

Criticality Computation in Parameterized Statistical Timing

Jinjun Xiong
Univ. of California
Los Angeles, CA
jinjun@ucla.edu

Vladimir Zolotov
IBM Research
Yorktown Heights, NY

Natesan Venkateswaran
IBM Sys. & Tech. Group
East Fishkill, NY
{zolotov,natesan,chandu}@us.ibm.com

Chandu Visweswariah
IBM Research
Yorktown Heights, NY

ABSTRACT

Chips manufactured in 90 nm technology have shown large parametric variations, and a worsening trend is predicted. These parametric variations make circuit optimization difficult since different paths are frequency-limiting in different parts of the multi-dimensional process space. Therefore, it is desirable to have a new diagnostic metric for robust circuit optimization. This paper presents a novel algorithm to compute the criticality probability of every edge in the timing graph of a design with linear complexity in the circuit size. Using industrial benchmarks, we verify the correctness of our criticality computation via Monte Carlo simulation. We also show that for large industrial designs with 442,000 gates, our algorithm computes all edge criticalities in less than 160 seconds. The high accuracy and fast speed of the algorithm warrant future application of criticality probability for robust circuit optimization.

1. INTRODUCTION

As technology nodes shrink to 90 nm and below, it becomes more difficult to manufacture chips with guaranteed parametric timing yield due to substantial increase of process variations [1]. If these effects are not considered properly, the potential for silicon failure is high, and the associated cost for a design re-spin is prohibitive. Therefore, statistical methods have recently attracted attention as a promising approach to improve parametric timing yield.

In the deterministic approach, a circuit is optimized for a single combination of process parameters. However, as a result of manufacturing we receive chips corresponding to various combinations of process parameters. Deterministic optimization cannot guarantee that the chip satisfies design requirements for all or most of these combinations. Statistical optimization [2, 3, 4] is targeted to solve this problem. The goal of statistical optimization is to maximize yield while satisfying timing, area, power and other design constraints. This goal can be achieved only by considering the whole space of process variations. Parameterized statistical static timing analysis (SSTA) [5, 6] provides that kind of exploration, computing the circuit delay as a function of process parameters. Unfortunately, knowing circuit delay is not enough. The optimization needs more detailed guidance to select circuit fragments requiring improvement. The timing analyzer drives deterministic optimization by identifying a critical path. In the presence of process variations, the critical path is not unique because different paths can be critical in different regions of the process space.

There is a useful logical extension of the concept of a critical path called criticality probability [6]. Similar concepts

were used in the context of PERT networks where it was called criticality index [7]. The criticality of a path is the probability of manufacturing a chip in which the path is critical. The higher the criticality, the more important it is to improve the timing characteristics of the path. In [8] it is shown that the criticality of a path is equal to the sensitivity of the mean of the circuit delay with respect to the mean of the path delay. Often the concept of criticality is more convenient than sensitivity. First, the definition of criticality is simpler, clearer and more intuitive. Second, criticality computation does not require complicated chain ruling used for computing sensitivities, as in [8]. Third, we define a new concept called conditional criticality that gives us information on how critical a gate is considering only manufactured chips that fail timing. Conditional criticality is more useful for optimization since the optimization should focus only on chips violating timing constraints. Fourth, conditional criticality can be used for other applications like circuit synthesis and test generation.

There was an attempt to compute criticalities by multiplying tightness probabilities [6]. This approach assumes that the tightness probabilities represent independent events and can be multiplied, which is not correct due to globally correlated parameters and path reconvergence. However, as was shown in [2], even such inaccurate criticalities can be useful for guiding optimization.

In this paper, we develop an accurate and efficient technique for computing criticalities of timing edges in the context of parameterized block-based SSTA. We use the same concept of graph cutset [9] as [4] for computing timing yield gradient by perturbing PDFs of timing edges. Our computation uses only efficient operations of statistical minimum and maximum, and tightness probability computation. We do not make any assumptions about independence approximations on tightness probabilities. The proposed algorithm has linear complexity in the number of timing edges. We propose a new concept of conditional criticality and develop an efficient technique for its computation. We implement our algorithms in an industrial statistical timing analyzer and verify the correctness of our implementation by Monte Carlo simulation. We show that for an industrial ASIC with 442,000 gates, our algorithm computes all edge criticalities in less than 160 seconds.

The rest of the paper is organized as follows. Section 2 gives an overview of parameterized SSTA. Section 3 gives a formal definition of criticality probability and describes their properties. Section 4 presents algorithms for computing criticalities of timing edges. In section 5 we discuss experimental results; and section 6 draws conclusions.

2. PARAMETERIZED SSTA

A parameterized block-based SSTA models a digital circuit with a timing graph $G(N, E, n_s, n_f)$, where N is a set of nodes, E is a set of gates and wires, n_s is a source node, n_f is a sink node. The nodes represent pins of circuit gates. The edges (weighted by their delays) model signal propagation (either gate or wire delay) from one node to another. Latches and flip-flops are modeled both by signal propagation and by checking timing constraints. These checks are transformed into additional circuit outputs. By introducing a virtual source and sink connected with the circuit inputs and outputs, a circuit is modeled by a timing graph with a single source and sink as is shown in Fig. 1. The delays of the edges connecting the primary circuit outputs to the sink are set to the negative of the required arrival times at the corresponding primary outputs. Assuming that circuit loops are broken for timing analysis, a timing graph is a directed acyclic graph (DAG). The longest path delay in the timing graph is the negative of the circuit slack in late mode, and the shortest path delay is equal to the circuit slack in early mode. For brevity, we consider only late mode analy-

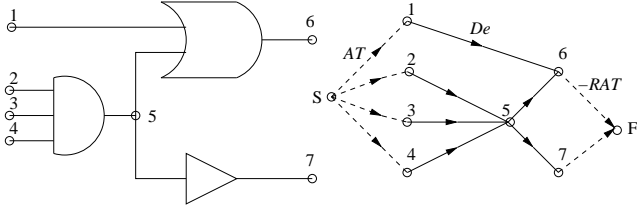


Figure 1: Circuit and its timing graph

sis when we compute the latest possible arrival times (ATs) and required arrival times (RATs). For computing ATs the signals are propagated forward from the source to the sink. Signals are propagated through edges by adding edge delays to signal ATs. At nodes the worst AT is calculated as the maximum of all ATs of the incoming edges. For computing RATs the signals are propagated backward from the sink to the source. Signals are propagated through edges by subtracting edge delays from signal RATs. At nodes the worst RAT is calculated as the minimum of all RATs of the incoming edges. Thus, node AT represents the maximum delay from the circuit source to the node while node RAT is the maximum delay from the node to the sink but with a minus sign.

In parameterized SSTA, edge delays, ATs and RATs are canonical first-order forms of process parameters (sources of variation):

$$D = d_0 + \sum_{i=1}^n d_i X_i + d_r X_r, \quad (1)$$

where d_0 is the mean or nominal delay, X_i and X_r are random variables representing globally correlated and uncorrelated sources of variation, and d_i and d_r are the sensitivities to the variations X_i and X_r , respectively. All the sources of variation have independent Gaussian distributions with zero mean and unit variance. However, the proposed technique can be generalized to more complex models [10].

Signal propagation in parameterized SSTA is performed similarly to deterministic timing. The addition of first-order forms is performed by summing their corresponding sensitivities to process parameters and statistical summation of

uncorrelated terms. The maximum and minimum operations are approximated linearly using Clark's formulas and the concept of tightness probability [11, 5, 6].

Given two random variables D_1 and D_2 , the tightness probability of D_1 is the probability of D_1 being greater than D_2 , i.e., $T_{D_1} = P(D_1 > D_2)$. If both D_1 and D_2 have normal distribution with means $d_{1,0}$ and $d_{2,0}$, variances σ_1^2 and σ_2^2 , and covariance $\delta = cov(D_1, D_2)$, then the tightness probability T_{D_1} is given by

$$T_{D_1} = P(D_1 > D_2) = \Phi\left(\frac{d_{1,0} - d_{2,0}}{\theta}\right) \quad (2)$$

where Φ is the cumulative probability distribution function (CDF) of a standard normal distribution, and $\theta = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\delta}$. The mean and variance of the statistical maximum of D_1 and D_2 are computed as

$$\mu = T_{D_1} d_{1,0} + T_{D_2} d_{2,0} + \theta \phi\left(\frac{d_{1,0} - d_{2,0}}{\theta}\right) \quad (3)$$

$$\sigma^2 = T_{D_1}(\sigma_1^2 + d_{1,0}^2) + T_{D_2}(\sigma_2^2 + d_{2,0}^2) + (d_{1,0} + d_{2,0})\theta \phi\left(\frac{d_{1,0} - d_{2,0}}{\theta}\right) - \mu^2, \quad (4)$$

where ϕ is the probability density function (PDF) of a standard normal distribution. The statistical maximum of D_1 and D_2 is approximated linearly as follows:

$$\max(D_1, D_2) = \mu + \sum_{i=1}^n (T_{D_1} d_{1i} + T_{D_2} d_{2i}) X_i + d_r X_r, \quad (5)$$

where d_r is determined by matching the variance of (5) to the value given by (4). The minimum operation is approximated in a similar way. The sequel assumes a parameterized statistical timing framework of the type described in [6].

3. STATISTICAL CRITICALITY

3.1 Definitions and Properties

DEFINITION 1. Criticality of a path is the probability of manufacturing a chip in which this path is critical.

DEFINITION 2. Criticality of a set of paths is the probability of manufacturing a chip in which at least one path from this set is critical.

DEFINITION 3. Criticality of an edge (node) is the probability of manufacturing a chip in which this edge (node) is on the critical path.

The criticality of a timing edge (node) is the criticality of the set of all paths going through that edge (node). The concept of criticality can be extended to a circuit gate, and even to any fragment of a timing graph or circuit. This flexibility is useful for circuit optimization. The concept of criticality is illustrated in Fig. 2 on the example of a simple circuit and its two-dimensional process space. A path $P1$ is critical if the process parameters fall in the area $P1$ in the process space during manufacturing. The criticality of the path $P1$ is therefore the probability of manufacturing a chip with the combination of parameters falling in this area $P1$.

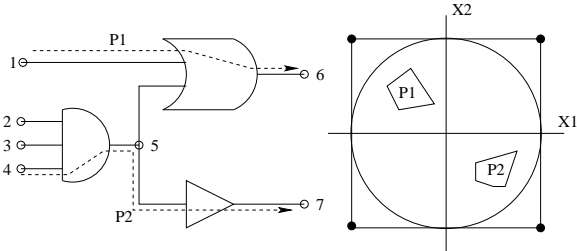


Figure 2: Critical paths and process subspaces.

The criticality of a set of paths S can be expressed by the following integral:

$$\int \cdots \int_{\substack{\max_{P_i \in S}(D(P_i)) > \\ \max_{P_j \notin S}(D(P_j))}} p(X_1, X_2, \cdots) dX_1 dX_2 \cdots \quad (6)$$

where $p(X_1, X_2, \cdots)$ is the joint PDF of process parameters X_1, X_2, \cdots ; P_i, P_j are circuit paths belonging and not belonging to the set S , respectively, and $D(P_i), D(P_j)$ are path delays as functions of process parameters. Obviously, this formula is not practical for computing criticalities as it requires multidimensional integration across complex polyhedra. So, we need a better way of computing criticalities.

It is useful to prove the following simple lemma.

LEMMA 1. *If two canonical forms A and B differ in at least one coefficient, then the probability that they are equal is 0.*

Proof: The equality of two canonical forms is expressed as

$$a_0 + \sum_{i=1}^n a_i X_i + a_r X_{ra} = b_0 + \sum_{i=1}^n b_i X_i + b_r X_{rb}. \quad (7)$$

This equality can be rewritten as

$$(a_0 - b_0) + \sum_{i=1}^n (a_i - b_i) X_i + a_r X_{ra} - b_r X_{rb} = 0. \quad (8)$$

This equation defines a hyperplane in the process space because at least one of its coefficients is not zero. The probability that the canonical forms are equal is a volume integral of the joint PDF of the process parameters over this hyperplane. Obviously this probability is zero for any practical probability distribution because the thickness of any hyperplane is zero. ■

Using this lemma we prove the following theorem:

THEOREM 1. *The criticality of a set S of paths is the sum of the criticalities of these paths.*

Proof: If no paths have the same canonical form of their delays, the theorem is obvious because the probability that several paths are critical simultaneously is 0. Assume that n paths have the same canonical form D of delay and that the probability that this delay is larger than any other path delay in the circuit is p . Then, according to convention, each of these paths is assigned criticality probability p/n . Thus, the theorem holds in this case, too. ■

From this theorem we derive the following corollaries.

COROLLARY 1. *Edge (node) criticality is equal to the sum of criticalities of the paths going through this edge (node).*

COROLLARY 2. *If all the circuit's paths are divided into two non-intersecting subsets A and B , then the criticality of*

subset A is the probability that the maximum delay of the paths belonging to the set A is larger than the maximum delay of the paths belonging to the set B .

In order to compute edge criticality, we compute the canonical form A of the maximum delay of the paths going through this edge and the canonical form B of the maximum delay of the paths not going through this edge. Then the edge criticality is simply the probability that $A > B$, or, in other words, the tightness probability of A with respect to B .

3.2 Principles of Computation

In order to explain edge criticality computation, we give several definitions and recall some facts from network theory.

DEFINITION 4. **Edge slack** of an edge is the maximum delay of all paths going through the edge.

DEFINITION 5. **Complement edge slack** of an edge is the maximum delay of all paths not going through the edge.

The maximum of any edge's slack and complement slack is the longest path of the circuit, which is the negative of the slack of the circuit in late mode (edge slacks are defined in this manner to avoid minus signs in the following derivations).

From these definitions, it follows that the edge criticality is the probability that the edge slack is greater than the complement edge slack, or, in other words, the tightness probability of the edge slack with respect to the complement edge slack. Tightness probability can be computed by formula (2).

The set of paths going through edge e forms a so-called edge flow graph G_e consisting of three parts: the edge input cone, the edge e itself, and the edge output cone. The slack s_e of edge e is simply delay $D(G_e)$ of its flow graph G_e . It can be expressed as:

$$s_e \equiv d(G_e) = D_{incone} + d(e) + D_{outcone}, \quad (9)$$

where D_{incone} is the delay of the edge input cone, $d(e)$ is the delay of the edge e , $D_{outcone}$ is the delay of the edge output cone. Recalling the definition of arrival and required arrival times, we express the slack of edge $e = (i, t)$ as follows:

$$s_e = AT(i) + d(e) - RAT(t) \quad (10)$$

where $AT(i)$ is the arrival time at the initial node i of the edge e and $RAT(t)$ is the required arrival time at terminal node t of the edge e . Fig. 3 illustrates edge slack computation.

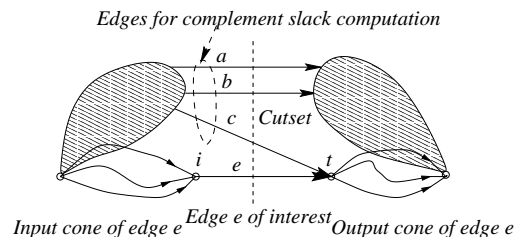


Figure 3: Edge slack computation.

Computation of complement edge slack is more complicated and requires additional considerations. In network theory [9], a cutset between source and sink nodes is defined as a set of edges whose removal from the network disconnects the source and sink nodes. Here, we consider only

those cutsets satisfying the condition that each path from the source to the sink has only one common edge with each cutset. We call these cutsets *minimal separating cutsets*. Algorithm 1 computes minimal separating cutsets Ω_i covering a given timing graph. For any edge of the timing graph, this algorithm computes at least one cutset containing that edge.

ALGORITHM 1. Cutset computation

1. Levelize the timing graph by topological sort
2. $\Omega_0 = \{\text{Edges outgoing from source node}\}$
3. For each level i do the following:
 - 3.1 $\Gamma = \{\text{Edges incoming to level } i\}$
 - 3.2 $\Lambda = \{\text{Edges outgoing from level } i\}$
 - 3.3 $\Omega_i = \Omega_{i-1} - \Gamma + \Lambda$

Fig. 4 illustrates cutset computation. We move a scan line along the levelized timing graph from the source to the sink. Each time we step over a level of the graph, we transform the current cutset into the next one by excluding the edges coming into the nodes of the current level and including the edges going out from the nodes of the current level. Any

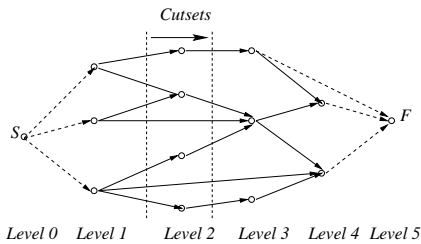


Figure 4: Computation of cutsets.

cutset computed by the algorithm separates the nodes of the timing graph into two sets: N_i containing the source node and N_f containing the sink node. Any node from the set N_i belongs to the lower level of the timing graph than any node from the set N_f . From that, we can conclude that any path from the source to the sink intersects any cutset computed by this algorithm exactly at one edge.

In order to compute the complement slack for edge e , we consider a minimal separating cutset C_e containing this edge. Let $C_{\bar{e}} = C_e - \{e\}$ be a set of all the cutset edges except e . Then the set of paths going through the edges of the set $C_{\bar{e}}$ includes all the paths of the timing graph except the paths going through the edge e . The maximum delay of the paths going through the set of edges $C_{\bar{e}}$ is exactly the complement slack of the edge e . The complement slack can be computed as the statistical maximum of all the edge slacks of members of the set $C_{\bar{e}}$. The complement slack of edge e , shown in Fig. 3, is computed as

$$s_{\bar{e}} = \max(s_a, s_b, s_c) = \max(s_a, \max(s_b, s_c)). \quad (11)$$

Unfortunately, a naive implementation of this approach has at least quadratic complexity since each edge criticality requires calculation of the maximum of all other edge slacks.

4. EDGE CRITICALITY COMPUTATION

The efficiency of complement slack computation can be improved if we use a tree data structure to re-use intermediate complement slack values. We construct a hierarchical partition of the cutset and compute all the complement slacks

simultaneously, remembering and reusing slacks and complement slacks of the partition subsets. For simplicity, we assume that the cutset has $n = 2^t$ edges and construct a balanced binary partition tree shown in Fig. 5. However, our approach can be applied to cutsets with an arbitrary number of edges. The construction of the partition tree can be done either top down by sequentially splitting the sets of edges into equal parts or bottom up by merging pairs of edges and then pairs of the subsets of edges. Each leaf node

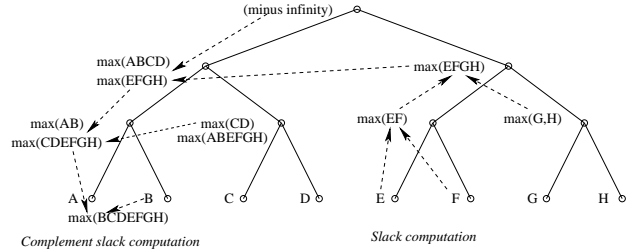


Figure 5: Binary partition tree.

in the partition tree represents one edge of the cutset. Each non-leaf node defines two sets of edges: the set of the node's children and the set of the edges that are not the node's children. With each node of the tree we associate a node slack and a node complement slack. The node slack is the maximum of slacks of its child edges. The node complement slack is the maximum of the slacks of non-child edges. For a leaf node, these two slacks are exactly the edge slack and the complement edge slack. The computation of node slacks and complement node slacks is illustrated in Fig. 5. The following algorithm computes the slacks associated with tree nodes and edge complement slacks.

ALGORITHM 2. Complement slack computation

1. Construct a partition tree of the cutset edges
2. Assign edge slacks to the leaf nodes
2. Traverse the tree bottom-up
 - 2.1. For each non-leaf node compute slack as the maximum of its children's slacks
3. Assign minus infinity slack to the root node
4. Traverse the tree top-down
 - 4.1 For each node compute its complement slack as the maximum of its parent's complement slack and its sibling node's slack
5. For each leaf node compute edge criticality as the tightness probability of the edge slack and the complement edge slack

This algorithm computes criticalities of all edges in the cutset simultaneously and has linear complexity as the number of max operations is proportional to the number of the tree nodes. A detailed analysis shows that it requires $4n - 6$ max operations where tightness probability computation is considered as a max operation, too.

4.1 Speedup Technique

In practical circuits, an edge may intersect many cutsets as is shown in Fig. 6. This often happens in sequential circuits with many flip-flops. In real circuits more than 50% of edges go through multiple cutsets. Occurrence of edges in multiple cutsets significantly slows down criticality computation because the same edge is processed multiple times in different cutsets. In order to improve efficiency, we developed

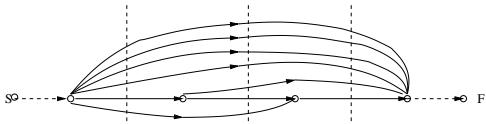


Figure 6: Edges going through many cutsets.

a technique to eliminate repeated processing of the same edge. This technique is based on the observation that after the criticality of an edge is computed once, the slack of this edge is used only for computing the complement slacks of the other edges by means of a max operation. The computation of the complement slacks does not require knowledge of the individual edge slacks and any group of edges can be represented by the maximum of their edge slacks.

We construct an array with elements corresponding to the circuit levels. The i -th element of the array holds the maximum slack of the edges that go through multiple cutsets and have a terminal node at the i -th level. At the beginning the array is initialized with minus infinity values. The edge criticalities are computed by processing cutsets in ascending order. Each time we encounter an edge going through many cutsets, we accumulate the slack of this edge in the proper element of this array by means of a max operation. From this array, we compute the maximum of the edge slacks of all the edges intersecting both the current cutset and the previously processed cutsets. The algorithm for computing complement edge slacks processes only the edges that are not present in any previous cutsets. The complement edge slacks are incremented with the maximum of edge slacks of the edges present in the previously processed cutsets. The complexity of the modified algorithm for computing the criticality of all edges is $O(N) + O(l^2)$, where l is the depth of the timing graph and N is the number of graph edges. In practice, the depth of a timing graph is much smaller than the number of its edges. Therefore, for all practical purposes, the complexity of the algorithm is $O(N)$, which implies *constant time* for computing the criticality of each edge.

4.2 Conditional Criticality

The criticality introduced above is the probability of a path, edge or node being critical among all the manufactured chips. However, optimization for yield is interested in improving only those chips that violate timing constraints without wasting resources speeding up chips that are already sufficiently fast. In other words, if an edge or path is only critical in a subset of the process space and there are no failing chips in that region, then there is no point improving that edge or path. The concept of conditional criticality helps to solve this problem by providing information on how critical an edge is among failing chips only.

DEFINITION 6. Conditional criticality of a path (edge, node) is the conditional probability of manufacturing a chip in which this path (edge, node) is critical, conditional upon the chip violating its timing constraints.

Conditional criticality of an edge e is expressed as follows:

$$p(e \sim crit | T_fails) = \frac{p(e \sim crit \& T_fails)}{p(T_fails)} \quad (12)$$

where $p(e \sim crit | T_fails)$ is the conditional probability that the edge e is critical conditional upon the timing constraints being violated; $p(e \sim crit \& T_fails)$ is the probability that the edge e is critical and the timing constraints

are violated; $p(T_fails)$ is the probability that the timing constraints are violated. In other words, conditional criticality is the probability of being critical computed for the sample space consisting of only those chips violating timing constraints.

In our case, the required time at the sink node is always 0. So the condition that the timing constraints are violated is $D_{ckt} > 0$ where D_{ckt} is the circuit delay. The probability $p(T_fails)$ that the timing constraints are violated is the tightness probability of D_{ckt} with respect to 0. The condition that the edge e is critical is $s_e > s_{\bar{e}}$ where s_e is the slack of the edge e and $s_{\bar{e}}$ is the complement slack of the edge e . This condition can be rewritten as $s_e - s_{\bar{e}} > 0$. Then the condition that both the edge e is critical and the timing constraints are violated is expressed as

$$\min(s_e - s_{\bar{e}}, D_{ckt}) > 0. \quad (13)$$

So, the probability $p(e \sim crit \& T_fails)$ that the edge e is critical and the timing constraints are violated is just tightness probability of $\min(s_e - s_{\bar{e}}, D_{ckt})$ with respect to 0. Using edge slacks and complement edge slacks, this probability can be efficiently calculated by linear approximation of the statistical min operation. If the distribution of statistical min is highly skewed to the left, the accuracy of the Gaussian approximation can be low. In this case, it is better to represent the result of the min operation with a better normal distribution and compute the tightness probability numerically. Substituting the computed probabilities into (12), we compute the conditional criticality of the edge e conditional upon the timing constraints being violated. Thus conditional edge criticalities can be computed with the same efficiency as unconditional ones. Similarly, it is possible to compute conditional criticality only considering chips that meet their timing constraints. This kind of conditional criticality is useful for statistical optimization by down-sizing gates to reduce power consumption.

5. EXPERIMENTAL RESULTS

We have implemented the proposed technique for computing both unconditional and conditional criticalities in the industrial statistical static timing analyzer EinsStat [6]. The criticalities are calculated for all timing edges for all clock phases, rising and falling transitions, early and late modes. We implemented three algorithms for criticality computation: the basic algorithm with straightforward calculation of the complement edge slacks, the partition tree based algorithm, and the algorithm with the speed up technique eliminating repeated processing of edges intersecting multiple cutsets.

Table 1 shows the run times of the different algorithms and compares them with the run time of basic SSTA. Columns 1 and 2 show chip name and the number of gates. Columns 3, 4, 5, 6 report CPU time for SSTA and three different versions of the criticality computation algorithm. Additionally, columns 5 and 6 provide information on the absolute and relative overhead of criticality computation (of all edges in the graph) over and above statistical timing. From Table 1 we see that both the partition tree approach and the elimination of repeated processing of edges intersecting multiple cutsets are important enhancements of the criticality computation technique. Only with these modifications is the criticality computation always faster than the base statistical timing, making it sufficiently fast for such applications as optimization, synthesis and test generation. From Table 1 we see

Table 1: Run time comparison

Chip	Size	SSTA	Basic techn.	Partition tree	Speedup techn.
D1	0.15k	1.97s	0.36s	0.20s/10.2%	0.19s/9.6%
D2	0.66k	7.28s	0.23s	0.04s/0.5%	0.04s/0.5%
D3	3.04k	3.37s	13.3m	4.44s/1.3x	0.39s/11.6%
D4	57.4k	21.5s	-	1.92m/5.4x	14.1s/65.6%
D5	87.2k	1.40m	-	52.5s/62.7%	27.5s/32.9%
D6	442k	24.9m	-	14.6m/58.6%	2.66m/10.7%

Table 2: Accuracy of criticality computation

σ (%)	0.3%	0.6%	1.5%	3%	4.5%	5%
maxDiff	0.0004	0.0003	0.0012	0.0006	0.0296	0.0754
sumDiff	0.0012	0.0359	0.144	0.0585	0.711	1.104

that for the large design with 442,000 gates the proposed technique can compute criticalities in only 2.66 minutes of CPU time, while it takes 24.9 minutes to perform the basic statistical timing analysis.

For verifying the accuracy of our algorithms, we implemented a Monte Carlo technique for criticality computation. We generate 10,000 random samples of process parameters, compute gate delays corresponding to them and perform deterministic timing analysis to find a critical path corresponding to each sample. Counting the number of times that each edge is on the critical path for all Monte-Carlo samples, we compute the criticality probability of that edge.

Table 2 shows the accuracy of the criticality computation for different amounts of process variation. The amount of process variation is given as the average standard deviation of gate delay expressed as a percentage of nominal gate delay. The first line of the table reports the maximum difference between criticalities computed by the proposed algorithm and the Monte-Carlo technique. The second line reports the sum of the absolute values of the differences between the criticalities computed by the proposed algorithm and Monte-Carlo for all edges of the timing graph. We see that the proposed technique has high accuracy.

Additional investigation of the sources of computational error shows that the main part of the error is due to the linear approximation of the statistical min and max operations used in the parameterized SSTA. This error grows when the delay variation is larger, which we can see from this table.

We performed a number of experiments to compute conditional criticalities. The experiments show that conditional criticalities can be computed with approximately the same accuracy and CPU time as the unconditional ones. This conclusion fully agrees with our expectation since the computation of conditional criticalities is only a little extra work after the computation of unconditional criticalities. On the other hand, our experiments show that the values of the conditional criticality can be significantly different from the values of the unconditional ones. In fact, conditional criticality significantly depends on the length of the circuit clock cycle while the value of unconditional criticality is always the same. Fig. 7 demonstrates the dependence of the conditional criticalities on clock cycle length for four different timing edges. We see that when the clock cycle is longer, the conditional criticality of different edges may either increase or decrease depending on the circuit topology and process variation.

6. CONCLUSIONS AND DISCUSSION

In this paper we described an accurate and efficient method for computing criticalities of all timing edges in the context of parameterized block-based statistical timing anal-

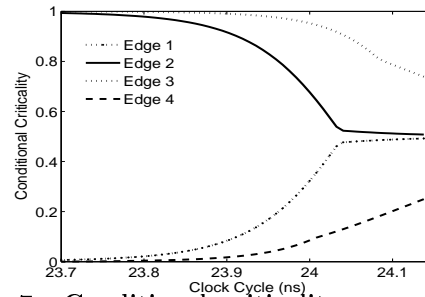


Figure 7: Conditional criticality as a function of clock cycle

ysis. Our algorithm computes criticalities of all edges of the timing graph with linear complexity with respect to the number of timing edges and can compute both unconditional and conditional criticalities. We implemented the proposed algorithm in an industrial SSTA tool targeted for both sign-off timing analysis and for guiding circuit synthesis and optimization. Computational experiments with industrial circuit designs having up to 442K gates demonstrated high accuracy and low run time of the proposed technique. The maximum error of criticality computation is about 7.5% compared to Monte-Carlo simulations. The CPU time of criticality computation varies from 10% to 65% of the CPU time of statistical timing analysis even for large designs.

7. REFERENCES

- [1] C. Visweswariah, "Death, taxes and failing chips," *Proc. 2003 Design Automation Conference*, pp. 343–347, June 2003. Anaheim, CA.
- [2] M. R. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov, "Gate sizing using incremental parameterized statistical timing analysis," *IEEE International Conference on Computer-Aided Design*, pp. 1029–1036, November 2005. San Jose, CA.
- [3] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov, "Circuit optimization using statistical timing analysis," *Proc. 2005 Design Automation Conference*, pp. 321–324, June 2005. Anaheim, CA.
- [4] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester, "Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation," *IEEE International Conference on Computer-Aided Design*, pp. 1023–1028, November 2005. San Jose, CA.
- [5] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," *IEEE International Conference on Computer-Aided Design*, pp. 621–625, November 2003. San Jose, CA.
- [6] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," *Proc. 2004 Design Automation Conference*, pp. 331–336, June 2004. San Diego, CA.
- [7] B. Dodin and S. Elmaghraby, "Approximating the criticality indices of the activities in PERT networks," *Management Science*, vol. 31, pp. 207–223, February 1985.
- [8] X. Li, J. Le, M. Celik, and L. T. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations," *IEEE International Conference on Computer-Aided Design*, pp. 844–851, November 2005. San Jose, CA.
- [9] N. Deo, *Graph theory with applications to engineering and computer science*. Prentice-Hall, 1974.
- [10] H. Chang, V. Zolotov, C. Visweswariah, and S. Narayan, "Parameterized block-based statistical timing analysis with non-Gaussian and nonlinear parameters," *Proc. 2005 Design Automation Conference*, pp. 71–76, June 2005. Anaheim, CA.
- [11] C. E. Clark, "The greatest of a finite set of random variables," *Operations Research*, pp. 145–162, March-April 1961.