A Hierarchical Technique for Statistical Path Selection and Criticality Computation

CHITHIRA P. R., Indian Institute of Technology Madras
VINITA VASUDEVAN, Indian Institute of Technology Madras

Due to process variations, every path in the circuit is associated with a probability of being critical and a measure of this probability is the criticality of the path. Identification of critical paths usually proceeds in two steps, namely, generation of a candidate path set followed by computation of path criticality. As criticality computation is expensive, the candidate path set is chosen using simpler metrics. However, these metrics are not directly related to path criticality and often,the set also contains low criticality paths that do not need to be tested. In this paper, we propose a hierarchical technique which directly gives all paths above a global criticality threshold. The circuit is divided into disjoint groups at various levels. We show that the criticality of a group at each level of hierarchy can be computed using criticality of the parent group and the local complementary delay within the group. Low criticality groups are pruned at every level, making the computation efficient. This recursive partitioning and group criticality computation is continued until the group criticality falls below a threshold. Beyond this, the path selection within the group is done using branch-and-bound algorithm with global criticality as the metric. This is possible, since our method for criticality computation is very efficient. Unlike other techniques, path selection and criticality computation are integrated together so that when the path selection is complete, path criticality is also obtained. The proposed algorithm is tested with ISCAS'85, ISCAS'89 and ITC'99 benchmark circuits and the results are verified using Monte Carlo simulation. The experimental results suggest that the proposed method gives better accuracy on average with around 90% reduction in run-time.

CCS Concepts: • **Hardware** → **Electronic design automation**; **Statistical timing analysis**; *Hardware test*;

Additional Key Words and Phrases: Statistical timing, path selection, path criticality, hierarchical partitioning, at-speed test

ACM Reference format:
Chithira P. R. and Vinita Vasudevan. 0000. A Hierarchical Technique for Statistical Path Selection and Criticality Computation. *ACM Trans. Des. Autom. Electron. Syst.* 0, 0, Article 0 (0000), 25 pages.
https://doi.org/0000001.0000001

Authors' addresses: Chithira P. R. and Vinita Vasudevan, Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai - 600036; emails: [ee13d027, vinita]@ee.iitm.ac.in.

1084-4309/0000/0-ART0 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

With increasing process variations, deterministic delay models have become inadequate for identification of critical paths. Every path is associated with a probability of being critical in some die and a measure of this probability is the criticality of the path. The path criticality therefore is indicative of the percentage of dies in which the path has the maximum delay and can be used as a measure for both timing optimization and selection of paths for delay fault-testing. However, computation of path criticality is time-consuming and unless done carefully, is also error prone. Typically a two step procedure is used, wherein a candidate set of paths is chosen using a simpler metric. This is followed by computation of path criticality.

The simplest metric for path selection is the delay of the path obtained using deterministic values, possibly at the process corners. Zhan *et al.* [18] use static timing analysis (STA) to obtain paths with the largest delays. They propose the use of either statistical MAX operation or Monte Carlo methods for the computation of path criticality. Monte Carlo methods are accurate but expensive. The use of Clark's approximation of the MAX operator leads to significant errors in large circuits with many similar paths. The method proposed in [8] uses a two step process for path selection. The first step is candidate path set generation where the longest sensitizable paths with delay greater than a specified value is obtained using STA. This is followed by path selection from the candidate set taking into account the correlation among the paths. In this paper, the criticality of the selected paths is not computed.

The other metric used for path selection is the test quality metric (TQM) which is defined as the probability that a tested chip has no timing violation, given that the test paths pass the at-speed test [20, 21]. Using this metric turns out to be equivalent to maximising the probability that the statistical path slack of the set of paths is less than zero (referred to as the process coverage metric). Zolotov *et al.* propose a branch-and-bound (BnB) technique using TQM as the metric. Xiong *et al.* propose a path selection technique based on multi-layer process coverage [11, 16]. They define the test quality metric in [20] to be a single layer metric in the sense that any point in the process space will be covered if it is covered by at least one of the paths and if the selected path is not sensitizable, there will be a resulting loss of process space coverage. With multi-layer TQM (mTQM), a point in the process space will be covered only if it is covered by at least m paths so that even if some paths are not sensitizable, it is likely to be covered by other paths. They use an m^{th} order statistic for computing mTQM and a branch-and-bound algorithm along with this metric for path selection. To account for the testability of the selected paths, Chung *et al.* proposed a branch-and-bound path selection using testable path coverage metric (TCM) [5]. This approach uses local criticality within a set a testable paths as the metric. Note that the local criticality of a path is the criticality within a subset of paths and the global criticality is the criticality with respect to all the paths in the circuit. The local criticality is an upper bound on the global criticality and approaches the global criticality only asymptotically. Therefore, the number of paths that are are inspected needs to be larger than the optimal solution. Moreover, whenever a new path is obtained, the criticality within the path set needs to be recomputed and paths are discarded based on the newly computed criticality.

In [2, 3], Chung *et al.* propose a recursive path selection which aims to select a set of paths so that the fault detection probability is maximized. The fault detection probability is also related to the statistical path slack and this approach attempts to select a set of paths of fixed size so that the probability that the path set delay exceeds the clock period is maximized. This approach assigns a budget of k to the sink node where k is the number of

paths to be selected. The graph is traversed from primary outputs to primary inputs and budgets are assigned to the nodes in such a way that the detection probability is maximized. The recursive traversal is continued until the budget drops to zero or the source node is reached.

The global criticality of a path is a measure of the percentage of dies in which the path has the maximum delay. It is an indicator of the timing yield in the following sense. If a path has a global criticality of 0.9, a successful at-speed test of this path indicates that the timing yield will most likely be greater than 90%. The slack based metric, defined in [20] has positive values for all paths and the top N paths are chosen for at-speed testing. This has two issues. Firstly, it is not clear how to determine N. Secondly, it is not clear how many, among these N paths need to be tested. Consider the following case. Assume there are two paths, both of which have a large TQM. However, due to large correlation, one of the paths dominates the other so that it has a much larger global criticality than the other. Depending on the actual values of the global criticality and required timing yield, it may not be necessary to do an at-speed test of both paths. For example if the global criticalities are 0.95 and 0.05, testing of the dominant path alone could be sufficient. On the other hand, if there are twenty uncorrelated paths with criticality 0.05, many more paths have to be tested. TQM can also be extended to include correlation using joint path metric (JPM) as in [20]. However, it is based on replacing each path in the current list with a new path and recomputing the metric. Since it is not clear how many new paths need to be tested, the method turns out to be expensive [20]. Also, even when all the paths selected are tested, the issue of estimating the timing yield remains. The local criticality in [5] is a better metric, but as discussed, it is not a good bound until the path set is large enough. On the other hand, if we could directly obtain paths based on global criticality, we can stop looking for new paths once the sum of the global criticalities of the selected paths are as close to one as desired. *In the rest of the paper, we use the terms criticality and global criticality interchangeably. Local criticalities, when used, will be explicitly indicated.*

The biggest problem with using criticality is that it is time-consuming to evaluate and unless done carefully, the evaluation is error prone. Hence we need accurate and efficient methods to evaluate criticality. In large circuits with many similar paths, the error in MAX operation is significant leading to errors in both circuit delay and the criticality. With a careful combination of pruning and ordering of MAX operations, the node/edge criticality computations can be made reasonably accurate [9, 10]. Path criticality computation is more challenging, since it is not possible to enumerate all paths in the circuit. This means that pruning of highly correlated similar paths is possible only to a limited extent, leading to errors. Experiments in [4] indicate that using the conditional probability method after removing topological correlation gives a lower error. But this paper does not discuss how to find the paths with the largest criticalities in the circuit.

Several techniques have been proposed which use global node and edge criticality for path enumeration and criticality computation [14, 15]. The method proposed in [14] computes node criticality and uses breadth first search along with node criticality for path enumeration. The path criticality is evaluated as the product of the arrival tightness probabilities of the edges along the path. But this method assumes independence between the edge delays, which is not a good assumption. Wang *et al.* [15] propose a path criticality computation method that uses conditional probability to take into account the correlations between the delays. They use the node and edge criticality as a metric and propose a "branch and prune" algorithm, where low criticality nodes and edges are pruned in backward traversal of the

circuit graph. However, it requires a full criticality computation of the nodes and edges of the graph. While this is efficient when there are dominant paths that have a large criticality, it is very time consuming if there are a large number of parallel paths. Their paper does not contain actual results of paths enumerated and the corresponding criticality, so the results cannot be compared.

Ideally we would like a method in which the global criticality of the path is used as a metric and the selection of paths is not decoupled from the criticality evaluation. In this paper, we propose a hierarchical method to obtain all paths that have global criticality above a specified threshold. At each level of the hierarchy, a group is partitioned into a set of disjoint groups at the next level. This is followed by pruning and criticality evaluation of the "next level" groups. We show that the criticality of each group can be computed using criticality of the parent group and a conditional local criticality within the parent group. The evaluation can be done using either a recursive technique or statistical MAX and conditional probability or a combination of both. Only those groups that have criticality above the threshold are further subdivided. Finally, when the criticality of the group drops below a threshold, we use a branch-and-bound method for path selection using the global criticality of a sub-path as the metric. As a result, the path selection and path criticality evaluation proceed simultaneously. It is also efficient as the path selection works on a smaller subset of nodes. At every level, we also place groups that have a large topological correlation and nearly identical means and standard deviations into clusters. Only one of the groups is then considered for further subdivision and criticality evaluation.

The rest of the paper is organised as follows. Section 2 explains the circuit model used. Section 3 describes some of the existing path criticality computation approaches and their drawbacks. Section 4 discusses the statistical path selection using BnB and its limitations. The proposed hierarchical partitioning algorithm is explained in Section 5. Section 6 contains implementation details and complexity analysis. Section 7 presents the results obtained for various benchmarks and section 8 concludes the paper.

2 MODEL

In timing analysis, the digital circuit is transformed into a directed acyclic graph $G(V, E)$ where V is the set of vertices and E is the set of edges. Each net/signal in the design is represented as a vertex and the gate delays represent the weights associated with the edges. A virtual source node is connected to all the primary inputs and all the primary outputs are connected to a virtual sink node. The gate delays and all the timing quantities are expressed in canonical delay format. The general form of canonical delay model is,

$$d = d_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_R \Delta R$$

where d_o is the nominal value, ΔX_i represents the i^{th} source of variation, a_i represents the corresponding sensitivity and a_R represents the sensitivity to the random component (ΔR) of parameter variation [1, 14].

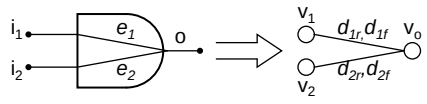


Fig. 1. Gate model

Figure 1 shows a gate with two edges e_1 and e_2. Each edge is associated with a rise delay and fall delay. Both the rise and fall delays, d_{ir} and d_{if}, corresponding to edge e_i are expressed in the canonical form. The arrival time (AT) at any node denotes the latest time at which the signal becomes stable at that node. A block based SSTA is used to find AT at each node. The graph is traversed in topological sort order. At each level, the AT distribution of rise and fall transitions at each node is computed using AT of its predecessors and the delay of the edges incident at that node. The AT at gate output o can be computed as follows. If the gate is positive unate,

$$\begin{aligned} AT_{o,r}(pu) &= \max\{AT_{v1,r} + d_{1r}, AT_{v2,r} + d_{2r}\} \\ AT_{o,f}(pu) &= \max\{AT_{v1,f} + d_{1f}, AT_{v2,f} + d_{2f}\} \end{aligned} \tag{1}$$

If the gate is negative unate,

$$\begin{aligned} AT_{o,r}(nu) &= \max\{AT_{v1,f} + d_{1r}, AT_{v2,f} + d_{2r}\} \\ AT_{o,f}(nu) &= \max\{AT_{v1,r} + d_{1f}, AT_{v2,r} + d_{2f}\} \end{aligned} \tag{2}$$

If the gate is both positive unate and negative unate,

$$\begin{aligned} AT_{o,r} &= \max\{AT_{o,r}(pu), AT_{o,r}(nu)\} \\ AT_{o,f} &= \max\{AT_{o,f}(pu), AT_{o,f}(nu)\} \end{aligned} \tag{3}$$

The AT computation requires SUM and MAX operation to be performed on a set of Gaussian random variables expressed in canonical form. The sum of a set of Gaussian random variables is also a Gaussian random variable. But MAX is a non-linear operator and Clark's formula is used to approximate it as a linear operation [6]. The path delay is obtained as the sum of its edge delays. Both the rise and fall transitions are propagated to the sink node taking into account the unateness of the gate. The path delay distribution is obtained as the statistical MAX of rise and fall delays at the sink node.

3 BACKGROUND: METHODS FOR PATH CRITICALITY COMPUTATION

Statistical criticality of a path is defined as the probability that the path in the circuit is a critical path. Assume that there are N paths in the design and let D_i denote the delay distribution of the i^{th} path. The criticality of this path, $Q_{g,i}$, can be written as

$$\begin{aligned} Q_{g,i} &= P\left(D_i > \max_{\substack{1 \le j \le N \\ j \ne i}} \{D_j\}\right) \\ &= P(D_i > CD_{g,i}) \end{aligned} \tag{4}$$

$CD_{g,i}$ denotes the global complementary delay distribution of path i. Note that the circuit delay $D_c = \max\{D_i, CD_{g,i}\}$. If all the variations are assumed to be normally distributed, then $Q_{g,i}$ is the tightness probability of path delay (PD) over complementary path delay (CPD) [1]. The following techniques have been used to compute the path criticality.

3.1 Using Statistical Reversible MAX Operation or Circuit Delay

The reversible MAX (RMAX) operation attempts to reconstruct the CPD using the PD and circuit delay distribution [12]. The PD and the CPD can then be used to compute the path criticality. In the circuit delay technique, an alternate measure of criticality is computed using the circuit delay [10]. A look up table is then used to obtain the conventional criticality measure. The two techniques are related and suffer from common pitfalls. There are two dominant sources of error namely,

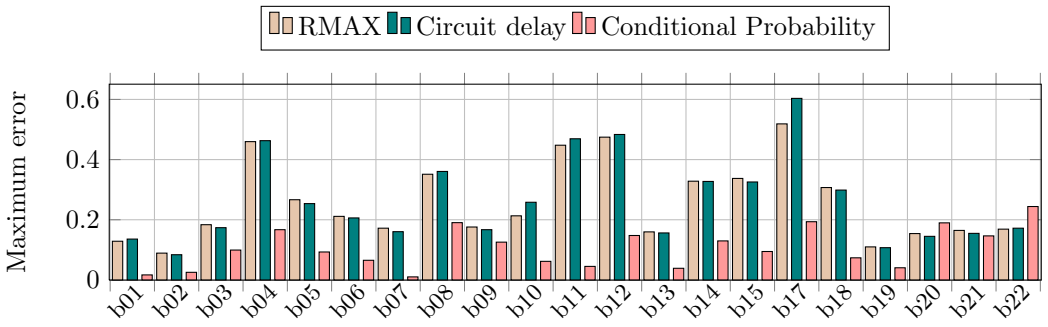


Fig. 2. Comparison of maximum error in criticality for various path criticality computation approaches in ITC'99 benchmark circuits

(1) Error in the arrival times due to correlation errors arising from reconvergent fanouts. The extended canonical form [19] is used to overcome this error.
(2) Errors due to very high correlation between the paths. This is referred to as the '*abc* problem' [9]. The effect of this error is reduced using pruning techniques [9, 10].

Figure 2 contains the maximum error in path criticality using RMAX and the circuit delay method for various benchmarks. The BnB-SPM algorithm [20] was used to extract ten paths that had the largest value of the process coverage metric. The BnB-SPM algorithm uses TQM as the metric for path selection and this will be referred to as BnB-TQM in the subsequent sections of the paper. The extended canonical model was used to represent the ATs and pruning techniques were used to mitigate the effect of high correlation within the path set. However, the maximum error continues to be as high as 0.6. One portion of this error is due to the large number of statistical MAX operations that are needed when the number of topological levels and/or output nets is large. The other source of error is because pruning is only partially effective for path criticality computations.

3.2 Conditional probability method

Chung *et al.* [4] attempt to improve the accuracy of the path criticality computation by reducing the effects of topological correlation and using a combination of MAX and conditional operations. The circuit graph is partitioned into a set of $t+1$ disjoint groups where $(t+1)$ is the number of nodes in the path whose criticality is to be determined. This path is contained in group G_0 and groups $\{G_1 \cdots G_t\}$ contain the paths that contribute to the complementary path delay. The authors propose a recursive algorithm that has at each step, a MAX operation involving a much smaller set of random variables and a transformation into conditional random variables. Elaborating on this, let $p = < v_0, \cdots, v_t >$ be the path under consideration. The first group G_0 contains the path p itself. Group G_1 contains all the paths that contain nodes $< v_1, \cdots, v_t >$ excluding the path in G_0. In general, group G_k contains all the paths that have the nodes $< v_k, v_{k+1}, \cdots, v_t >$, other than the paths already present in the groups G_0 to G_{k-1}. Note that the groups G_0 and G_k have the sub-path $< v_k, v_{k+1}, \cdots, v_t >$ in common and hence their delays are topologically correlated.

The criticality computation is done after removing this topological correlation. If D_i is the delay associated with group G_i, i.e. the maximum of the delays of all paths contained in

group G_i, then the path criticality Q can be written as

$$Q = P(D_0 \geq \max(D_1, \cdots D_t)) = P(\bigcap_{k=1}^{t} D_0 \geq D_k)$$
$$= \prod_{k=1}^{t} P(A_k \geq B_k | S_{k-1}) = \prod_{k=1}^{t} P(A_{c,k} \geq B_{c,k}) \quad (5)$$

where A_k is the delay of the sub-path $< v_0, v_1, ...v_k >$ in group G_0, B_k is the maximum of delay of all paths in group G_k up to node v_k, $S_{k-1} = A_1 \geq B_1, A_2 \geq B_2, ..., A_{k-1} \geq B_{k-1}$. $A_{c,k}$ and $B_{c,k}$ are conditional random variables given by $A_{c,k} = A_k | S_{k-1}$ and $B_{c,k} = B_k | S_{k-1}$ respectively. This process of partitioning and criticality computation is repeated for each path for which criticality is required.

In this method, the authors recommend removal of topological correlation due to the common sub-path to improve the accuracy. However, if the model contains both rise and fall time, removing the common sub-path gives rise to significant errors. Therefore, in our implementation, we have propagated both rise and fall times to the sink node. Figure 2 contains the error in the path criticality. It is seen that there is a significant reduction in the error when conditional probability is used to evaluate criticality, even though we have not removed topological correlation.

4 BACKGROUND: METHODS USED FOR PATH SELECTION

The two methods used in the literature for path selection are (a) use the K longest paths identified using STA and (b) use a branch-and-bound algorithm along with a test quality metric (TQM) or joint path metric (JPM) [20] and identify K paths that have the largest value of the metric. All methods, including using STA to find the longest path, work well when the circuit has a dominant path with large criticality and several uncorrelated paths with smaller criticality. The differences occur when there are many paths with significant criticality and not one among them is dominant or when there are several highly correlated paths that have similar means and variances. It then becomes difficult to fix a value for K or even choose a threshold for the metric in the branch and bound algorithm. In order to get all such paths, we need a K value that is much larger than required.

Ckt	ΣQ_{mc}	
	$K = 10$	$K = 50$
b17	0.6887	0.8710
b22	0.4511	0.7542

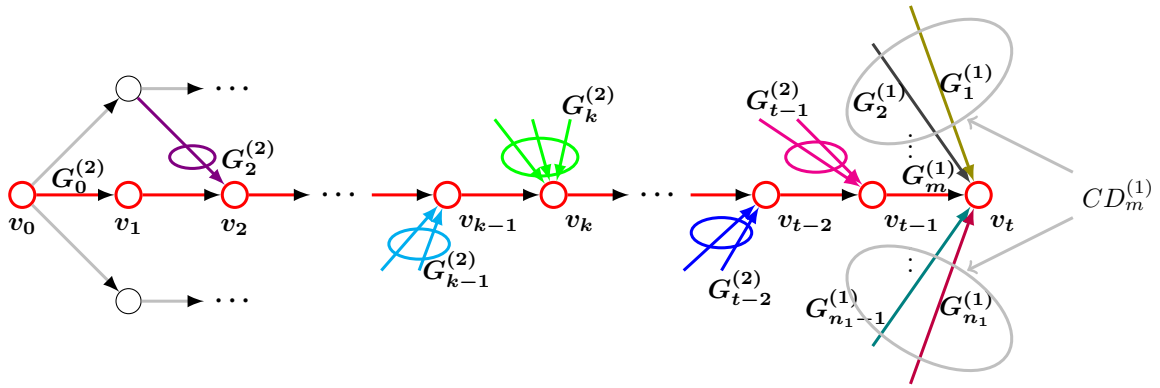
Table 1. Sum of Monte Carlo criticality(ΣQ_{mc}) for top K paths in the design based on TQM

This is illustrated in the results contained in Table 1 for two benchmarks b17 and b22. The table contains the results of top 10 and top 50 paths reported by the branch and bound algorithm with TQM as the metric. The results show that there is a large increase in criticality coverage when K is increased from 10 to 50 for both the circuits. Upon investigation, it is found that there is a path with criticality of 0.11 in b17 circuit and 0.21 in b22 circuit which are not reported if we set the value of K to ten. In both cases, the 10^{th} path reported has zero criticality. For b17, TQM of the 10^{th} path is 0.48 whereas the TQM of the path with criticality 0.11 is 0.47 and it is the 34^{th} path. Similarly for b22, the TQM of the 10^{th} path is 0.28 whereas the path with criticality 0.21 has a TQM of 0.25 and it is the 42^{nd} path.

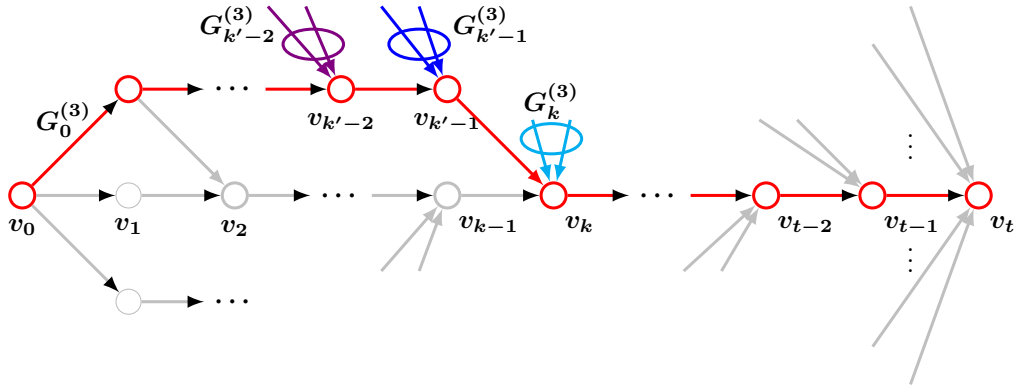
5 HIERARCHICAL ALGORITHM

In this section, we propose a hierarchical technique to identify all the paths that have criticality above a specified threshold. *As mentioned, by criticality, we mean the global criticality of the path or the group.* Each level of hierarchy consists of partitioning of groups into disjoint subgroups, K-centre pruning and clustering of similar groups, computation of group criticalities and removal of groups that have criticality below the threshold. For each level other than the first level, the partitioning into disjoint subgroups is a generalisation of the method in [4]. Instead of finding a distinct disjoint partition of the circuit corresponding to each path, we use the method to recursively divide each group into disjoint subgroups. We also prove that the group criticality at every level can be computed using criticality of its parent group and local complementary delay within the parent group. Since the computation turns out to be efficient, we can directly use the criticality of a set of paths as the metric in the branch and bound algorithm for path selection at the final level.

5.1 Group partitioning and criticality computation



(a) Level 1 partitioning based on output net and level 2 partioning of m^{th} group into subgroups



(b) Level 3 partitioning of k^{th} group into subgroups

Fig. 3. Hierarchical partitioning of circuit into disjoint subgroups

The partitioning technique is illustrated in Figure 3. Figure 3(a) shows how the first two levels are obtained. Level 1 partitioning is done based on the output net. Each of the level 1 groups is subdivided further using the technique in [4]. Within the level 1 group, a path is first identified (it can be any path, but we use the longest path within the group identified using STA). This path is the first group at the next level (level 2) and is denoted as $G_0^{(2)}$ in the figure. The other level 2 groups within the top level group are obtained based on this path, as described in section III. If the path has $t+1$ nodes, t level 2 groups are formed within each level 1 group. The level 2 groups are similarly subdivided into level 3 groups as shown in Figure 3(b). Since all the groups at each level are disjoint, the following property holds for all groups at any level.

Property 1: The sum of the criticalities of the subgroups (groups at level $(i+1)$) within a parent group (at level i), is equal to the criticality of the parent group i.e.,

$$\sum_{k=1}^{N_s} Q_{g,k}^{(i+1)} = Q_g^{(i)} \tag{6}$$

Here N_s is the number of disjoint partitions of the i^{th} level group. Since criticality of any group (and hence path) can never exceed the criticality of its parent group, further subdivision is done only for groups with criticality above a threshold. The criticality is computed as follows. Let

$D^{(i)}$: Delay of a group at the i^{th} level

$CD^{(i)}$: Complementary delay of a group at the i^{th} level within the parent group

$Q_l^{(i)}$: Local criticality (within the parent group) of the group at the i^{th} level

Therefore,

$$\begin{aligned} D^{(i)} &= \max(D^{(i+1)}, CD^{(i+1)}) \\ CD_g^{(i)} &= \max(CD^{(i)}, CD^{(i-1)}, \cdots, CD^{(1)}) = \max(CD^{(i)}, CD_g^{(i-1)}) \\ Q_l^{(i)} &= \mathrm{P}(D^{(i)} > CD^{(i)}) \\ Q_g^{(i)} &= P(D^{(i)} > CD_g^{(i)}) \end{aligned}$$

Here, $CD_g^{(i)}$ is the global complementary delay and $Q_g^{(i)}$ is the global criticality of the group at the i^{th} level. Based on this, the criticality of a group at the i^{th} level of hierarchy can be written as

$$Q_g^{(i)} = P(D^{(i)} > CD_g^{(i)}) = P(D^{(i)} > CD^{(i)} | D^{(i-1)} > CD_g^{(i-1)}) Q_g^{(i-1)} \tag{7}$$

Proof. Let $\{G_1^{(i)}, \cdots, G_{N_s}^{(i)}\}$ be a disjoint partition of a group at level $(i-1)$, i.e. no path is a member of more than one group and the set contains all paths of the parent group. X be the set of random variables $\{D_1^{(i)}, D_2^{(i)}, \cdots, D_{N_s}^{(i)}, CD_g^{(i-1)}\}$. Here $D_k^{(i)}$ is the delay of the group $G_k^{(i)}$. Let $R_k^{(i)}$ denote the set where $D_k^{(i)}$ has the largest value among all the entries in X. Therefore, the global criticality of the group $G_k^{(i)}$ is given by

$$Q_{g,k}^{(i)} = P(R_k^{(i)}) \tag{8}$$

If $R^{(i-1)}$ is the set where one of $D_1^{(i)}$ to $D_{N_s}^{(i)}$ has the highest value among the entries in X, the global criticality of the parent group can be written as,

$$Q_g^{(i-1)} = P(R^{(i-1)}) \tag{9}$$

Now, if $R_k^{'(i)}$ is the set for which $D_k^{(i)}$ is larger than the other delays $D_j^{(i)}$, the local criticality of the group G_k within the parent group is given by

$$P(R_k^{'(i)}) = P\left(D_k^{(i)} > \max_{\substack{1\leq j\leq N_s \\ j\neq k}} \{D_j^{(i)}\}\right) = P(D_k^{(i)} > CD_k^{(i)})$$

Clearly,

$$R_k^{(i)} = R_k^{'(i)} \cap R^{(i-1)} \tag{10}$$

Therefore,

$$\begin{aligned} Q_{g,k}^{(i)} &= P(R_k^{'(i)} \cap R^{(i-1)}) \\ &= P(R_k^{'(i)} | R^{(i-1)}) \times P(R^{(i-1)}) \\ &= P(D_k^{(i)} > CD_k^{(i)} | D^{(i-1)} > CD_g^{(i-1)}) Q_g^{(i-1)} \end{aligned}$$

□

An alternative version of this result can be written as follows.

$$Q_g^{(i)} = P(\bigcap_{k=1}^{i} D^{(k)} > CD^{(k)}) \tag{11}$$

Proof. By definition,

$$\begin{aligned} Q_g^{(i)} &= P(D^{(i)} > \max(CD^{(i)}, CD^{(i-1)}, \cdots CD^{(1)})) \\ &= P(D^{(i)} > CD^{(i)} \ \& \ D^{(i)} > \max(CD^{(i-1)}, \cdots CD^{(1)})) \end{aligned}$$

Using the equivalent of equation (10), we get

$$\begin{aligned} Q_g^{(i)} &= P(D^{(i)} > CD^{(i)} \ \ \& \ \ \max(D^{(i)}, CD^{(i)}) > \max(CD^{(i-1)}, \cdots CD^{(1)})) \\ &= P(D^{(i)} > CD^{(i)} \ \ \& \ \ D^{(i-1)} > \max(CD^{(i-1)}, \cdots CD^{(1)})) \\ &= P(D^{(i)} > CD^{(i)} \ \ \& \ \ D^{(i-1)} > CD^{(i-1)} \ \ \& \ \ \max(D^{(i-1)}, CD^{(i-1)}) > \max(CD^{(i-2)}, \cdots CD^{(1)})) \end{aligned}$$

This can be continued until we get

$$\begin{aligned} Q_g^{(i)} &= P(D^{(i)} > CD^{(i)} \ \& \ \cdots \ \& \ D^{(1)} > CD^{(1)}) \\ &= P(\bigcap_{k=1}^{i} D^{(k)} > CD^{(k)}) \end{aligned}$$

□

The joint probability can be written in terms of conditional probability as follows

$$\begin{aligned} Q_g^{(i)} &= P(\bigcap_{k=1}^{i} D^{(k)} > CD^{(k)}) \\ &= \prod_{k=1}^{i} P\big(D^{(k)} > CD^{(k)} | S_{k-1}\big) \end{aligned} \tag{12}$$

where,

$$S_{k-1} = D^{(1)} > CD^{(1)}, D^{(2)} > CD^{(2)} \cdots D^{(k-1)} > CD^{(k-1)}$$

Both formulations, equations (7) and (12), require the evaluation of conditional random variables which can be done using the following result proved in [4].
Let X, Y, T and U be normally distributed random variables with mean values μ_1, μ_2, μ_3 and μ_4 respectively and with standard deviations $\sigma_1, \sigma_2, \sigma_3$ and σ_4 respectively. Let cov(X,Y)=ρ, cov(X,T) = ρ_1, cov(Y,T) = ρ_2, cov(X,U) = ρ_3, cov(Y,U) = ρ_4 and cov(U,T) = ρ_5. Then

$$E[T|X > Y] = \mu_3 + \beta(\rho_1 - \rho_2)/a$$
$$cov(T, U|X > Y) = \rho_5 - (\beta^2 + \alpha\beta)(\rho_1 - \rho_2)(\rho_3 - \rho_4)/a^2$$

where,

$$a = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho}$$
$$\alpha = (\mu_1 - \mu_2)/a$$
$$\beta = \frac{\phi(\alpha)}{\Phi(\alpha)}$$

If the formulation in (7) is used, we explicitly evaluate all the group criticalities at various levels of hierarchy. This involves one more MAX operation at every level of the hierarchy to find $CD_g^{(i-1)}$ and the evaluation of the tightness probability using conditional random variables. The recursive method proposed in [4] can be used to compute the criticality using (12). While the recursive algorithm is supposed to have a better accuracy than using additional MAX operators, it is computationally not very efficient if the extended canonical form is used. An alternative could be to find CD_g up to the k^{th} level and then use the recursive formula as follows.

$$Q_g^{(i)} = \left(\prod_{j=k+1}^{i} P\left(D^{(j)} > CD^{(j)} | S_{j-1}\right) \right) \times Q_g^{(k)} \tag{13}$$

where,

$$S_{j-1} = D^{(k)} > CD_g^{(k)}, D^{(k+1)} > CD^{(k+1)} \cdots D^{(j-1)} > CD^{(j-1)}$$

5.2 Path selection and path criticality computation

As mentioned, the global criticality of the path is used as the metric for path selection. For each currently traversed sub-path, the path criticality metric (PCM) is computed as follows. Let Π denote the set of paths having the sub-path in common. Assume that D_Π is the delay of this path set and CD_Π, its complementary delay within the parent group. Using equation (7), it can be written as

$$PCM_\Pi = P(D_\Pi > CD_\Pi | D^{(N-1)} > CD_g^{(N-1)}) \times Q_g^{(N-1)} \tag{14}$$

Here, N stands for the number of levels of the hierarchy. Note that PCM_Π is the global criticality of the path set Π. In equation (14), $Q_g^{(N-1)}$, $D^{(N-1)}$ and $CD_g^{(N-1)}$ are known and are the same for all paths within the group. Therefore, the evaluation of the metric requires the evaluation of CD_Π, which is the local complementary delay within the parent group. This is done as follows.

Assume that the path selection is done in group G_k at level $(N-1)$. As explained previously, this effectively means that nodes v_k to v_t will be common for all the paths

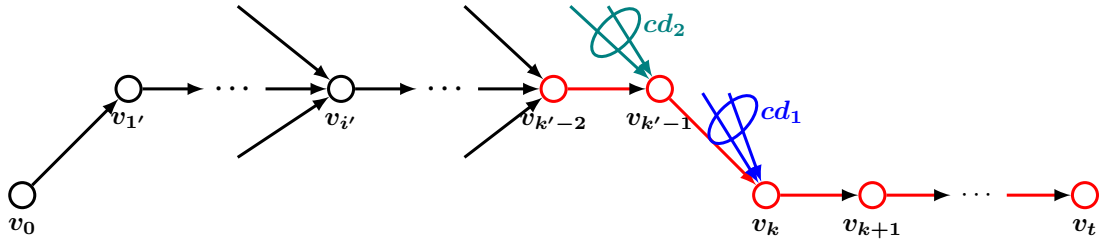


Fig. 4. An example to illustrate evaluation of the local complementary delay of a subpath within the parent group G_k

in group G_k. We start a backward traversal from node v_k (excluding v_{k-1} which is not a member of group G_k) and extend the sub-path. Corresponding to each sub-path, the following example illustrates how CD_Π is computed. Consider the set of paths (Π) passing through the sub-path $< v_{k'-2}, v_{k'-1}, v_k, v_{k+1}, \cdots, v_t >$ (marked in red) in Figure 4. The delay and local complementary delay of the path set is given by,

$$D_\Pi = AT_{v_{k'-2}} + d_{(v_{k'-2} \to v_t)}$$
$$CD_\Pi = \max(cd_1, cd_2) + d_{(v_k \to v_t)}$$

where,

$$cd_1 = \max_{\substack{p \in pred(v_k) \\ p \neq v_{k'-1} \\ p \ in \ Group \ G_k}} \{AT_p + d(p, v_k)\}$$

$$cd_2 = \max_{\substack{p \in pred(v_{k'-1}) \\ p \neq v_{k'-2}}} \{AT_p + d_{(p \to v_k)}\}$$

Once the delay and local complementary delay is obtained, PCM_Π is computed using Eqn.(14). We use this metric in a branch-and-bound algorithm as follows. PCM_Π is the sum of criticality of all the paths having this sub-path in common. Therefore, if PCM_Π is less than a lower bound, the fan-in cone of that node is pruned. This is possible since we are using global criticality of the paths as the metric. On the other hand, if the metric exceeds the bound, then there is possibility of existence of critical paths and we continue the backward traversal until all possible sub-paths are visited and pruned or source node is reached.

PCM_Π is a good metric as it is easy to evaluate and is directly reflective of the global criticality of the path set. The additional advantages we have in using this algorithm are (a) by the time we reach the source node, we get both the path and its criticality and (b) the branch and bound algorithm using TQM requires an initial set of paths for which we need to go right up to the source node. This is not a requirement in our case.

The hierarchical method is efficient because we prune at multiple levels. If the group criticality is itself low, no path search is required within this group. Within a group, the path search can be terminated at any point without first having to get an initial set of paths with the largest value of the metric. Theoretically, the algorithm is guaranteed to give all paths that have global criticality above the lower bound. Practically, as will be seen in the

results, due to errors in criticality computation, a few paths that have criticality close to this bound are not detected.

6 IMPLEMENTATION

There are two aspects to the implementation namely, (a) partitioning into groups, removal of non-dominant groups using pruning techniques, clustering and computation of group criticalities and (b) path selection and criticality computation. Algorithm 1 has the steps involved in group partitioning. It is explained in more detail in the following subsections.

6.1 Group partitioning

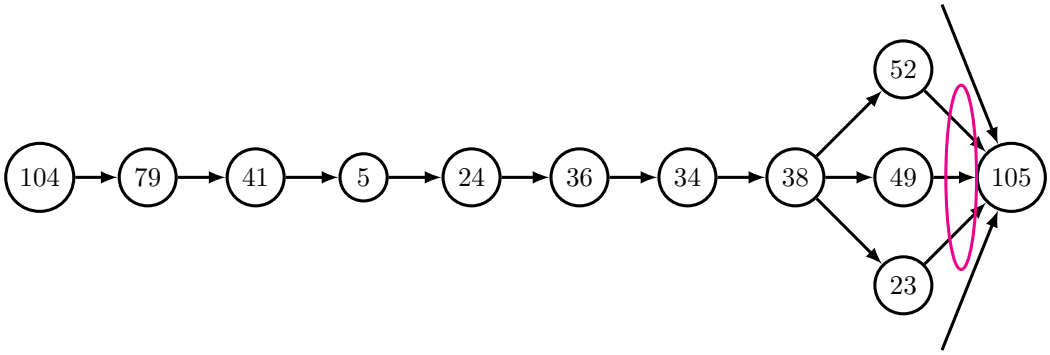


Fig. 5. Level (1) cluster in b03 circuit where the same gate is driving multiple output nets

6.1.1 Clustering of similar groups. Very often circuits have a large number of groups/paths that differ by one or two edges. As a result, their means and standard deviations are virtually identical and they are highly correlated. An example from the benchmark b03 is shown in Figure 5. Each of the three paths belong to a separate group since the output nodes are different. However, their means and standard deviations are virtually identical and the (topological) correlation between the groups is close to one.

The question in these cases is what is the criticality of the three groups. If we follow [17], if the criticality of a set of n groups with identical canonicals is c, the criticality of the individual groups is given by $\frac{c}{n}$. However, it is not clear that this is correct in all cases. If the groups are uncorrelated, the division by n is logical, since each of these will be critical in one in $\frac{n}{c}$ dies. Moreover, the actual paths in each group will be entirely different. However, if the correlation between the groups is close to one, all groups will either be critical or not critical in one in $\frac{1}{c}$ dies. In this case, it does not make sense to divide the criticality between groups. Rather, it seems more logical to regard the group as a whole as a cluster and choose a representative group for further sub-division. Since the groups have a high topological correlation, the final paths obtained from each group will differ in just one or two edges (depending on the correlation). Therefore, the paths obtained from the representative group are essentially reflective of the entire cluster and can be used for optimization/testing.

This idea of clustering and using a representative path has also been used to identify aging related changes in critical paths by [7]. They use singular value decomposition to find representative paths.

In our algorithm, at every level of partitioning, a check is performed to see if there are any similar groups that can be put in a cluster. Two groups are considered to be similar if their means and standard deviations are identical to two decimal places (approximately the error in mean and standard deviation of the arrival times) and the correlation between them is high and exceeds a certain threshold. We have assumed that paths in a cluster can have at most one node that is different and the threshold is fixed based on the independent random component of the variation.

ALGORITHM 1: Recursive partitioning of groups

Input : $G_k^{(i)}$: Group at level i containing paths with nodes $< v_k, .., v_t >$ in common
$L_k^{(i)}$: list of indices indicating hierarchy for group $G_k^{(i)}$ (last index is k)
L_c : list of critical paths

Select a path p in group $G_k^{(i)}$ to get $G_{k,0}^{(i+1)}$
Form groups at level $(i+1)$ within $G_k^{(i)}$
Prune low criticality groups and form clusters of similar groups
$N_g' \leftarrow$ Number of groups after removing non-dominant and similar groups
$Q_0 \leftarrow$ criticality of $G_{k,0}^{(i+1)}$ using Eqn.(7)
if $Q_0 \geq \beta$ **then**
  Add path p to L_c
end if
for j in range(1,N_g') **do**
  $Q_j \leftarrow$ criticality of $G_{k,j}^{(i+1)}$ using Eqn.(7)
  if $Q_j \geq \beta$ **then**
    $L_j^{(i+1)} = [L_k^{(i)},$ j]
    if $Q_j \geq \alpha$ **then**
      Recursive partitioning with inputs $(G_{k,j}^{(i+1)}, L_j^{(i+1)}, L_c)$
    else
      Path selection with inputs $(G_{k,j}^{(i+1)}, L_j^{(i+1)}, L_c)$ [Algo. 2]
    end if
  end if
end for

6.1.2 Recursive partitioning and criticality computation. Partitioning and path selection is controlled by two thresholds α and β. Groups are recursively partitioned as long as their criticality is greater than α. Once the criticality drops below α, we do a path selection within the group based on branch-and-bound to identify all paths that have criticality above the threshold β.

Level 1 groups are based on the output net and the group criticality is computed using the conventional technique after K-center pruning and clustering of similar groups. This is reasonably accurate, since the number of MAX operations is limited by the number of dominant output nets, which is typically not very large. All groups that have a criticality below β are removed. A recursive algorithm is used for subsequent partitioning into disjoint groups as detailed in Algorithm 1. We start with a path p in the current group and partition the paths into N_g disjoint groups where N_g is the length of the chosen path p after removing the common sub-path $< v_k, .., v_t >$. Once again, non-dominant groups are removed using K-center pruning and similar groups are put in a cluster. The global criticality of all the remaining groups (γN_g) is computed using Eqn. (7) and groups for which the criticality is

below β are removed. The group G_0 contains the path p alone and if its criticality exceeds the threshold (β), the path is added to the list of critical paths. For all the other groups with criticality above α, we do a recursive partitioning. Once the group criticality falls below α, the next stage is path selection within the group.

ALGORITHM 2: Path selection using Branch-and-Bound

Input : G_k, L_k, L_c

Sub-path, $\pi \leftarrow < v_k, .., v_t >$
Queue, $Q \leftarrow \{\pi\}$
while $len(Q) > 0$ **do**
  $\pi_{curr} \leftarrow Q.pop()$
  for each $v \in pred(\pi_{curr}[0])$ in G_k **do**
    $\pi \leftarrow \{v, \pi_{curr}\}$
    Compute PCM_π using Eqn.(14)
    if $PCM_\pi \geq \beta$ **then**
      if $v == source$ **then**
        Add π to L_c
      else
        Push π to Q
      end if
    end if
  end for
end while
return

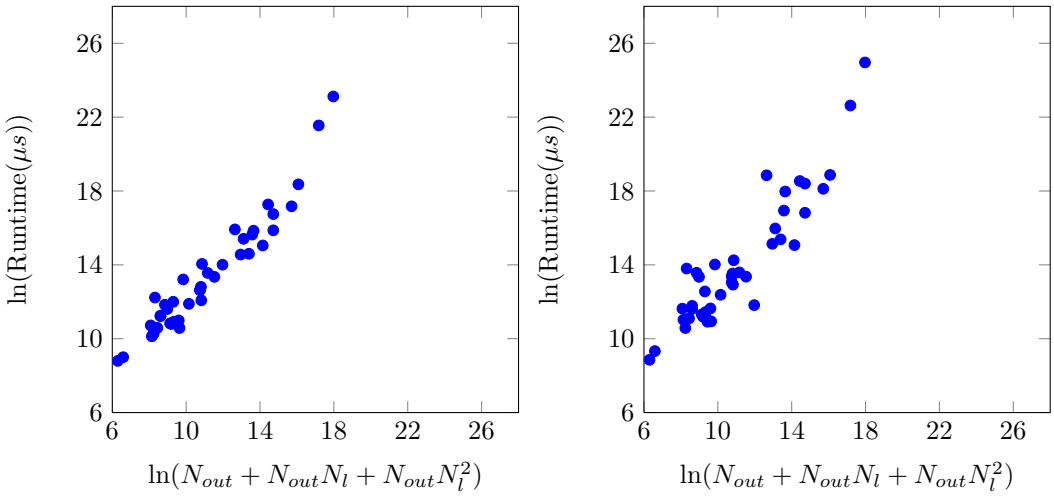
6.2 Path selection

The critical paths contained in the group are identified using branch-and-bound (BnB) technique with PCM as a metric. Algorithm 2 describes the steps. We maintain a queue where we store all the candidate partial paths with metric above the lower bound β. The paths in current group G_k have nodes v_k to v_t in common. Therefore, we start the path selection with the partial path $< v_k, .., v_t >$. At each point, we pop a partial path from the queue and extend it using the predecessor nodes. We compute the metric for the newly found partial path using Eqn.(14). If the metric exceeds β, the new sub-path is pushed back into the queue. Otherwise, the currently traversed path is not useful and the fan-in cone of the node is pruned. Once the source node is reached, we have a complete path and its global criticality. It is added to the list of critical paths if the criticality exceeds β. This process is continued until the queue is empty.

6.3 Complexity

In this analysis, we assume that a block-based SSTA is performed and all the arrival times are known. This is a pre-requisite for all methods for path selection proposed in the literature.

The overall run time of our algorithm depends on the time required for partitioning and path selection within the group. The presence of dominant groups in the circuit results in a larger number of levels in the hierarchy and the time required for partitioning dominates the total run time. On the other hand, if there are many groups with small criticalities, but not one of them is dominant, then the time required for path selection will be more than the time required for partitioning. Moreover, both partitioning and path selection are based



(a) Case 1: Hierarchical(α=0.08 and β=0.05) (b) Case 2 :Hierarchical(α=0.05 and β=0.01)

Fig. 6. Scatter plot of run time as a function of $N_{out} + N_{out}N_l + N_{out}N_l^2$ for various benchmark circuits for two different cases of threshold

on the actual criticalities and the time taken is therefore heavily dependent on the circuit topology. Unlike [5, 20], we only find paths that have global criticality above a threshold. Therefore, the number of paths found is not dependent on the circuit size and it is entirely possible that a large circuit will have only one or two paths above the threshold.

In our analysis we obtain the time complexity of the algorithms in terms of the number of criticality computations and statistical MAX operations. In Algorithm 1, we first use K-center pruning [9] to remove low criticality groups and form clusters of similar groups. It is an $\mathcal{O}(KN_g)$ operation where N_g is the number of groups and K is the number of clusters. The computation of group criticality can be done in linear time [9]. The number of groups at level (1) depends on the number of output nets (N_{out}). Let γ denote the fraction of unpruned groups at any level. In the worst case, only γN_{out} groups need to be further partitioned. The number of groups at levels above one is determined by the number of topological levels (N_l) in the circuit. The total number of groups at level two can be at most $\gamma N_{out}N_l$ and the total number of groups at level three can be at most $\gamma^2 N_{out}N_l^2$ and so on. As seen in Table 2, γ is typically small. As the number of hierarchical levels in most cases is three or four (as seen in Fig. 7), the complexity of the partitioning process is $\mathcal{O}(N_{out} + N_{out}N_l + N_{out}N_l^2)$.

The path criticality metric computation in Algorithm 2 requires $\mathcal{O}(N_l)$ time where N_l is the number of topological levels in the circuit. If there are $N_{out}N_l$ groups that invoke path selection, then the complexity of path selection is of the order of $N_{out}N_l^2$. Thus the overall time complexity of the hierarchical partitioning algorithm is $\mathcal{O}(N_{out} + N_{out}N_l + N_{out}N_l^2)$. Figure 6 shows the run-time time for various benchmarks for two different thresholds as a function of $N_{out} + N_{out}N_l + N_{out}N_l^2$. It is a (natural)log-log plot and the best fit line has slope approximately equal to one (varies between 1 and 1.3 depending on the data set).

Ckt	Level 1			Level i($i > 1$)		
	N_{out}	$N_{g,k}$	γ	N_l	$Max\{N_{g,k}\}$	$Max\{\gamma\}$
b01	6	2	0.33	9	3	0.33
b02	5	4	0.80	7	2	0.29
b03	31	10	0.32	10	3	0.30
b04	67	1	0.01	19	5	0.26
b05	60	4	0.07	28	4	0.14
b06	10	6	0.60	8	2	0.25
b07	50	1	0.02	17	1	0.06
b08	22	3	0.14	14	5	0.36
b09	29	8	0.28	10	4	0.40
b10	18	2	0.11	14	3	0.21
b11	32	1	0.03	21	2	0.10
b12	122	6	0.05	19	4	0.21
b13	54	6	0.11	13	2	0.15
b14	246	2	0.01	75	5	0.07
b15	450	4	0.01	64	14	0.22
b17	1445	3	0.00	81	6	0.07
b18	3294	34	0.01	93	15	0.16
b19	6570	39	0.01	98	16	0.16
b20	513	11	0.02	69	3	0.04
b21	513	2	0.00	69	4	0.06
b22	726	8	0.01	95	8	0.08

N_{out} : Number of output nets, N_l : Number of topological levels

$N_{g,k}$: Number of dominant groups after pruning

γ : Ratio of number of unpruned groups to N_{out}(level 1) or N_l (levels > 1)

Table 2. Fraction of unpruned groups in ITC'99 benchmark circuits with α=0.08 and β=0.05

7 RESULTS

We have implemented the proposed algorithm in C++ and the experiments are performed for ISCAS'85, ISCAS'89 and ITC'99 benchmark circuits. The benchmark circuits are synthesized using 90 nm UMC library. The variations in three process parameters, V_T, L and W are taken into account and each parameter variation is assumed to have a $\frac{\sigma}{\mu}$ of 10%. To model the spatial correlation between parameter variations, three level quad-tree model is used and the parameter variations are equally distributed among the three layers. The random variation is assumed to be 5% of the nominal value. Both cluster MBT [13] and pruning techniques [9, 10] were used to improve the accuracy of the computation. The experiments were performed for two different values of criticality threshold ($\beta = 0.05$ and 0.01) and the path criticality results are verified by comparing against that of Monte Carlo simulation considering 10,000 samples. Experiments were carried out for all benchmarks in the ISCAS'85, ISCAS'89 and ITC'99 sets. Table 3 contains the number of nodes and edges in the directed acyclic graph of each of these benchmark circuits.

ISCAS'85		ISCAS'89		ITC'99			
Ckt	$\mid N \mid (\mid E \mid)$	Ckt	$\mid N \mid (\mid E \mid)$	Ckt	$\mid N \mid (\mid E \mid)$	Ckt	$\mid N \mid (\mid E \mid)$
c17	14 (25)	s953	270 (690)	b01	36 (82)	b11	277 (770)
c432	138 (347)	s1196	302 (818)	b02	19 (42)	b12	732 (1860)
c499	208 (461)	s1238	317 (846)	b03	106 (270)	b13	219 (526)
c880	226 (578)	s1423	388 (889)	b04	329 (863)	b14	2786 (7981)
c1355	223 (494)	s1488	313 (908)	b05	360 (907)	b15	4060 (12032)
c1908	220 (522)	s1494	312 (906)	b06	38 (93)	b17	12959 (38816)
c2670	602 (1374)	s5378	837 (2006)	b07	225 (625)	b18	37926 (109711)
c3540	541 (1480)	s9234	630 (1558)	b08	105 (275)	b19	75258 (219683)
c5315	876 (2316)	s13207	1849 (4438)	b09	108 (248)	b20	5785 (16751)
c6288	1538 (3948)	s15850	2300 (5441)	b10	131 (332)	b21	5967 (17187)
c7552	976 (2416)	s35932	5839 (12786)	-	-	b22	8682 (24952)
-	-	s38417	5517 (14589)	-	-	-	-
-	-	s38584	6799 (17195)	-	-	-	-

Table 3. Number of nodes ($\mid N \mid$) and edges ($\mid E \mid$) in the directed acyclic graph of various ISCAS'85, ISCAS'89 and ITC'99 benchmark circuits

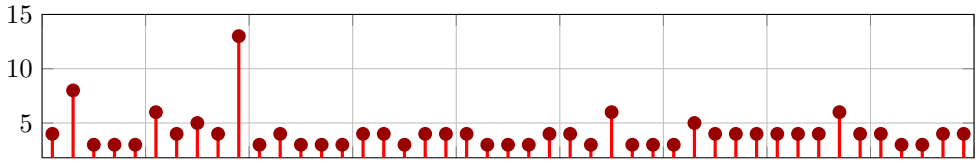


Fig. 7. Maximum number of levels in hierarchical partitioning for ISCAS'85, ISCAS'89 and ITC'99 benchmark circuits for $\alpha = 0.08$

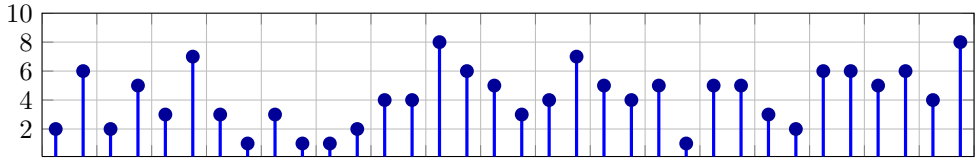


Fig. 8. Maximum number of groups (over all levels) after pruning in hierarchical partitioning for most of the benchmarks circuits in ISCAS'85, ISCAS'89 and ITC'99 sets for $\beta = 0.05$

Figure 7 shows the maximum number of levels in hierarchical partitioning for each benchmark. The thresholds β and α were chosen to be 0.05 and 0.08 respectively. The results indicate that the maximum number of levels is typically 3 or 4 and very rarely exceeds 5. The maximum number of levels depends on the threshold α. When β and α were reduced to 0.01 and 0.05 respectively, the maximum number of levels did not increase by more than 1 or 2 in all the benchmark circuits.

Figure 8 contains the maximum number of groups (over all levels) remaining after pruning for most of the benchmark circuits. As can be seen from the figure, the typical number of groups ranges between two and five, indicating the presence of relatively high criticality

Ckt	$\beta = 0.05$		$\beta = 0.02$		$\beta = 0.01$	
	Max$\{N_{d,g}\}$	Max$\{N_{c,g}\}$	Max$\{N_{d,g}\}$	Max$\{N_{c,g}\}$	Max$\{N_{d,g}\}$	Max$\{N_{c,g}\}$
c499	18	14	18	18	18	18
c1355	18	16	24	21	27	24
c6288	17	1	20	1	20	1
s5378	33	6	37	13	40	16
s35932	80	8	97	40	109	54
s38417	47	40	53	52	53	52
s38584	18	3	29	18	35	33
b15	14	1	17	7	23	16
b18	34	4	48	4	57	6
b19	39	15	39	25	39	25
b20	11	5	17	14	18	6

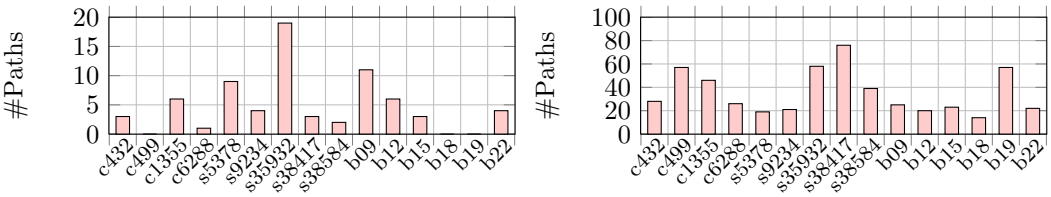
Table 4. Maximum number of dominant groups and critical groups in hierarchical partitioning. $N_{d,g}$: Number of dominant groups after pruning, $N_{c,g}$: Number of groups with criticality greater than β

Ckt	$\beta = 0.05$		$\beta = 0.02$		$\beta = 0.01$	
	#Clusters	#Groups in clusters	#Clusters	#Groups in clusters	#Clusters	#Groups in clusters
c499	2	4	2	4	2	4
c5315	2	4	2	4	3	6
s5378	7	21	8	23	8	23
s35932	40	140	45	157	50	170
s38417	18	50	24	68	24	68
s38584	5	10	11	30	14	43
b03	2	5	2	5	2	5
b06	1	2	1	2	1	2
b09	1	4	1	4	1	4
b18	2	32	2	32	2	32
b19	6	20	6	20	6	20

Table 5. Number of clusters at level (1) in various benchmark circuits

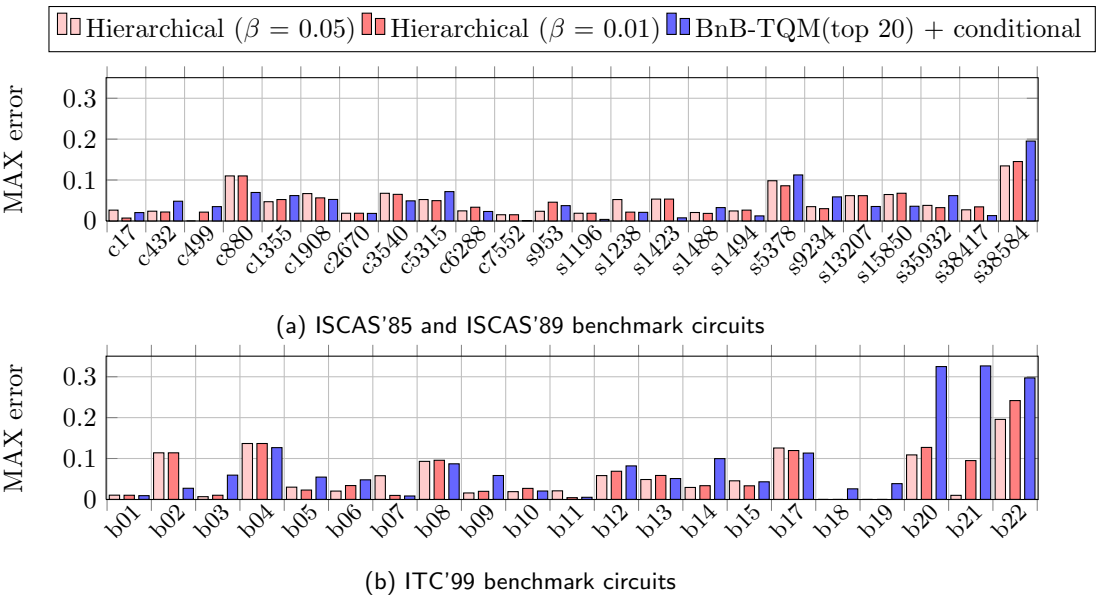
paths. The benchmarks that have a higher number of groups after pruning are listed in Table 4. The table contains the maximum number of groups after pruning as well as groups that have criticality above the threshold β, for three different values of β. As expected, smaller values of β result in a larger number of groups for which recursive partitioning and path selection needs to be done.

Table 5 shows the number of clusters and the total number of groups within clusters for some of the benchmark circuits. In each of these cases, only a representative group from the cluster is selected for further partitioning. The number of clusters increases when the threshold β is lowered in some cases like c5315, s5378, s35932, s38417 and s38584. This is because the total number of groups remaining after pruning increases when the threshold is lowered. Therefore, larger number of similar groups are identified. As seen from the table, in



(a) $\beta = 0.05$ (b) $\beta = 0.01$

Fig. 9. Number of paths reported by the hierarchical method for $\beta = 0.05$ and 0.01



(a) ISCAS'85 and ISCAS'89 benchmark circuits

(b) ITC'99 benchmark circuits

Fig. 10. Comparison of maximum error in path criticality for various approaches : Hierarchical with $\beta = 0.05$, Hierarchical with $\beta = 0.01$ and BnB-TQM (top 20) + conditional probability

some cases, the number of groups in clusters is quite large. The idea of using a representative group for each cluster also contributes to improving the run time.

Figure 9 contains the number of paths reported by the hierarchical method for two different values of criticality threshold. The figure contains the results for those benchmarks that do not have any dominant paths. For circuits with dominant paths, the number of paths increases by 1 or 2 only when the criticality threshold is reduced. But when there are many nearly critical paths, the number of paths reported can be high for smaller values of threshold.

Figure 10 contains a comparison of maximum error in path criticality obtained using (a) BnB-TQM (top 20) for path selection and conditional probability for criticality computation and (b) hierarchical method with criticality thresholds of 0.05 and 0.01. The experimental results indicate that the proposed algorithm is able to give accuracy comparable to the conditional probability approach for most cases and better accuracy for benchmark circuits

S_{MC} : Set of paths with Monte Carlo Criticality greater than β
S_{hier} : Set of paths reported by hierarchical method with Monte Carlo Criticality greater than β
$|S|$: Number of paths in set S
$P(S)$: Sum of Monte Carlo criticalities of the paths in set S

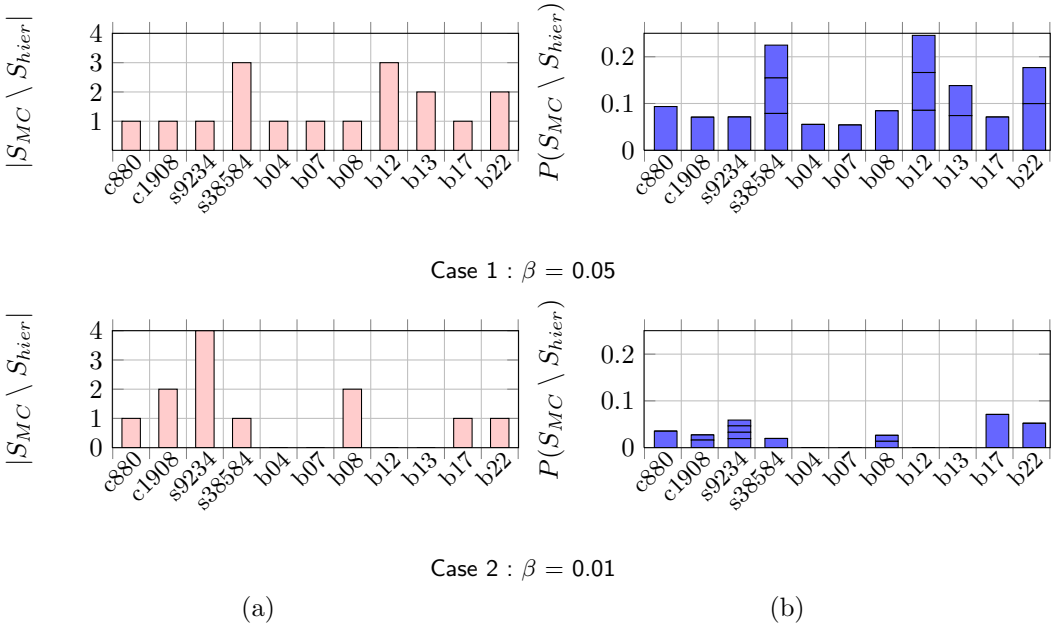


Fig. 11. Comparison of the set of paths reported by hierarchical approach against the set of paths with Monte Carlo criticality greater than β (a) Number of paths with Monte Carlo criticality greater than β, but not reported by hierarchical method and (b) The associated Monte Carlo criticality for the paths in the difference set as a stacked plot

like b20, b21 and b22. This error is zero for $\beta = 0.05$ for some of the benchmark circuits like c499, b18 and b19, since they do not have any paths above the criticality threshold.

We then compared the paths obtained using the hierarchical and Monte Carlo methods. For the Monte Carlo analysis top K paths were enumerated using STA so that the sum of Monte Carlo path criticalities exceeds (1-β) where β is the criticality threshold in hierarchical partitioning. The maximum limit on K was set to be 1000. The aim is to see if the hierarchical method reports all paths above the criticality threshold. We tried using two thresholds; $\beta =$ 0.05 and 0.01. The figure shows that the hierarchical method captures most paths in all the benchmark circuits. Benchmarks that contained paths that the hierarchical method was not able to identify are shown in Figure 11(a). Figure 11(b) shows the criticality of these paths obtained using Monte Carlo analysis. It is a stacked plot containing the criticality of each of the unidentified paths. It indicates that in some of the cases, there are paths with criticality close to 0.1, that are not identified by the hierarchical analysis. However, if the threshold is

S_{hier} : Set of paths reported by hierarchical method
S_{cond} : Set of paths reported by BnB-TQM (top 20) +conditional probability method
$|S|$: Number of paths in set S
$P(S)$: Sum of Monte Carlo criticalities of the paths in set S

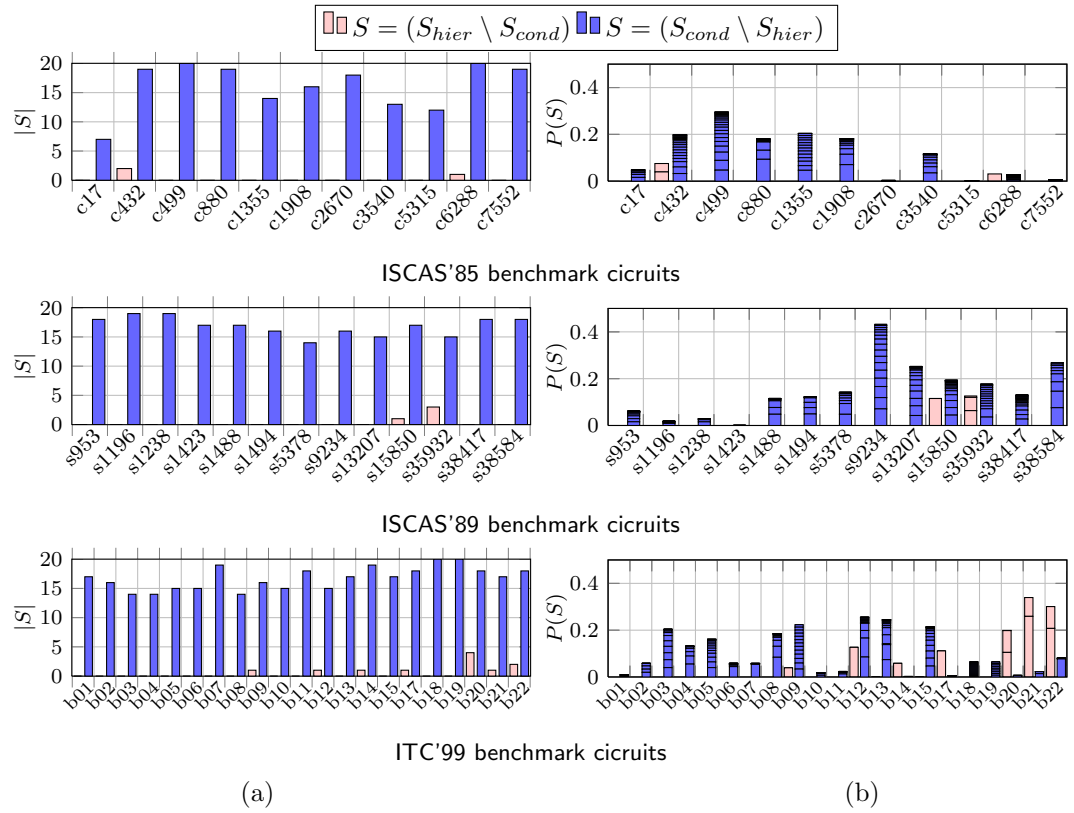
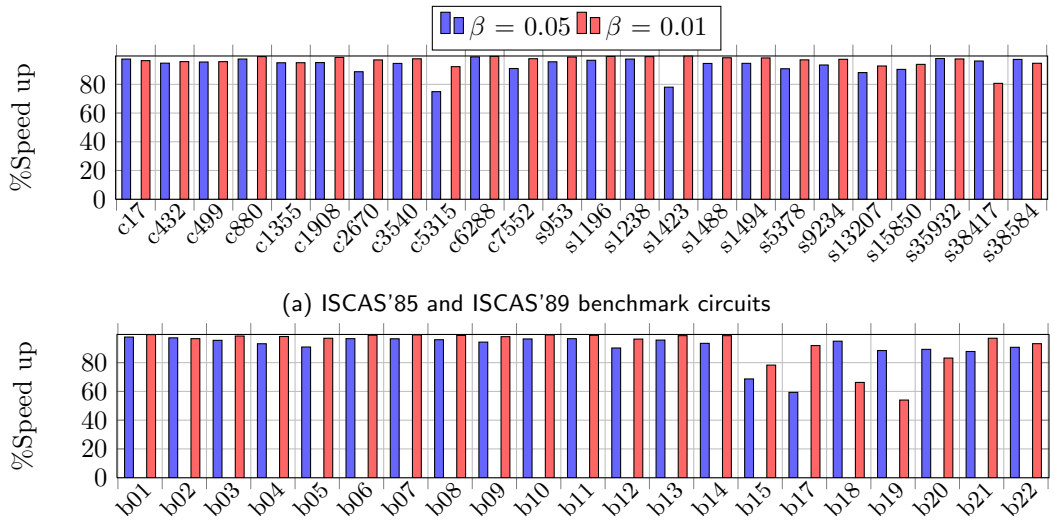


Fig. 12. Comparison of the set of paths reported by hierarchical(β=0.05) and BnB-TQM (top 20) + conditional probability approaches for ISCAS'85, ISCAS'89 and ITC'99 benchmark circuits (a) Difference in the number of paths reported by both approaches and (b) The associated Monte Carlo criticality for the paths in the difference set as a stacked plot

reduced to 0.01, most of these paths are identified. All these paths have a criticality above the threshold, but within the error band. When the threshold is reduced, these paths are identified, but a few other paths close to, but above the lower value of the threshold are not. Overall, we have found that we need to keep the threshold around 0.02-0.04 below the value that we actually need, to make sure we get all the paths.

Figure 12 contains the set of paths and criticality of each of the paths that are (i) reported by the hierarchical method, but not contained in the top 20 paths of the BnB-TQM method and (ii) the paths in the BnB-TQM method that are not reported by the hierarchical method. We see from the figure that most paths not reported by the hierarchical method



(a) ISCAS'85 and ISCAS'89 benchmark circuits

(b) ITC'99 benchmark circuits

Fig. 13. Speed up obtained in using hierarchical method against BnB-TQM + conditional probability method for various benchmark circuits. The results for $\beta = 0.05$ are compared against the top 20 paths using BnB-TQM and the results for $\beta = 0.01$ are compared against BnB-TQM with top 100 paths

have a criticality below the threshold. There are a few paths above the threshold of 0.05, not reported by the hierarchical method (these paths are also contained in the comparison with Monte-Carlo analysis). This is primarily due to the error in criticality computation. As seen from Figure 11, these paths are reported once the threshold is decreased. The BnB-TQM + conditional method fails to report paths with criticality greater than 0.1 in circuits like s15850, b12, b17, b20, b21 and b22. It turns out that these are paths that have higher criticality, but a lower value of the metric (lower than the metrics of top 20 paths). This is expected since the metric used is not directly related to the criticality.

The comparison of run time using hierarchical approach with $\beta = 0.05$ and BnB-TQM (top 20) + conditional probability is shown in Figure 13. For this case, α was fixed at 0.08. The results indicate that hierarchical method is able to give an average speed up of about 90% when compared to conditional method for all the benchmark circuits. The reason for this speed up can be explained as follows. In BnB-TQM method, before we decide whether to continue the traversal or to discard the current sub-path, we need an initial set of complete paths, against which the metric of the sub-paths can be compared. This requires traversal right up to the source node and is time-consuming for large circuits. As mentioned previously, it is not clear how large this initial set needs to be. As a guess, we choose an initial set of twenty (hundred) paths for β equal to 0.05 (0.01), since that is the maximum number of paths possible with that criticality. However, for β equal to 0.05, we get a speed up of nearly 87%, even when the initial path set is limited to 10. In the hierarchical approach, most of the non-critical paths are pruned at the initial levels of hierarchy when the group criticality falls below the threshold. Hence the path selection is done only within a smaller group compared to the entire set of paths. Even for path selection within a group, we do not need any initial set of paths. Whether to continue a traversal or to discard a path is decided

based on the criticality alone and hence most of the non-critical paths will be pruned at the earlier stages itself thereby reducing the run-time.

8 CONCLUSION

In this paper, we propose a hierarchical partitioning algorithm for statistical path selection and criticality computation. Unlike other approaches where path selection is followed by criticality computation, both proceed simultaneously in our approach and finally reports all paths with global criticality above a threshold. The circuit is partitioned into disjoint groups at every level and the criticality of the group, as a whole, is computed. We show that the group criticality can be computed efficiently using the criticality of the parent group and conditional local criticality within the parent group. The low criticality groups are identified and pruned at every level, thereby reducing the search space for critical paths. We also place groups with similar means and standard deviations and high correlation into clusters and consider only a representative group for further partitioning. The recursive partitioning is done until the group criticality falls below a threshold, beyond which the critical paths within the group are identified using a branch-and-bound technique. Since our method for criticality computation is efficient, path criticality is used as the metric in branch-and-bound algorithm and hence once the path is identified, criticality is also obtained.

The experimental results suggest that the number of groups remaining after pruning is typically low compared to the total number of groups and hence partitioning and path selection is done only for a smaller subset of groups. The comparison with Monte Carlo analysis shows that the hierarchical method is able to identify the paths with criticality above the threshold in almost all the cases. The maximum error in path criticality is comparable to the path selection by BnB-TQM and criticality computation using conditional probability approach with around 90% reduction in run time.

REFERENCES

[1] Hongliang Chang and Sachin S. Sapatnekar. 2005. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 9 (2005), 1467–1482. https://doi.org/10.1109/TCAD.2005.850834

[2] Jaeyong Chung and J A Abraham. 2009. Recursive Path Selection for Delay Fault Testing. *VLSI Test Symposium, 2009. VTS '09. 27th IEEE* 1 (2009), 65–70. https://doi.org/10.1109/VTS.2009.50

[3] Jaeyong Chung and Jacob A. Abraham. 2013. Concurrent Path Selection Algorithm in Statistical Timing Analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 9 (Sept. 2013), 1715–1726. https://doi.org/10.1109/TVLSI.2012.2218136

[4] J Chung, J Xiong, V Zolotov, and J A Abraham. 2012. Path criticality computation in parameterized statistical timing analysis using a novel operator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 4 (2012), 497–508. https://doi.org/10.1109/TCAD.2011.2179042

[5] J Chung, J Xiong, V Zolotov, and J A Abraham. 2012. Testability-driven statistical path selection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 8 (2012), 1275–1287. https://doi.org/10.1109/TCAD.2012.2190067

[6] Charles E. Clark. 1961. The Greatest of a Finite Set of Random Variables. *Operations Research* 9, 2 (1961), 145–162. https://doi.org/10.1287/opre.9.2.145

[7] Farshad Firouzi, Fangming Ye, Krishnendu Chakrabarty, and Mehdi B. Tahoori. 2015. Aging- and Variation-Aware Delay Monitoring Using Representative Critical Path Selection. *ACM Trans. Des. Autom. Electron. Syst.* 20, 3, Article 39 (June 2015), 23 pages. https://doi.org/10.1145/2746237

[8] Zijian He, Tao Lv, Huawei Li, and Xiaowei Li. 2013. Test Path Selection for Capturing Delay Failures Under Statistical Timing Model. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 7 (2013), 1210–1219. https://doi.org/10.1109/TVLSI.2012.2208661

[9] Hushrav D. Mogal, Haifeng Qian, Sachin S. Sapatnekar, and Kia Bazargan. 2009. Fast and Accurate Statistical Criticality Computation Under Process Variations. *IEEE Transactions on Computer-Aided*

Design of Integrated Circuits and Systems. 28, 3 (March 2009), 350–363. https://doi.org/10.1109/TCAD.2009.2013278

[10] S. Ramprasath and V. Vasudevan. 2014. Statistical Criticality Computation Using the Circuit Delay. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 33, 5 (May 2014), 717–727. https://doi.org/10.1109/TCAD.2013.2296436

[11] Yiyu Shi, Jinjun Xiong, Vladimir Zolotov, and Chandu Visweswariah. 2013. Order Statistics for Correlated Random Variables and Its Application to At-speed Testing. *ACM Trans. Des. Autom. Electron. Syst.* 18, 3, Article 42 (July 2013), 20 pages. https://doi.org/10.1145/2491477.2491486

[12] Debjit Sinha, Chandu Visweswariah, Natesan Venkateswaran, Jinjun Xiong, and Vladimir Zolotov. 2012. Reversible statistical max/min operation: Concept and applications to timing. In *Proceedings of 49th Design Automation Conference*. IEEE, 1067 –1073. https://doi.org/10.1145/2228360.2228554

[13] Debjit Sinha, Hai Zhou, and Narendra V. Shenoy. 2007. Advances in Computation of the Maximum of a Set of Gaussian Random Variables. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 26, 8 (July 2007), 1522–1533. https://doi.org/10.1109/TCAD.2007.893544

[14] C Visweswariah, K Ravindran, K Kalafala, S G Walker, S Narayan, D K Beece, J Piaget, N Venkateswaran, and J G Hemmett. 2006. First-Order Incremental Block-Based Statistical Timing Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 10 (2006), 2170–2180. https://doi.org/10.1109/TCAD.2005.862751

[15] F Wang, Y Xie, and H Yu. 2007. A novel criticality computation method in statistical timing analysis. *Design, Automation and Test in Europe (DATE-07)* (2007), 1611–1616. https://doi.org/10.1109/DATE.2007.364532

[16] Jinjun Xiong, Yiyu Shi, Vladimir Zolotov, and Chandu Visweswariah. 2009. Statistical multilayer process space coverage for at-speed test. In *Proceedings of the 46th Design Automation Conference, DAC 2009, San Francisco, CA, USA, July 26-31, 2009*. 340–345. https://doi.org/10.1145/1629911.1630004

[17] J Xiong, V Zolotov, N Venkateswaran, and C Visweswariah. 2006. Criticality computation in parameterized statistical timing. *ACM/IEEE 43rd Design Automation Conference (DAC-06)* (2006), 63–68. https://doi.org/10.1109/DAC.2006.229166

[18] Y Zhan, A J Strojwas, M Sharma, and D Newmark. 2005. Statistical critical path analysis considering correlations. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD-05)* (2005), 699–704. https://doi.org/10.1109/ICCAD.2005.1560156

[19] Lizheng Zhang, Weijen Chen, Yuhen Hu, and Charlie Chung-ping Chen. 2005. Statistical Timing Analysis with Extended Pseudo-Canonical Timing Model. *Design, Automation and Test in Europe* (2005), 952–957. https://doi.org/10.1109/DATE.2005.280

[20] V Zolotov, Xiong Jinjun, H Fatemi, and C Visweswariah. 2010. Statistical Path Selection for At-Speed Test. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29, 5 (2010), 749–759. https://doi.org/10.1109/TCAD.2010.2043570

[21] Vladimir Zolotov, Jinjun Xiong, Hanif Fatemi, and Chandu Visweswariah. 2008. Statistical path selection for at-speed test. In *2008 International Conference on Computer-Aided Design, ICCAD 2008, San Jose, CA, USA, November 10-13, 2008*. 624–631. https://doi.org/10.1109/ICCAD.2008.4681642

Received October 2016; revised January 2017; accepted June 2017