and (b), where a balanced and unbalanced path distributions are shown with their associated circuit delay distributions. While both path distributions have the same deterministic circuit delay, the unbalanced distribution results in a better statistical circuit delay since it has fewer near-critical paths. Hence, deterministic optimization can actually worsen the true statistical circuit delay. Furthermore, deterministic optimization cannot correctly choose between paths with low mean / high standard deviation and paths with high mean / low standard deviation, as shown in Figure 1c. While deter-

ministic optimization would choose the path with higher mean delay, the path with higher standard deviation clearly would have greater impact on the 99-percentile circuit delay. Hence, deterministic optimization works in the wrong direction due to lack of a correct statistical objective function.

Historically, much research has focussed on deterministic gate/transistor sizing [7-10], while more recently, a number of statistical optimization algorithms have also been proposed in [11-14]. In [11] it is shown that by performing deterministic optimization, it is possible to degrade the performance of the die statistically due to the creation of a timing wall (Figure 1). Hence, in [12] the authors propose a method to avoid the formation of such a wall by purposely improving non-critical paths in the deterministic optimization. In [13] and [14], the statistical optimization problem has been considered as a nonlinear programming problem. Delays are considered to be gaussian and approximations are used for computing the statistical maximum. In [14], a heuristic approach is proposed using the concept of statistically ‘undominated’ paths. All these approaches, however, are heuristic in nature and suffer from a lack of true sensitivity information which expresses the change of the circuit delay CDF due to a unit change in a gate size in the circuit.

In this paper, we therefore propose a new sensitivity based, statistical gate sizing algorithm. We use a coordinate descent algorithm where in each iteration the gate with the highest sensitivity is sized up. We show that such a statistically aware optimization can improve the 99-percentile delay by up to 10.5% over that obtained with traditional deterministic optimization. It should be noted, as shown in Figure 3, that a sizing change can impact both the mean and the shape of the circuit delay CDF. Depending on the objective specified by the user, the CDF perturbation can be evaluated in a number of ways. In this paper, we consider as objective the CDF delay at the 99% probability or confidence point, as shown in Figure 3. Hence, the computed sensitivity is measured as the change in the 99-percentile delay of the circuit delay CDF. However, other objective functions could be equally well supported by the proposed framework.

Since brute-force computation of such CDF perturbation sensitivities is extremely expensive, a key contribution of this paper is an efficient and exact pruning algorithm that allows for identification of the most sensitive gate in the circuit. Our pruning approach is based on a proposed theory of bounds on CDF perturbations due to sizing. We establish the useful property that these perturbation bounds can only diminish as the arrival time perturbations are propagated through the circuit using convolutions and maximum operations. Based on this property, we propose a

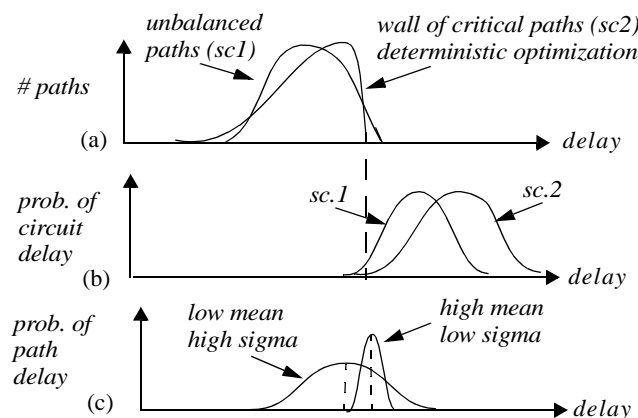


Figure 1. a) distribution of paths in a circuit b) corresponding circuit delay PDFs c) path delay PDFs

pruning algorithm for finding the highest sensitivity gate in a sizing iteration, without complete propagation of the perturbed arrival times for all gates. We perform an iterative propagation of perturbed arrival times for a pruned set of gates by maintaining their so-called *perturbation fronts* (defined later). We test our approach on a number of benchmark circuits, and demonstrate up to 20 times faster average runtimes than brute force sensitivity computation based optimization, without loss in accuracy.

The remainder of this paper is organized as follows. In Section 2, we present our modeling assumptions, along with the problem formulation, basic definitions and delay model. In Section 3, we present our approach for sensitivity computation and optimization. In Section 4, we present our results and compare deterministic optimization with brute force statistical and our proposed accelerated approach. Finally, in Section 5 we draw our conclusions.

2 Problem Formulation

In this section we define our modeling assumption and our SSTA approach. We also formulate the statistical optimization problem and present basic definitions and the delay model.

Gate delay variability is composed of two primary components: inter-die (between-die) variability and intra-die (within-die) variability. Inter-die variability expresses the change in gate delay from one die to the next and has traditionally been modeled using corner analysis with reasonable accuracy. The main focus of SSTA has therefore been on intra-die variability, which corner-based analysis is unable to model. Hence, similar to the optimization approaches proposed in [13,14] we focus on intra-die variability in this paper.

One of the difficulties in SSTA arises from reconvergent circuit structures, which results in correlations between arrival times. In [2,4], it was shown that the worst-case runtime for exact computation of the circuit delay CDF in a reconvergent circuit is exponential with circuit size, making its computation impractical. However, in [4] a simple method where these correlations are ignored was shown to result in an upper bound on the circuit delay CDF and hence a conservative analysis. Furthermore, it was shown that these bounds are typically tight and give a reasonably close approximation of the exact circuit delay CDF while their computation runtime is linear with circuit size. In this paper, we therefore use the bounds proposed in [4] for computation of the circuit delay CDF. It is important to note that the optimization objective is defined on this bound of the circuit delay CDF and not on the exact circuit delay CDF itself, since this would lead to prohibitive runtimes. However, we show in the result section that the optimization of the bounds, as performed by our method, results in nearly equivalent improvement of the exact circuit delay, as verified using Monte-Carlo simulation.

In addition to reconvergent circuit structures, spatial correlation of the gate delays can also give rise to correlation of arrival times [6]. However, similar to previous optimization methods [13,14], our optimization approach does not model such correlations at this time, although the proposed methods form a basis from which such correlations can be incorporated.

In a statistical timing paradigm, the delay of the circuit is a random variable. As a result, one needs to determine an appropriate objective function for optimization, defined on the distribution of the circuit delay random variable. The objective function can be the cost, obtained by applying a cost function to the shape of the delay distribution, or the delay at a particular confidence point. Since we use propagation of discretized arrival time PDFs, and not merely the statistical measures such as mean and variance, we obtain the entire shape of the circuit delay distribution. Hence, the proposed framework can support a wide range of cost functions as optimization objectives. For example, consider a linear cost function as shown in Figure 2.

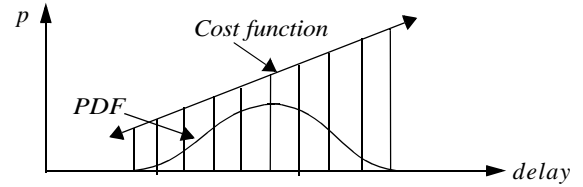


Figure 2. Cost function

The cost function increases linearly with an increase in delay. An objective function which utilizes this linear cost function can be obtained by computing the weighted sum of the cost function with respect to the corresponding probabilities of the circuit delay PDF. Such an objective function for circuit optimization can truly capture the expected cost. Our goal is to minimize this objective function by performing gate sizing. For simplicity of explanation, however, we choose as our optimization objective the p -percentile point $T(p)$ of the delay distribution. In our experiments, we choose p to be the 99-percentile point and minimize delay for that point, as shown in Figure 3.

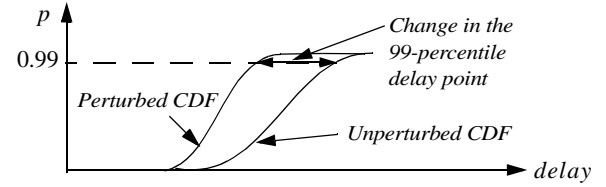


Figure 3. Optimization objective (99-percentile delay point)

2.1 Delay Model

We use a simple delay model for our experiments, similar to that used in [8], based on the logic effort model. In this model, the pin-to-pin delay (edge delay) of a gate is defined as

$$D_e = D_{int} + K \times C_{load} / C_{cell}, \quad (\text{EQ 1})$$

where, D_{int} is a constant, intrinsic delay due to cell-internal capacitances, C_{load} is the total load capacitance, K is a constant for the standard cell and C_{cell} is the total capacitance of the standard cell. We determine these constants for all the standard cells in our synthesis library for our experiments. For the statistical modeling of these delays we assume that the standard deviation is a fixed percentage of the nominal delay, although our method is not restricted to this model.

3 Proposed Optimization Approach

We first present in Section 3.1 the straightforward approach to performing statistical optimization using sensitivities. In Section 3.2, we develop novel properties of sensitivity propagation based on which an efficient pruning algorithm is presented in Section 3.3.

3.1 Straightforward Approach

Our brute force statistical algorithm is similar in structure to a deterministic coordinate descent algorithm, such as TILOS [7]. The deterministic optimization is sensitivity based and iteratively minimizes the circuit delay starting from a minimum size implementation. During each iteration, the gate with the maximum sensitivity is identified and sized up. If the optimization is deterministic in nature, any gate that improves the circuit delay by being sized up, must lie on the critical path of the circuit and hence, the sensitivity computation can be restricted to only those gates on the critical path.

However, in optimizing statistical circuit delay, there may be no single longest path, because the circuit delay PDF is a combination of all the path delay PDFs. Hence, statistical sensitivity needs to be computed for all gates in the circuit making a sensitivity based statistical optimization significantly more computationally demanding. According to the objective function defined in Section 2, the statistical sensitivity is the change in the p -percentile point of the circuit delay CDF due to the upsizing of a gate. This means that the perturbation of sizing a gate must be propa-

gated to the sink node in order to calculate the sensitivity of the gate. This necessitates a statistical timing analysis run for each gate in the circuit at every sizing step of the algorithm with a runtime complexity of $O(N \cdot E)$ for every sizing iteration, where N is the number of nodes and E is the number of edges of graph G . This results in unacceptable runtimes. Therefore, we propose an approach where the gate with maximum sensitivity can be identified without explicit propagation of perturbed arrival time CDFs for each gate.

3.2 Properties of sensitivity propagation

To allow for pruning of sensitivities, we now introduce the following useful definitions and properties of sensitivity propagation.

As shown in Figure 6, A_i is the CDF of the arrival time random variable at node i and A'_i is the corresponding perturbed CDF obtained by scaling up a gate. Their PDFs are denoted by a_i and a'_i , respectively.

We define the difference in the p -percentile point of the CDFs A_i and A'_i as $\delta_i(p) = T(A_i, p) - T(A'_i, p)$. The maximum difference over all p is given by $\Delta_i = \max_p \delta_i(p)$.

First, we assume that the perturbed CDF A'_i has the exact same shape as the unperturbed CDF A_i and differs from A_i only by a constant shift in time, i.e. $A'_i(t) = A_i(t - \Delta_i)$ and also $a'_i(t) = a_i(t - \Delta_i)$. This is assumed to be true for all perturbed CDFs. Under this assumption, we prove in Theorems 1 through 3 that the maximum difference Δ_i between the perturbed and unperturbed CDFs at a node can not increase as the perturbed CDFs are propagated through the circuit using convolution and statistical maximum. This property is useful in bounding the difference between the perturbed and unperturbed CDFs at the sink node, without complete propagation of the gate's perturbed CDF to the sink node. However, a change in a gate size often affects not only the mean of the gate delay, but also the shape of the CDF. Therefore, we show that it is possible to construct a lower bound on the perturbed CDF, such that the shape of this lower bound is identical to the unperturbed CDF, as illustrated by CDF B'_i in Figure 6. We then apply Theorems 1 through 3 to this lower bound on the perturbed CDF and show that these theorems are true for any shape perturbation. Finally, we show the use of these theorems to effectively prune out the propagation of perturbed CDFs.

Theorem 1. Convolution operation: Consider the timing graph shown in Figure 4a. Let a_i and a'_i be the original and the perturbed PDF at node i such that $a'_i(t) = a_i(t - \Delta_i)$ and let d_e be the delay PDF of edge e . If the arrival time PDF a_j and the perturbed a'_j at node j are given by $a_j = \text{Conv}(a_i, d_e)$ and $a'_j = \text{Conv}(a'_i, d_e)$, then $\Delta_j = \Delta_i$.

Proof : The proof is clear from the definition of convolution assuming independence. It is omitted for brevity.

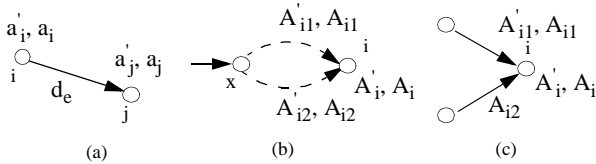


Figure 4. Timing graphs

In the following two theorems, we show that a similar property to that of Theorem 1 holds for the maximum operation. As previously mentioned, we assume that correlation of the arrival times due to reconvergent fanout can be ignored for the maximum operation and hence, the theorems are defined for an upper bound of the exact arrival time CDF [4].

Theorem 2. Max operation with multiple perturbed arrival times:

Consider a node i in the probabilistic timing graph shown in Figure 4b. Let A_{i1} and A_{i2} be the arrival time CDFs of two fanin subgraphs incident at node i . Let A'_{i1} and A'_{i2} be the perturbed CDFs obtained by scaling a single gate x that is common to the fanin cones of A_{i1} and

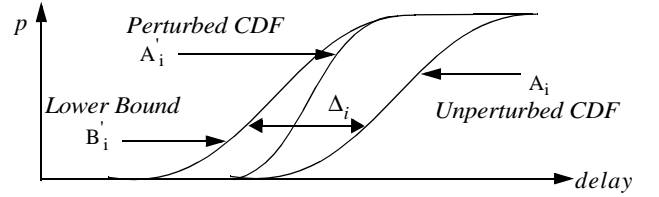


Figure 6. Arrival time CDFs at node i

A_{i2} . If the arrival time CDF A_i and perturbed CDF A'_i at node i are given by, $A_i = \max(A_{i1}, A_{i2})$ and $A'_i = \max(A'_{i1}, A'_{i2})$ respectively, then $\Delta_i \leq \max(\Delta_{i1}, \Delta_{i2})$.

Proof : We consider two cases.

case 1: Consider $\Delta_{i1} = \Delta_{i2}$.

By definition of maximum operation assuming independence,

$$A_i(t) = A_{i1}(t) \cdot A_{i2}(t) \quad (\text{EQ 2})$$

$$\text{and } A'_i(t - \Delta_{i1}) = A'_{i1}(t - \Delta_{i1}) \cdot A'_{i2}(t - \Delta_{i1}) \quad (\text{EQ 3})$$

R.H.S. of EQ2 and EQ3 being same, $A_i(t) = A'_i(t - \Delta_{i1})$, but we know that $A_i(t) = A'_i(t - \Delta_i)$. Hence, $\Delta_i = \Delta_{i1} = \Delta_{i2}$.

case 2: Consider $\Delta_{i1} \neq \Delta_{i2}$.

Without loss of generality, assume $\Delta_{i1} > \Delta_{i2}$, and also $\Delta_{i1} = \Delta_{i2(\text{case1})}$ and $\Delta_{i2} < \Delta_{i2(\text{case1})}$ as shown in Figure 5. We define a new CDF due to Δ_{i2} as A''_{i2} , and the new resultant max as A''_i . Again by definition,

$$A''_i(t - \Delta_{i1}) = A'_{i1}(t - \Delta_{i1}) \cdot A''_{i2}(t - \Delta_{i1}) \quad (\text{EQ 4})$$

Also, $A''_{i2}(t - \Delta_{i1}) < A'_{i2}(t - \Delta_{i1})$, because $\Delta_{i2} < \Delta_{i2(\text{case1})}$. Hence, by equating EQ3 and EQ4, we get $A''_i(t - \Delta_{i1}) < A'_i(t - \Delta_{i1})$. This implies, $T(A''_i, p) > T(A'_i, p)$, and $T(A_i, p) - T(A''_i, p) < T(A_i, p) - T(A'_i, p)$ by algebraic manipulation. Hence, $\Delta_i < \Delta_{i1}$. \square

Note that the proof can be trivially extended for gates with more than two inputs.

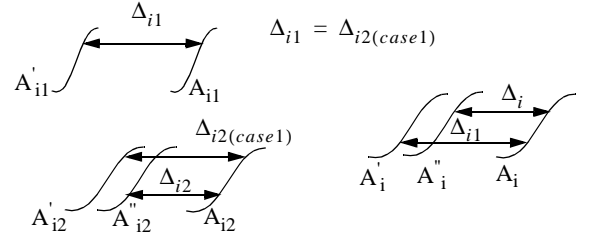


Figure 5. Arrival time CDFs - max operation (case 2)

Theorem 3. Max operation with single perturbed arrival time:

Consider a node i in the timing graph shown in Figure 4c. Let A_{i1} and A_{i2} be the arrival time CDFs of two fanin subgraphs incident at node i . Let A'_{i1} be the only perturbed CDF. If the arrival time CDF A_i and perturbed CDF A'_i at node i are given by, $A_i = \max(A_{i1}, A_{i2})$ and $A'_i = \max(A'_{i1}, A_{i2})$ respectively, then $\Delta_i \leq \Delta_{i1}$.

Proof : This is a special case of Theorem 2, where $\Delta_{i2} = 0$.

The above three theorems were defined assuming that the perturbed CDF has the exact same shape as the unperturbed CDF. As mentioned, this may not be true in practice and hence, we define a lower bound on the perturbed CDF which has the exact same shape as the unperturbed CDF as follows.

Definition 1. The lower bound CDF B'_i of perturbed arrival time A'_i is defined as the time shifted CDF A_i by Δ_i (Figure 6).

Since the shape of the lower bound B'_i is the same as that of the unperturbed CDF A_i , Theorems 1 through 3 can be applied to this lower bound. Note, however, that the maximum time difference between the lower bound of the perturbed CDF B'_i and the unperturbed CDF A_i is equal to the maximum difference between the perturbed CDF A'_i itself,

and A_i (by Definition 1). Hence, implicitly, Theorems 1 through 3 also hold for arbitrary shaped perturbations of an arrival time CDF. This allows the use of the perturbation bound Δ_i as an upper bound on the actual difference between the perturbed and unperturbed CDFs at the sink node. Using this bound allows gates to be pruned from consideration for the highest sensitivity gate as explained in more detail in Section 3.3.

Before presenting such a general upper bound on the perturbation of a CDF at the sink node, we first recognize that when we propagate a perturbed CDF in a circuit, multiple perturbed CDFs are generated at points of multiple fanout. We therefore present a so-called *perturbation front*, to represent such multiple perturbed CDFs, as follows:

Definition 2. Given a particular gate x for which a size change is considered, resulting in a perturbed arrival time CDF A'_x at the output of x , a perturbation front of level k , $P_k = \{n_{k1}, n_{k2}, n_{k3}, \dots\}$ is a set of nodes in the fanout cone of x which have a topological level less than k , and which have atleast one fanout node with a topological level greater than k .

In essence, the *perturbation front* corresponds to the set of nodes that is visited in each iteration of a breadth-first propagation of the perturbed CDF A'_x (explained in details in Section 3.3). We now define the maximum over all Δ_i where node i belongs to a *perturbation front* due to upsizing gate x as $\Delta m_x = \max_i \Delta_i$. Note that when the *perturbation front* reaches the sink node, it consists of only a single perturbed arrival time corresponding to the sink node.

We now present the following key theorem for bounding the perturbation of the arrival time CDF at the sink node due to a change in the size of a gate in the circuit.

Theorem 4. Given a perturbation front P_k associated with a gate x then $\Delta_{nf} \leq \Delta m_x$, where Δm_x is the maximum difference between perturbed and unperturbed arrival times over all nodes in the perturbation front P_k and Δ_{nf} is the maximum difference between perturbed and unperturbed CDF at the sink node nf .

Proof : The proof follows from Theorems 1 through 3.

In other words, the maximum difference between the perturbed and unperturbed CDF at the sink node, is bounded by the maximum change of the perturbed and unperturbed CDFs in the *perturbation front*. Note that as the *perturbation front* progresses closer to the sink node, the bound becomes increasingly tight and, by definition, becomes exact at the sink node itself.

3.3 Algorithm

In this section, our statistical gate sizing algorithm is presented. The objective function is the p -percentile point of the circuit delay CDF. The sensitivity S_x of gate x is computed numerically using the ratio of the change in p -percentile circuit delay per unit change in gate width: $S_x = \delta_{nf}(p) / \Delta w_x$ where, Δw_x is the change in gate width and nf is the sink node of G . Based on the computed sensitivities, the most sensitive gate in each iteration of the coordinate descent is selected.

The goal of the inner loop of the optimization is to find the gate with maximum sensitivity without performing a complete SSTA run for each gate perturbation in the circuit. The idea is to propagate highly sensitive gates (i.e. gates which have a large value of S_i) to the sink node and then use their S_i value to prune out gates which can be shown to have a lesser sensitivity using the proposed bounds. Given a gate x with partially propagated arrival time CDFs at *perturbation front* P_k , we define the perturbation front sensitivity bound $Sm_x = \Delta m_x / \Delta w_x$, where Δm_x is the maximum perturbation change across the nodes of the front. From Theorem 4 it follows that $Sm_x < S_x$ and hence the sensitivity bound Sm_x can be used to prune gate x before its *perturbation front* reaches the sink

node. In other words, if at any time during the propagation of the front for gate x the bound Sm_x become less than a previously computed sensitivity S_i of gate i , gate x can be eliminated from further consideration.

It is advantageous to identify a gate with a high sensitivity value S_i early in the analysis so that a large number of gates can be pruned. In our approach, we therefore perform level by level propagation of perturbed arrival times in an iterative manner. During every iteration the perturbation front with the maximum Sm_x value is propagated one level forward and its Sm_x value is recomputed. When a perturbation front reaches the sink node, its true sensitivity S_i is computed and is used to prune the perturbation front of other gates.

The pseudo-code of the statistical gate sizing algorithm is given in Figure 7.

Statistical_Gate_Sizing(G)

```

1. do {
2.   SSTA(G);
3.   For each gate x
4.     x.A'set = Initialize(x);
5.   gate_list = list of all gates x in G sorted by  $Sm_x$ 
6.   Max_S = 0;
7.   while( gate_list is not empty) {
8.     x = Head(List);
9.     PropagateOneLevel(x);
10.    Update  $Sm_x$ ;
11.    if (x.curr_prop_level = # of levels in G) {
12.      gate_list = gate_list - {x};
13.      if (Max_S <  $S_x$ ) {
14.        Max_S =  $S_x$ ;
15.        best_gate = x;
16.      }
17.    }
18.    else
19.      Update position of x in gate_list;
20.    gate_list = gate_list - {x |  $Sm_x < Max_S$ }
21.  }
22.  best_gate.w = best_gate.w +  $\Delta w$ ;
23. } while ( Max_S > 0 );

```

Figure 7. Statistical Gate Sizing Algorithm

First, SSTA is performed to compute the arrival times at each node (step 2). To implement level by level propagation, we maintain propagated arrival times A' and Sm_x for each candidate gate and use the notation ' $x.A'set$ ' which represents the super-set of gates in the current *perturbation front* of gate x . It is a super-set as it also contains the fanout gates of the current *perturbation front*, which is required to advance the *perturbation front* one level forward. Sm_x and $A'set$ are initialized for every gate in the circuit by calling procedure **Initialize** (step 3 and 4). In step 5, a sorted list of all gates in G is created by arranging gates in descending order of Sm_x . It represents the list of all unpruned candidates which may or may not result in maximum S_x , i.e. the set of all gates having $Sm_x > Max_S$, where Max_S is the maximum S_x amongst all candidate nodes whose *perturbation front* have reached the sink node. Max_S is initialized to be '0' (step 6) before beginning the search for the most sensitive gate.

In each iteration, the head of the sorted *gate_list* is selected for propagation. The procedure **PropagateOneLevel** propagates the arrival times by one level and updates the $A'set$ of gate x accordingly, as explained later. During propagation, new nodes are added to $A'set$ and nodes which do not belong to the *perturbation front* are deleted. Sm_x is re-computed in step 10. If perturbation front of gate x reaches the sink node, gate x is removed from the candidate *gate_list* (step 11 and 12) and Max_S is updated (step 13-17). On the other hand, if the perturbation front has not yet reached the sink node, the position of gate x in the sorted *gate_list* is

updated with respect to its new Sm_x (step 19). In Step 20, gates in the list for which $Sm_x < Max_S$ are removed from the list. When the candidate $gate_list$ becomes empty the propagation loop terminates and the gate with maximum sensitivity is sized up by Δw . The algorithm can be easily modified to size multiple gates in the same iteration.

The pseudocode for procedure **Initialize** is given in Figure 8. Given a

Initialize(gate x)

1. Change delays of x & fanin(x) for Δw increase in x.w;
2. $x.A'set = x \cup fanin(x)$;
3. $x.curr_prop_level = \min_{i \in x.A'set} x.Aset[i].level$;
4. while (x.curr_prop_level <= x.level)
5. **PropagateOneLevel(x)**;
6. Compute Sm_x ;
7. Restore Change in delay of x & fanin(x);

Figure 8. Initialize creates a perturbation front for gate x

gate x, the algorithm initializes the *perturbation front* A'set and computes the initial value Sm_x for gate x. The width w of gate x is temporarily sized up by Δw and the CDF for all pin-to-pin delays of gate x are updated. In addition, the delay CDFs of fanin gates that drive the inputs of gate x are also updated due to the increased loading by gate x (step 1). Consequently, the arrival time CDFs at gate x and its fanins will be perturbed and hence, these nodes are added to the initial *perturbation front* (step 2). In general, it is possible that the fanin of x is also a fanin of other fanins of x. Therefore, perturbed arrival times are computed starting from the minimum level gate (step 3). The procedure **PropagateOneLevel** is called iteratively until the propagation level reaches the level of gate x (step 4 and 5). The *perturbation front* created after step 5 is shown in Figure 9. Thereafter, Sm_x is computed in step 6. Finally, the original pin-to-pin delays of gate x and its fanin gates are restored (step 7).

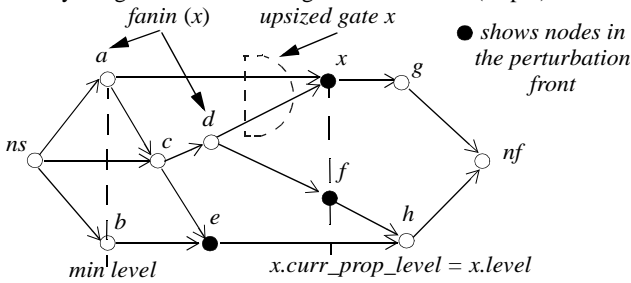


Figure 9. Perturbation front after the Initialize routine for gate x

The pseudocode for procedure **PropagateOneLevel** is given in Figure 10. In step 1-4, a list of gates at the propagation level is created from the A'set of x. For every node in the list, the perturbed arrival time distribution AT_PDF, is computed by performing convolution and max operations (step 6). Then Δ_i is computed by comparing the propagated perturbed arrival time and the original arrival time (i.e. the arrival time computed in step 2). For each node in prop_list, fanout nodes are added to the A'set in steps 8-12. Note that a node can be removed from the A'set once the perturbed arrival times at all its fanouts are computed. This is performed using a count variable *fo_count* which is initialized to the number of fanouts of the gate (step 11). When a perturbed arrival time is propagated from a node, *fo_count* of all fanin gates is decremented. A node is removed from A'set when its *fo_count* becomes zero.

4 Results

The proposed statistical optimization method was implemented and tested on a synthesized version of ISCAS'85 [15] benchmark circuits using a 180nm commercial cell library. We compare our proposed approach with deterministic and brute-force statistical optimization methods. Intra-die process variation was modeled using a truncated Gaussian gate delay distribution. The standard deviation was 10% of the nominal delay and the distribution was truncated at the 3 sigma point. However, any delay distribution could be used in our framework. As

PropagateOneLevel(gate x)

1. $prop_list = \phi$;
2. for_each_gate i \in x.A'set
3. if (i.level = x.curr_prop_level)
4. $prop_list = prop_list \cup i$;
5. for_each_gate i \in prop_list {
6. Compute x.A'set[i].AT_PDF using prop. & max;
7. Compute Δ_i ;
8. for_each_gate j \in fanout(i)
9. if (j \notin x.A'set) {
10. $x.A'set = x.A'set \cup j$;
11. $x.A'set[j].fo_count = j.fo_count$;
12. }
13. for_each_gate k \in fanin(i)
14. if (k \in x.A'set) {
15. $x.A'set[k].fo_count = x.A'set[k].fo_count - 1$;
16. if (x.A'set[k].fo_count = 0)
17. $x.A'set = x.A'set - \{ k \}$;
18. }
19. }
20. $x.curr_prop_level = x.curr_prop_level + 1$;

Figure 10. Propagate One Level

shown later, the arrival time bounds as suggested in [4] were compared with Monte Carlo simulation, showing an acceptable difference, especially for the 99-percentile point (< 1%).

The deterministic optimization that we use for comparison is similar to TILOS [7]. Sensitivities are computed for all the gates on the critical path and the gate with the highest sensitivity is sized up. These sensitivities are computed as the change in the *circuit delay* due to a change in the gate size. The approach is therefore better than traditional TILOS where the sensitivity is computed only in respect to the *critical path delay*. In the deterministic optimization, the delay of each gate was set to its 99-percentile point. The brute force statistical optimization, on the other hand, computes statistical sensitivities exactly by performing an SSTA run for every candidate gate.

Table 1 shows a comparison between the proposed statistical optimization and deterministic optimization. Results have not been shown for the brute force approach as they match exactly with the proposed optimization algorithm. The optimization results for the 99-percentile circuit delay point, after performing over 500 sizing iterations, is shown in *column* 3-6. Note that while the deterministic optimization does not use SSTA in the optimization process, the reported 99-percentile delay point was obtained by running SSTA on the circuit solution after every sizing iteration. In *column* 3, we report the % increase in the total gate size of the circuit due to optimization. *Column* 4 and 5 show the 99-percentile delay obtained from deterministic and statistical optimization, respectively. The improvement obtained from statistical optimization over deterministic optimization is shown in *column* 6. The average improvement is 7.8% over all benchmarks with a maximum improvement of 10.5%.

Table 2 shows a comparison of runtimes between brute force statistical optimization and our accelerated approach. Our optimization results are identical with those of the brute force approach, but provide an average runtime improvement of up to 20x for large circuits. The results demonstrate that as many as 19 out of 20 candidate nodes are pruned, demonstrating the effectiveness of the proposed bounds and pruning algorithm. We also found that the runtime per iteration varies significantly over the optimization iterations. In certain cases, there is a large range of gate sensitivities, and a highly sensitive gate quickly prunes out many inferior gates, whereas in other cases, many gates in the circuit have similar sensitivities, making pruning more difficult. Note that in the latter case, exact identification of the most sensitive gate will not be as important for the optimization result, opening the way for future research on fast heuristics for finding the most sensitive gate. In *column* 2 and 3, we report the average runtime per iteration (computed over 500 iterations) using the brute force and our accelerated approach, respectively. *column* 4 shows the runtime improvement factor. The improvement is

Table 1. Results for the 99-percentile delay point

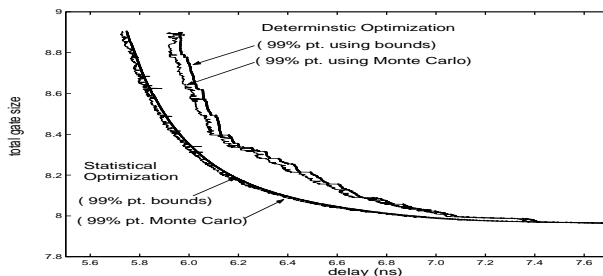
Circuit		Results for the 99-percentile delay pt. (ns)			
name	nodes/edges	% increase in total gate size	deterministic	statistical	% impr.
c432	214/379	97	3.49	3.14	10.03
c499	561/978	25.6	3.98	3.56	10.55
c880	425/804	93	4.09	3.74	8.55
c1355	570/1071	23.7	4.80	4.30	10.41
c1908	466/858	20.9	6.48	6.12	5.50
c2670	1059/1731	21.4	3.65	3.40	6.85
c3540	991/1972	11.5	5.98	5.70	5.0
c5315	1806/3311	6.7	5.90	5.40	8.47
c6288	2503/4999	28.1	16.00	15.05	5.93
c7552	2202/3945	13.1	8.10	7.60	6.17

higher for larger circuits, since in these circuits the cost of maintaining additional data structures is amortized over the savings of statistical computations, such as convolution and max. Finally, *columns* 5 and 6, show the range of runtimes per iteration for our approach and corresponding range of improvement factors, respectively.

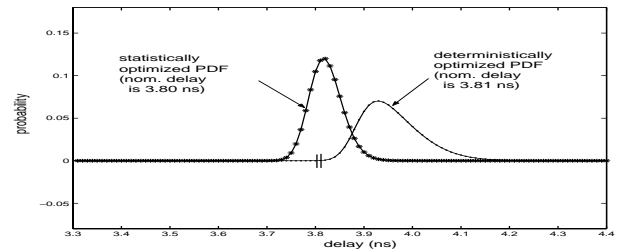
Table 2. Results for the runtime improvement

Circuit name	Average time per iteration (sec)			Range of time per iteration(s)	Range of impr. factor
	brute force	our algo.	imp. factor		
c432	5	1.21	4.13	0.12-1.7	3-42
c499	90	19.9	4.52	0.6-30	3-150
c880	15	3.37	4.45	0.5-4.3	3.5-30
c1355	95	19.8	4.79	1.56-31	3-61
c1908	102	22	4.63	0.9-29	3.5-113
c2670	43	4.47	9.62	0.4-6.4	7-107
c3540	194	23	8.43	1-30	6.5-194
c5315	403	33	12.2	2.1-47	8.6-191
c6288	3600	180	20	18-220	16-200
c7552	1190	87	13.68	2.5-136	9-476

Figure 11 shows the area-delay curve using our approach and deterministic optimization for c3540. The 99-percentile points of the circuit delay CDF are plotted on the x-axis and the corresponding total gate size value on the y-axis, for every sizing iteration. We have also plotted the 99-percentile points of the circuit delay using Monte Carlo simulations. As shown in the figure, there is a very small difference between the bounds and Monte Carlo results. Thus, using bounds as an optimization objective results in a comparable improvement in the exact circuit delay.

**Figure 11. Area-delay curve for c3540**

In Figure 12, we show the circuit delay PDF as obtained by the deterministic and statistical optimization methods, after the total gate size has been increased by 33% for c880. We can see that statistical optimization shifts the entire distribution to the left, along with reducing the variability of circuit delay. This is desirable as it increases the robustness of circuit to process variations and also improves the yield distribution. Note that, nominal delays for both the optimized circuits are almost the same, whereas their PDFs are entirely different.

**Figure 12. Comparison of circuit delay PDFs for c880**

5 Conclusions

In this paper, we have demonstrated the need for a fast statistical optimization algorithm. We have shown through our experiments that there is a clear advantage in using statistical optimization compared to a deterministic one. We proposed a fast statistical optimization which is exact in comparison with a brute force statistical optimization but provides significant speedup. Our approach is based on proposed theory of perturbation bounds which allow efficient identification of the highest sensitive gate for sizing. The perturbation bound is used in pruning out less sensitive gates without explicitly propagating their effect, and is a key contribution of this paper. Finally, we demonstrated the accuracy and efficiency of our approach over a large number of test cases. Our results show an average runtime improvement up to a factor of 20. Future work includes extending this framework to include spatial correlations and development of heuristics for fast and approximate identification of the statistically most sensitive gate in a circuit.

References

- [1] M. Berkelaar, "Statistical Delay Calculation, a Linear Time Method," Proceedings of TAU 97, Austin, TX, December 1997
- [2] J.J Liou, K.T. Cheng, S. Kundu, A. Krstic, "Fast Statistical Timing Analysis By Probabilistic Event Propagation", DAC 2001
- [3] L. Sheffer, "Explicit Computation of Performance as a Function of Process Variation", Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, TAU 2002.
- [4] A. Agarwal, D. Blaauw, V. Zolotov, S. Vrudhula, "Computation and Refinement of Statistical Bounds on Circuit Delay," DAC 2003.
- [5] A. Devgan, C. Kashyap, "Block-based Static Timing Analysis with Uncertainty," ICCAD 2003, pp.607-614
- [6] H. Chang, S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-like Traversal," ICCAD'03.
- [7] J. Fishburn, A. Dunlop. "TILOS: A Posynomial Programming Approach to Transistor Sizing", ICCAD 1985, pp. 326-328
- [8] M. Berkelaar, J.A.G. Jess, 'Gate Sizing in MOS Digital Circuits with Linear Programming', Proceedings of the European Design Automation Conference 1990, pp. 217-221.
- [9] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, S. M. Kang. "An Exact Solution to Transistor Sizing Problem for CMOS Circuits using Convex Optimization", IEEE Transactions on CAD, 1993.
- [10] A. R. Conn, I. M. Elfadel, W. W. Molzen Jr., P. R. O'Brien, P. N. Strenski, C. Visweswariah, C. B. Whan. "Gradient-based optimization of custom circuits using a static-timing formulation", DAC'99.
- [11] H. Hashimoto, H. Onodera. "Increase in delay uncertainty by performance optimization", ISCAS 2001, pp. 379-382.
- [12] X. Bai, C. Visweswariah, P. N. Strenski, and D. J. Hathaway. "Uncertainty-aware circuit optimization", DAC 2002, pp. 58-63
- [13] E.T.A.F. Jacobs, M.R.C.M Berkelaar. "Gate sizing using a statistical delay model", DATE 2000, pp. 283-289.
- [14] S. Raj, S. Vrudhula, J. Wang. "A methodology to improve timing yield in the presence of process variations", DAC 2004.
- [15] F. Brglez, H.Fujiwara, "A Neutral Netlist of 10 Combinatorial Benchmark Circuits", Proc. ISCAS, 1985, pp.695-698.