Statistical Critical Path Analysis Considering Correlations

Yaping Zhan, Andrzej J. Strojwas, Mahesh Sharma*, David Newmark*
Department of ECE, Carnegie Mellon University, Pittsburgh, PA,
Advanced Micro Devices Inc., Austin, TX*

Abstract

Critical Path Analysis is always an important task in timing verification. For today's nanometer IC technologies, process variations have a significant impact on circuit performance. The variability can change the criticality of long paths [1]. Therefore, statistical approaches should be incorporated in Critical Path Analysis. In this paper, we present two novel techniques that can efficiently evaluate path criticality under statistical non-linear delay models. They are integrated into a block-based Statistical Timing tool with the capability of handling arbitrary correlations from manufacturing process dependence and also path sharing. Experiments on ISCAS85 benchmarks as well as industrial circuits prove both accuracy and efficiency of these techniques.

1. Introduction

For today's increasing complexity of VLSI designs and tighter timing constraints, timing verification has become a more challenging and important task. Timing information is very useful for both design optimization and yield improvement in manufacturing. Meanwhile, as IC technologies are scaled down to the nanometer regions, circuit delays become highly dependent on manufacturing process variations, especially the intra-die variations, of both gates and interconnects. Due to the correlations among component (gate and interconnect) delays, corner case analysis using traditional Static Timing Analysis (STA) tools is very pessimistic and not capable of finding the circuit delay in variational environments.

As a solution, Statistical Static Timing Analysis algorithms (SSTA) have been proposed recently. Instead of propagating fixed delay values through gates and interconnect, SSTA propagates delay distributions characterized by delay Probability Density Functions (PDFs) [2-8].

Path-based SSTA algorithms perform delay analysis path by path [2][3]. They are accurate, good at capturing correlations, and able to report statistical critical paths for circuit optimizations. However, due to the large number of long paths in commercial circuits, we cannot afford to analyze all those paths. Usually, only the top K longest paths are selected and passed to the path-based algorithms for analysis. The task of selecting the top paths before statistical analyses, when both inter-die and intra-die variations are present, is a very challenging one. Since non-critical paths in some part of the process space might become critical in the other part of the process space, no one knows whether a path may or may not become critical in a particular variational environment without statistical analyses. Therefore, some statistical critical paths may not be included in the path set, thus not be analyzed by the path-based SSTA algorithms. All these paths that are not in the top path set will never be reported for optimization which makes the sizing of the reported long paths not effective.

On the other hand, block-based algorithms [4-8] are efficient and incremental, but they often do not provide good estimates on critical paths. Since block-based approaches are breadth-first and done level by level, no critical paths are reported to circuit designers. However, from the circuit designers' standpoint, critical paths are equally valuable as the final circuit delay distributions, since designers can improve circuit performance by optimizing critical path delays. Meanwhile, to ensure correct timing behavior, it is a common practice to include testing of a set of critical paths. Critical paths are very useful for test engineers to generate the test vectors as well. Moreover, the critical paths can be treated as the top K longest paths for path-based SSTA algorithms that enable us to do more specific analyses for better timing accuracy. Therefore, for block-based algorithms, accurate critical path analysis is very useful.

As we know, under statistical delay model assumptions, many paths will change their rankings due to process variations. Non-critical paths under nominal delay environment may become critical under certain variations. Therefore, traditional critical path analysis must be extended to statistical critical path analysis. Moreover, the correlations resulting from path sharing as well as manufacturing process dependence make this problem even more challenging.

In this paper, we propose two novel approaches for path criticality analysis with the capability of handling any delay correlations between any two paths. Both approaches have been integrated into our block-based SSTA tool as a post-processing step. The approaches require little CPU overhead and provide very accurate results. The organization of the rest of the paper is as follows: In Section 2, we review previous research on path criticality calculations. We propose our two efficient approaches and discuss our algorithms in Section 3. Experimental results for both ISCAS85 and industrial circuits are given in Section 4. We conclude this paper in Section 5.

2. Background

Traditionally, critical paths are defined as the longest sensitizable paths under fixed delay models. However, under statistical delay models, each path has a delay distribution. A path may be longer than another path under some process variations, but shorter in other cases. The criticality probability is associated with certain circumstances under which a sensitizable path is critical, i.e., it determines the delay of the circuit. Any path that has a non-zero probability of being critical is defined as one of the statistical critical paths. The sum of the criticality probabilities of all statistical critical paths is one. The statistical critical paths are ranked by their criticalities. The path with the largest criticality probability is most important and is the one that circuit designers should look at first for performance and yield optimization.

To calculate path criticality in the nanometer process technologies, correlations between paths must be taken into account. There are two major sources of correlations. The first source is from re-convergent fan-outs of the circuit, and the other source comes from the spatial correlations of process parameters, such as inter-die variations and systematic intra-die

0-7803-9254-X/05/$20.00 ©2005 IEEE.

variations. To consider these correlations, we use the parameterized SSTA technique to represent all the delays and arrival times. Quadratic delay models are used to account for nonlinear delays. Suppose $X=(x_1,x_2,...,x_n)$ is a set of independent process parameters with normalized Gaussian distributions, which can be derived from a Principal Component Analysis (PCA) approach. We can represent any gate/interconnect delay D_i and signal arrival time AT_i in the following forms:

$$D_i = X^T A_{Di} X + B_{Di} X + C_{Di} \quad (1)$$

$$AT_i = X^T A_{ATi} X + B_{ATi} X + C_{ATi} \quad (2)$$

In a previous research [5], the authors proposed their path criticality calculation method in a block-based SSTA methodology. In a timing graph, each edge is annotated with an Arrival Tightness Probability (ATP), which is the probability that the edge determines the arrival time of its sink node. The probability of a path being critical is calculated as the product of the ATP's of all edges along the path. However, this method is valid only when the criticality probabilities of any two paths are independent to each other. When complicated correlations are present in modern circuits, this assumption is not true. To demonstrate this, let us examine the circuit example with re-convergent fan-outs and process spatial correlations shown in Figure 1.

We assume that the delay of each circuit node consists of two parts, the systematic part due to spatial correlations and the independent random part. More specifically, we assume the systematic delays of all nodes are equal to d_0 due to their close locations, and the random parts are independent normalized Gaussian variables (d_A through d_E). This assumption is not necessary for our analysis, but just to simplify the criticality calculations for this example.

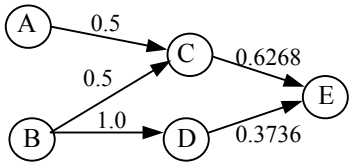

Figure 1 Circuit example with re-convergent fan-outs

Table 1. Node delays and arrival times of the example in Figure 1

Node	Delay	Output Arrival Time
A	d_0+d_A	d_0+d_A
B	d_0+d_B	d_0+d_B
C	d_0+d_C	$2d_0+d_C+max(d_A,d_B)$
D	d_0+d_D	$2d_0+d_B+d_D$
E	d_0+d_E	$3d_0+d_E+max(d_B+d_D, d_C+max(d_A,d_B))$

The delays and arrival times of all the nodes are listed in Table 1. Based on our simplified assumption, the true criticality probabilities of all the paths can be manually obtained from path analyses. For path B-C-E to be longer than path A-C-E, condition $d_B>d_A$ must be satisfied, and for it to be longer than path B-D-E, $d_C>d_D$ should be true. Since conditions $d_B>d_A$ and $d_C>d_D$ are independent, the criticality of path B-C-E is equal to $P(d_B>d_A)*P(d_C>d_D)$, which is 0.25. Path A-C-E and B-D-E are symmetric when compared to path B-C-E, so their criticalities are equal. The true criticalities of all the paths are listed as the second column of Table 2.

The ATP of each edge is obtained from a block-based SSTA algorithm and marked directly in Figure 1. The last two columns of Table 2 show path criticalities calculated as the products of all ATP's along the paths, and their corresponding errors compared to the true criticalities.

Table 2. Path criticalities from ATP products

Path	True	Independence	Error
A-C-E	0.375	0.5*0.6268=0.3134	16.4%
B-C-E	0.25	0.5*0.6268=0.3134	25.4%
B-D-E	0.375	1.0*0.3736=0.3736	0.4%

From the above example, we find that the method proposed in [5] does not work correctly on the circuits with the re-convergent fan-outs.

Let us now consider the path B-C-E as an example to see the cause of the errors. Operator $AT(.)$ denotes the node output arrival time. The probability for path B-C-E of being critical is:

$$p_1 = P\{[AT(B) > AT(A)] \cap [AT(C) > AT(D)]\}$$
$$= P[AT(B) > AT(A)] \bullet P[AT(C) > AT(D) \mid AT(B) > AT(A)] \quad (3)$$

while under the independence assumption, the probability is instead calculated as the product of ATP's:

$$p_2 = P[AT(B) > AT(A)] \bullet P[AT(C) > AT(D)] \quad (4)$$

As we know from probability theorems, only when events $AT(C)>AT(D)$ and $AT(B)>AT(A)$ are statistically independent, (3) and (4) are equivalent. Due to the re-convergent fan-out, when $AT(A)>AT(B)$, it is more probable that $AT(C)>AT(D)$, because $AT(C)=d_0+d_C+AT(A)$ and $AT(D)=d_0+d_D+AT(B)$. Therefore, when correlations are present, we have to use conditional probabilities instead of the products of individual ATP's. On the other hand, for paths like B-D-E, because the tightness variables at node D and E are independent, the previous method still results in good accuracy.

3. Criticality calculation

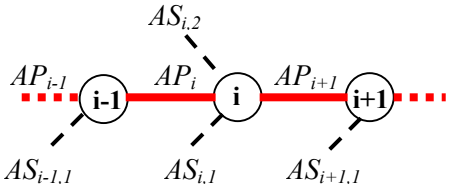

Figure 2 Path criticality

From Section 2, we know that conditional probabilities should be used when the correlations exist. Figure 2 shows a general case for path criticality calculation. If a path with l nodes is considered, AP_i $(1\le i\le l)$ is the arrival time of the on-path signal of the i-th node. We also assume that at this node, there are k_i side-inputs and $AS_{i,j}$ $(1\le i\le l, 1\le j\le k_i)$ is the signal arrival time of its j-th side input.

After a block-based statistical timing analysis is performed, we have arrival times of all nodes as quadratic functions of all the process parameters. Accordingly, the criticality probability can be defined by the following conditions:

$$p = P\{\bigcap_{i=1}^{l}[\bigcap_{j=1}^{k_i}(AP_i > AS_{i,j})]\} \quad (5)$$

$$m = \sum_{i=1}^{l} k_i \quad (6)$$

In (5), the inner intersection operation defines the local conditions at a particular node, which requires the on-path signal to arrive later than all side-input signals. This is consistent with the calculation of the ATP's. The outer intersection operation requires that the local conditions of all nodes along the path to be satisfied at the same time. The total number of conditions in (5) is *m*.

The path criticality calculation turns out to be equivalent to the problem of finding the probability of a sub-space formed by all these *m* conditions. Figure 3 illustrates a two-dimensional (with two process parameters) sub-space example with three hypothetical linear conditions. Because all process parameters are assumed to be normalized Gaussian random variables, the sub-space is defined by the 3σ bounding box and all the conditions. The shaded area indicates the sub-space for the assumed linear conditions:

$$(X_2 - a_1X_1 - b_1 > 0) \cap (X_2 - a_2X_1 - b_2 > 0) \cap (X_2 - a_3X_1 - b_3 > 0) \quad (7)$$

and its probability is what we want to know.

For general cases with *n* process parameters and *m* nonlinear conditions, the sub-space can be similarly represented by a hyper-plane. To derive the sub-space is very complicated and computationally expensive. However, since only the probability of the sub-space is sought, we hereby propose two efficient techniques to solve this problem.

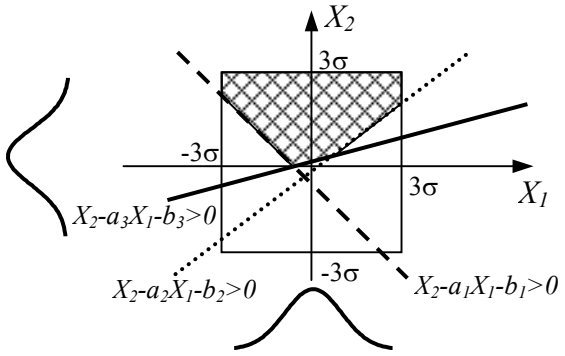

Figure 3 Sub-space example

3.1 *Max* approach

The first proposed solution is to use *max* operation. To use *max* operation, we hereby re-write (5) as:

$$\begin{aligned} p &= P\{\bigcap_{i=1}^{l}[\bigcap_{j=1}^{k_i}(AS_{i,j} - AP_i < 0)]\} \\ &= P[max(AS_{1,1} - AP_1, AS_{1,2} - AP_1, ..., AS_{1,k_1} - AP_1, \\ &\qquad AS_{2,1} - AP_2, AS_{2,2} - AP_2, ..., AS_{2,k_2} - AP_2, \\ &\qquad \bullet\bullet\bullet, \\ &\qquad AS_{l,1} - AP_l, AS_{l,2} - AP_l, ..., AS_{l,k_l} - AP_l) < 0] \end{aligned} \quad (8)$$

Because we have already derived all the *m* conditions as quadratic functions, we can do the *max* operation pair-wise by moment matching techniques [7]. Equations (9)-(12) present the quadratic approximation of the *max* operation, where inputs D_1 and D_2 are quadratic condition functions.

$$D = max(D_1, D_2) = \sum_{i,j} \alpha_{ij} x_i x_j + \sum_i \beta_i x_i + \gamma \quad (9)$$

$$\beta_i = \underset{(x_1,...x_n)}{E}(x_i D) \qquad 1 \le i \le n \quad (10)$$

$$\begin{pmatrix} \alpha_{11} \\ \alpha_{22} \\ \vdots \\ \alpha_{nn} \\ \gamma \end{pmatrix} = \underbrace{\begin{pmatrix} 3 & 1 & \dots & 1 & 1 \\ 1 & 3 & \cdots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 3 & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}^{-1}}_{n+1} \bullet \begin{pmatrix} E(x_1^2 \cdot D) \\ E(x_2^2 \cdot D) \\ \vdots \\ E(x_n^2 \cdot D) \\ E(D) \end{pmatrix} \quad (11)$$

$$\alpha_{ij} = \underset{(x_1,...x_n)}{E}(x_i x_j D) \qquad 1 \le i < j \le n \quad (12)$$

The detailed derivation of the moments $E(D), E(x_iD), E(x_i^2D),$ and $E(x_ix_jD)$ can be found in [7]. Based on this moment matching technique, we are able to fit the result of a *max* operation with two quadratic inputs back to a new quadratic function of the same parameters. After *m*-1 *max* operations over all the *m* conditions, we get the final quadratic function $X^TAX+BX+C$ that represents the condition sub-space, where *A*, *B* and *C* are the fitted quadratic approximation coefficients of the sub-space defined by all the *m* conditions. The path criticality can thus be obtained by numerical integrations over the normalized Gaussian distributed process parameters *X*:

$$p = P(X^T AX + BX + C < 0) \quad (13)$$

As the *max* operation is the core of this approach, both accuracy and efficiency of this solution rely on the *max* operation.

3.2 Monte Carlo integration

The second proposed solution is to use Monte Carlo integration to get the sub-space probability. In (5), we know all the conditions for a given path to become critical from the block-based SSTA. The path criticality is the probability for all the conditions to be satisfied simultaneously. To calculate this criticality, we generate *N* sets of random process parameters and substitute them into (5). A counter will be used to calculate how many process parameter sets satisfy all the conditions. The path criticality is calculated as the final counter value divided by *N*. The accuracy of Monte Carlo Integration is determined by the number of simulations *N*: [9]

$$\int f\, dV \approx V \cdot E(f) \pm V \cdot \sqrt{\frac{E(f^2) - E^2(f)}{N}} \quad (14)$$

where *f* is the function that bounds the sub-space defined by all the *m* conditions, V is the corresponding multidimensional volume, and E(.) is the expectation operator. From (14), the error of Monte Carlo Integration is proportional to $1/\sqrt{N}$. Therefore, a 10,000-point Monte Carlo integration holds the error to the order of 0.01, which usually gives good enough accuracy.

3.3 Algorithm

Both approaches have been integrated into our block-based SSTA tool. The algorithm is illustrated in Figure 4.

The algorithm takes three input variables: *path_set*, the paths to analyze; *mode*, the working mode; and *PT,* probability threshold. *Path_set* is a set of sorted long paths to be evaluated for criticality. They can be the long paths (both critical and non-critical) under nominal delays from STA tools, sorted in a descending order of their delays. The *mode* input indicates whether we use *max* operation or Monte Carlo Integration for

criticality calculation. *PT*, the probability threshold can be interpreted as a performance/cost tradeoff criterion. We evaluate the criticality probability path by path. Once the total criticality of those paths that have already been evaluated reaches *PT*, the algorithm exits. If *PT* is too small, some important paths may be missing during the analysis, while if it is set to 1, we will obtain all statistical critical paths including those that only have tiny probabilities of being critical. This will increase the computational cost. For example, if *PT* is set to 0.95, our algorithm will report the top 95% critical paths. Suppose you have a simple circuit with 10 paths. Among them, path A has a criticality probability of 0.70 and path B has a criticality probability of 0.26, while other paths have small criticalities. When *path_set* includes all the 10 paths, the algorithm terminates exactly when both path A and B have been reported.

```
Statistical_Critical_Path_Analysis (path_set, mode, PT)
{
  Block_Based_SSTA();
  if(mode==Monte Carlo) {
    generate_random_vectors();
  }
  While(path_set!=NULL && total_probability<PT)
  {
    pick_next_path();
    get_all_conditions();
    prob=get_criticality(mode);
    total_probability+=prob;
  }
}
```

Figure 4 Critical path analysis algorithm

After the algorithm is launched, a block-based SSTA algorithm is first applied to get arrival times of all the nodes in a circuit. Then the first path from *path_set* is selected and the conditions for it to be critical are derived at all on-path nodes. On each node, the conditions are that the on-path signal arrives later than all side-input signals. Next, one of the proposed approaches will be used to calculate the probability when all the conditions are satisfied. After one path is evaluated, the next longest path in *path_set* will be analyzed until either all paths are evaluated or the probability threshold *PT* is reached.

In path-based SSTA algorithms, criticality of a certain path is highly dependent to all the other paths being analyzed. If some statistical critical paths are missing, the criticalities of all the analyzed paths will be overestimated. Therefore, unless all the statistical critical paths are considered in the path-based SSTA, the path criticalities obtained are not accurate. Our path criticality analysis is much more robust since path's criticality is independent to the other paths and totally determined by the analysis on itself. Moreover, due to the probability threshold, not all paths in *path_set* are necessary to be evaluated. Therefore, we recommend that *path_set* includes non-critical long paths under nominal delays as well. This is very important especially when we are analyzing large industrial circuits. If due to limited resources, the analysis of all paths can't finish, the outputs of path-based SSTA's are useless. However, in our approach, even partial results are still very useful to circuit designers.

When we apply the two proposed approaches to the example in Figure 1, the corresponding results are shown in Table 3.

Column True lists the correct path criticalities, column Max shows the criticalities from proposed *max* approach and column MC is the result from the proposed Monte Carlo with 10,000-point Monte Carlo integrations. Comparing them to the Independence results in Table 2, we can see that the proposed approaches are much more accurate.

Table 3. Result of our algorithm on the example in Figure 1

Path	True	Max	MC
A-C-E	0.375	0.371	0.379
B-C-E	0.25	0.248	0.246
B-D-E	0.375	0.381	0.375

3.4 Gate criticality

As mentioned earlier, path criticalities can be used in statistical gate sizing for circuit optimizations. The long paths with high criticalities are the statistical critical paths that should be optimized. However, sizing different gates on the same critical path has different impacts on final circuit delays. To look into this problem further, we hereby introduce a measurement of the importance of a particular gate to the final circuit performance: gate criticality. A simple calculation of gate criticalities from statistical critical path criticalities will be discussed as well.

We can group all the gates in a circuit into three categories:

- Gates that are not on any statistical critical paths
  For this type of gates, their criticalities are zero. These gates are always on non-critical paths in all the process space, which means that they have no impact on the final circuit delay under any process environments.
- Gates that are on one statistical critical path
  Their criticalities are the criticalities of the paths passing through them. These gates are only on one statistical path. It means that only when that specific long path becomes critical, will these gates determine the circuit delay. In other parts of the process space, sizing of these gates does not help in improving the circuit performance.
- Gates that are on multiple statistical critical paths
  The criticalities of these gates are the sum of the criticalities of all the paths that pass through them. Intuitively, the more critical paths go through a gate, the more important this gate is.

Upon the definition, gate criticality is a non-negative value from zero to one. If a gate has a criticality of one, all statistical critical paths of the circuit go through this gate. With gate criticalities, circuit designers can size up the highly critical gates or switch from high V_t to low V_t on these gates for better circuit performance.

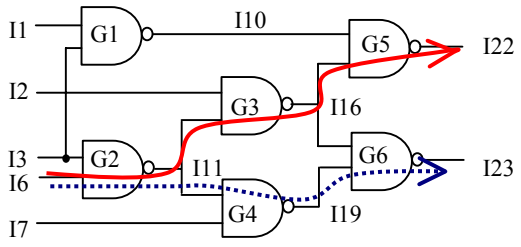

Figure 5 Gate criticalities of circuit C17

Here we use an example to show the gate criticality concept. In Figure 5, C17 circuit from ISCAS85 benchmark is shown. Let us assume that path I6-G2-I11-G3-I16-G5-I22 (the solid line)

and I6-G2-I11-G4-I19-G6-I23 (the pointed line) are the two statistical critical paths in the circuit with criticality of p and q respectively, where $p+q=1$.

Based on our definition of gate criticality, gate G1 belongs to Category 1, which has a zero criticality. Gates G3, G4, G5 and G6 belong to Category 2, with criticalities of p, q, p and q, accordingly. Gate G2 is on both critical paths, thus it has a criticality of $p+q=1$. Therefore, from our analysis, G2 is the first priority for optimization.

4. Experiments

The algorithm in Figure 4 has been integrated into our block-based SSTA tool in C++. In this section, extensive experiments are performed for the ISCAS85 benchmark circuits as well as an industrial circuit M1 with 1K gates, under 90nm process technology conditions. The results from different approaches are compared for both accuracy and efficiency. In the experiments, *PT* is set to 0.95 to get top statistical critical paths up to 95% probability, and 10,000-point simulations are used in our proposed Monte Carlo approach. All experiments were performed on an AMD Athlon MP 2600+ Linux workstation.

Table 4 lists the probabilities of the top statistical critical path of all the experimented circuits. The Independence experiment assumes path independence and calculates path criticalities as the product of the tightness of all the on-path nodes. The Proposed *Max* and the Proposed MC columns show the results from our algorithm described in Section 3. To verify the accuracy of these three methodologies, we performed a true Monte Carlo experiment as well. In this true Monte Carlo experiment, we randomly select 10,000 variational process parameter sets based on the parameters' distributions, and perform Static Timing Analysis under fixed delay models for every condition. In each STA run, we collect the static critical path of the circuit. The top statistical critical paths are those paths that are reported as critical under most given process conditions. The true Monte Carlo has the best accuracy but is too expensive for practical circuits. It is hundreds of times slower than the other three approaches. The CPU runtimes of all four experiments to get up to 95% statistical critical paths are listed in Table 5.

Compared to the True Monte Carlo approach, the Independence, the proposed *Max*, and the proposed Monte Carlo approach have an average error of 8.69%, 1.51%, and 0.39% respectively over all the eleven circuits. When we look more closely at the Independence approach, we find that for some circuits, it is very accurate, but for some others, the errors are as large as 38.37%. Its accuracy is strongly related to circuit topologies. For those circuits with many re-convergent fan-outs and/or strong spatial correlations, such as C6288, the accuracy deteriorates considerably.

If we compare the two proposed approaches, the proposed Monte Carlo approach excels over the proposed *Max* approach in terms of accuracy with similar CPU run times. The reasons can be justified as follows. Based on (14), the accuracy of the proposed Monte Carlo approach is only determined by the number of Monte Carlo integrations N, and irrelevant to the number of process parameters involved. On the contrary, the accuracy of the proposed *Max* approach relies on the number of process parameters as well as the number of conditions. Although we have used a non-linear approximation for the *max* operation, errors may still accumulate during the *max* operations of conditions, especially when we have a large number of conditions. Unfortunately, this is often the case in large circuits. The results in Table 6 can justify this issue further. Table 6 shows the criticalities of all the statistical critical paths up to 95% probability threshold of benchmark circuit C2670. The true Monte Carlo experiment and the proposed Monte Carlo approach both report four statistical critical paths while the proposed *Max* approach reports five. The proposed Monte Carlo approach gets very good results on all top critical paths while the *max* approach doesn't do very well on the third and fourth critical paths. That is because, for shorter paths, the arrival times of on-path signals and side-inputs become closer. This can cause the mean values of the *max* operands, $AS_{i,j} - AP_i$, to be very close, which is the case when *max* operation suffers from largest errors in approximations. Of course, the smaller criticalities on shorter paths also make the relative errors look larger. Nevertheless, we want to point out that the reported statistical critical paths from both proposed approaches match the ones from the true Monte Carlo approach, which gives circuit designers and test engineers correct set of paths to focus on. Also, if higher order *max* operation is used, the proposed *max* approach should show much better accuracy.

The complexities of both proposed approaches of calculating a single path criticality are linear with respect to the number of conditions m. The number of conditions of a circuit is bounded by the number of the gate fan-ins times the path length. In modern digital circuits, the number of gate fan-ins is typically less than 4, and the path length is less than 20. This guarantees the efficiency of our approaches. Compared between the two proposed approaches, the *max* approach needs m-1 *max* operations, and the complexity of *max* operation is linear to the number of process parameters and super linear to the number of sampling points for numerical calculations [7]. On the contrary, the complexity of the proposed Monte Carlo approach is linear to the number of simulations, which is 10,000 in our experiments. Moreover, when the proposed Monte Carlo approach verifies the conditions, once a condition is violated, it just skips all the other conditions and goes on with the next Monte Carlo simulation. This makes the approach very efficient.

5. Conclusions

In this work, we proposed and developed two efficient statistical critical path analysis approaches for block-based SSTA tools. Our algorithm reports good results for statistical critical path analysis in circuits with arbitrary correlations caused by re-convergent fan-outs and process spatial correlations. They enable block-based algorithms to get accurate critical paths with little computing overhead. The algorithm can be added to any block-based SSTA tools as a post-processing step and benefit both circuit designers and test engineers.

Compared to the path-based SSTA algorithms, our approaches are much more robust and efficient in calculating path criticalities. In path-based SSTA, if we don't include all long paths in the analysis, the accuracy of all criticalities will be impaired. However, if we analyze a large number of paths, the computational cost increases considerably, since before all paths are evaluated, we do not know the criticality of any path. In our approaches, even if not all statistical critical paths are considered, the criticalities of those analyzed paths are still accurate. Moreover, a large path set usually does not hurt the speed, because most non-critical paths will not be analyzed due to the probability threshold.

6. Acknowledgement

This work was supported by the MARCO Focus Center for Circuit & System Solutions (C2S2, www.c2s2.org) under contract 888 and Advanced Micro Devices.

7. Reference

[1] A. Gattiker, S. Nassif, R. Dinakar and C. Long, "Static Timing Analysis Based Circuit-Limited-Yield Estimation", IEEE International Symposium on Circuits and Systems, 2002. Volume 5, 26-29 May 2002

[2] M. Orshansky and K. Keutzer, "A General Probabilistic Framework for Worst Case Timing Analysis", Proc. DAC, pp 556-561, June 2002

[3] J. A. G. Jess and K. Kalafala et al, "Statistical timing for parametric yield prediction of digital integrated circuits", Proc. DAC, pp. 932-937, June 2003

[4] H. Chang, S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal", IEEE ICCAD, pp. 621-625 November 2003

[5] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, "First-Order Incremental Block-Based Statistical Timing Analysis", Proc. 2004 DAC, pp. 331-336, June 2004

[6] J. Le, X. Li, L. T. Pileggi, "STAC: Statistical Timing Analysis with Correlation", Proc. 2004 DAC, pp. 343-348, June 2004

[7] Y. Zhan, A. J Strojwas, X. Li, L. T. Pileggi, D. Newmark, M. Sharma, "Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions", Proc. 2005 DAC, June 2005

[8] H. Chang, V. Zolotov, S. Narayan, C. Visweswariah, "Parameterized Block-Based Statistical Timing Analysis with Non-Gaussian Parameters and Nonlinear Delay Functions", TAU workshop, 2005

[9] J. M. Hammersley and D. C. Handscomb, Monte Carlo Methods, Methuen's monographs, London, 1964

Table 4. Probabilities of the top statistical critical path

Circuit	True MC	Independence		Proposed Max		Proposed MC	
	Probability	Probability	Error (%)	Probability	Error (%)	Probability	Error (%)
C432	0.8682	0.8147	-6.16	0.8696	0.16	0.8686	0.05
C499	0.8603	0.7591	-11.76	0.8664	0.71	0.8611	0.09
C880	0.6966	0.6849	-1.68	0.6844	-1.75	0.6953	-0.19
C1355	0.8108	0.7637	-5.81	0.8241	1.64	0.8148	0.49
C1908	0.8723	0.8810	1.00	0.8874	1.73	0.8739	0.18
C2670	0.3556	0.3790	6.58	0.3504	-1.46	0.3587	0.87
C3540	0.9963	0.9991	0.28	1.0000	0.37	0.9967	0.04
C5315	0.8279	0.8119	-1.93	0.8069	-2.54	0.8253	-0.31
C6288	0.3706	0.2284	-38.37	0.3689	-0.46	0.3720	0.38
C7552	0.3805	0.4000	5.12	0.3603	-5.31	0.3870	1.71
M1	0.5502	0.4575	-16.85	0.5478	-0.44	0.5501	0.02

Table 5. CPU run time of getting top 95% statistical critical paths

Circuit	True MC	Independence	Proposed Max	Proposed MC
C432	845.58	2.90	6.39	3.01
C499	978.6	1.81	5.07	2.03
C880	1357.01	1.88	3.70	3.65
C1355	2455.24	4.72	4.92	4.84
C1908	3221.11	3.15	3.90	3.28
C2670	4188.79	5.89	11.74	6.73
C3540	6102.67	6.85	7.51	6.78
C5315	9766.77	22.23	25.23	22.77
C6288	11690.84	22.03	28.49	27.80
C7552	12830.86	20.21	19.99	20.19
M1	4505.38	11.80	12.01	12.07

Table 6. Criticalities of top 95% statistical critical paths of C2670

Path Rank	True MC	Proposed Max		Proposed MC	
	Criticality	Criticality	Error (%)	Criticality	Error (%)
1	0.3556	0.3504	-1.46	0.3587	0.87
2	0.3471	0.3305	-4.78	0.3476	0.14
3	0.2291	0.2056	-10.26	0.2271	-0.87
4	0.0340	0.0457	34.41	0.0326	-4.12
5	N/A	0.0182	N/A	N/A	N/A