Longest Path Selection for Delay Test Under Process Variation*

Xiang Lu†, Zhuo Li†, Wangqi Qiu‡, D. M. H. Walker‡, Weiping Shi†

†Dept. of Electrical Engineering
Texas A&M University
College Station, TX 77843-3124, USA
wshi@ee.tamu.edu

‡Dept. of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
walker@cs.tamu.edu

Abstract - Under manufacturing process variation, a path through a fault site is called *longest* for delay test if there exists a process condition under which the path has the maximum delay among all paths through that fault site. There are often multiple longest paths for each fault site in the circuit, due to different process conditions. To detect the smallest delay fault, it is necessary to test all longest paths through the fault site. However, previous methods are either inefficient or their results include too many paths that are not longest.

This paper presents an efficient method to generate the longest path set for delay test under process variation. To capture both structural and systematic process correlation, we use linear delay functions to express path delays under process variation. A novel path-pruning technique is proposed to discard paths that are not longest, resulting in a significantly reduction in the number of paths compared with the previous best method. The new method can be applied to any process variation as long as its impact on delay is linear.

I. Introduction

Delay test of combinational circuits is to ensure that the signal from any primary input to any primary output is propagated in less time than the system clock cycle time. Under the path delay fault model [1], a circuit is considered faulty if the delay of any path exceeds the specification. A delay fault caused by a local defect, such as a resistive open or short, can only be detected by testing a path through it and testing the longest path is most likely to capture a delay increase along the path due to the fault. Therefore, testing the longest path can detect the smallest local delay fault.

When process variation is not considered, the problem of finding the longest and testable path set that covers all delay fault sites has been extensively studied [2][3][4]. In their work, to test a local delay defect, only the path with the maximum delay is generated among all paths through it. However, under process variation, path delay becomes a function of process variables. Among all paths passing through a fault site, there are often multiple paths whose delay can be the maximum under different process conditions [5]. For each fault site s, we call a path P *longest* if there is a process condition under which path P has the maximum delay among all paths through s. On the other hand, if a path can never be the longest under any process condition, we call the path *redundant*.

Since modern delay optimization tools tend to make many paths critical or near critical [6], too many paths need to be tested if we do not perform path-pruning to exclude redundant paths [7]. According to our experiments on the ISCAS85 circuits, for a delay fault site, the path generator [8] may find more than 500 paths whose delay is within 20% of the worst-case path delay. However, most of these paths are redundant and only a few paths are longest. For example in Fig. 1, we show the delay of four longest paths, P_1, P_2, P_3 and P_4, through a fault site s in ISCAS85 circuit c432. There are two process variables y_1 and y_2, representing Metal2 thickness and Metal3 thickness, respectively. In this example, four paths form the upper bound of the path delay for all paths through the fault site over the range of process variation.

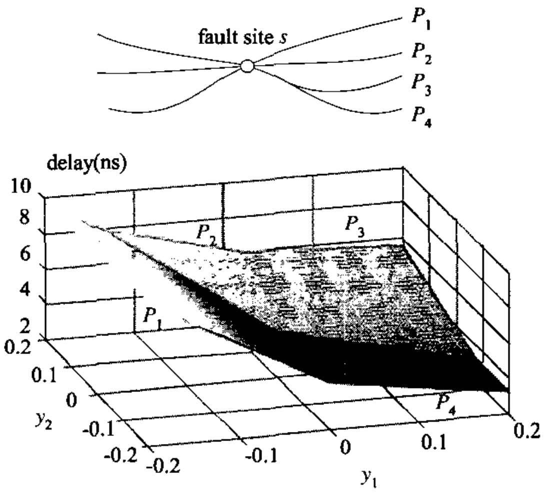

Fig. 1. Delay of four longest paths under process variation.

To detect the smallest local delay fault, all longest paths through s must be tested. Otherwise, if tests are only performed on the longest path under one special process condition, such as the worst-case condition, it might not be guaranteed that the smallest fault be detected under other process conditions, because the longest path under the worst-case condition might not have the maximum delay

*This research was supported in part by the SRC grant 000-TJ-844, NSF grants CCR-0098329, CCR-0113668, EIA-0223785, and ATP grant 512-0266-2001.

under other conditions. In Fig.1, although path P_1 is the longest path for test under the worst-case condition (y_1=-0.2 and y_2=0.2), it does not have the maximum delay under all process conditions. For example, under condition (y_1=0.2 and y_2=-0.2) path P_4 is the longest path, and tests need to be performed on P_4 instead of P_1 to detect the smallest fault under such condition. Therefore, to guarantee the smallest delay fault detection, it is necessary to test all longest paths under any process condition.

The focus of this paper is to generate the set of longest paths for each fault site in the circuit. In order to maximize the fault coverage, we want to find as many longest paths as possible. At the same time, to minimize the number of paths for test, we want to eliminate all redundant paths. Clearly, eliminating redundant paths does not decrease the fault coverage.

Tani *et al.*, tried to solve this problem using the min-max comparison [9]. However, the min-max method is overly pessimistic, even considering the path structural correlation (shared parts between paths). Some researchers considered the problem for global delay faults. Luong and Walker [7] considered the effect of process variation on the gate delay and the spatial correlation between path delays. They selected paths by estimating path delays based on a complex second-order delay model. The method is too time consuming as shown in their paper, and still, some redundant paths exist in the path set due to inaccurate constraints. Liou *et al.* [10] used Monte Carlo simulation to select a set of critical paths that maximizes the probability of covering all critical paths under all process conditions. However, Monte Carlo simulation is very slow for large circuits.

In this paper, we present a new method to find the set of longest paths in a combinational circuit under process variation for delay test. The new method works as follows. We first model the delay of every buffer-to-buffer path segment as a linear function of systematic process variation variables. Then for each fault site s, a path generator [8] is used to obtain a set of critical paths $P(s)$ through s. For each path P_i in $P(s)$, a linear delay function for P_i is derived. Then we construct longest path constraints, and use path-pruning algorithms to remove redundant paths from $P(s)$. The pruning algorithms do not prune longest paths yet prune all redundant paths. Therefore, the remaining paths form the set of longest paths for s. We repeat the process for each fault site to obtain the set of longest paths for all fault sites in the circuit. Experiments on the ISCAS85 circuits show the new method is efficient and significantly reduces the number of paths for test, compared to previous methods.

This paper is organized as follows. The delay model is presented in Section 2. The path pruning algorithms are described in Section 3. The longest path set generation is given in Section 4. Experimental results are shown in Section 5, and the conclusion is drawn in Section 6.

II. Delay Modeling

There are many forms of process variation, see for example Stine *et al.* [11] and Nassif [12]. In this paper, we only consider inter-die systematic process variation, such as variations on gate length, metal width, metal thickness, and inter-layer-dielectric (ILD) thickness. Our method can be extended to include other process variations such as temperature and supply voltage, as long as the approximated delays can be expressed as linear functions of the process variables.

Let the *buffer-to-buffer* delay be the delay from an input pin of a cell to an input pin of a downstream cell. In this paper, we approximate each buffer-to-buffer delay as a linear function of process variation variables:

$$b(x) = b_0+b_1x_1+b_2x_2+ \cdots +b_px_p, \qquad (1)$$

where x=(x_1, x_2, ..., x_p) is a vector of process variables, each representing the deviation from the nominal value, b_0 is the nominal delay, and b_1, b_2, ..., b_p are coefficients.

The validity of the linear model is supported by extensive simulation. We performed multiple parasitic extractions and SPICE simulations under different process conditions. It is found that for any single process variation variable, its effect on delay is approximately linear and symmetric within its variation range. In Fig. 2 we show the SPICE simulation results on a buffer-to-buffer segment in the circuit for several typical process variation variables. Each variable changes within its typical manufacturing range (metal width ±5%, metal thickness ±20%, ILD thickness ±40%, and gate length ±5%). Furthermore the effect of different process variables is independent and additive. Other researchers also noticed the linear property and used it in their models [13][14].

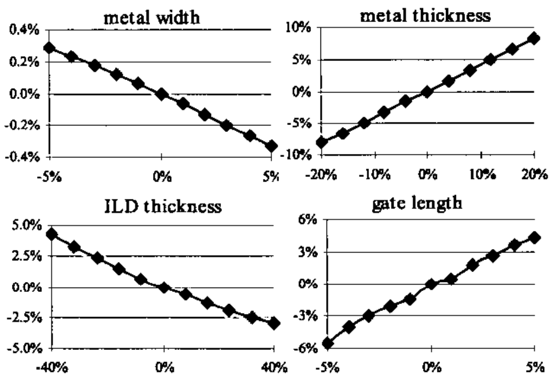

Fig. 2. Delay deviations due to process variation are linear and symmetric in SPICE simulation. The x-axis indicates the range of process variation variables and the y-axis indicates the percentage deviation from the nominal delay.

We generate the coefficients in (1) using a delay sensitivity generation method [15]. The less efficient response surface method (RSM) [16] could also be used to sample the output response to determine the coefficients. The nominal delay b_0 is computed by commercial delay evaluation tools, such as PathMill^{TM}.

III. Path Pruning

Based on the linear delay model, the delay of a path can be derived as a linear function by accumulating all buffer-to-buffer delays along the path:

$$D(x) = d_0 + d_1x_1+ d_2x_2 + \cdots + d_px_p, \qquad (2)$$

where d_0 is the nominal path delay, d_1, d_2, ..., d_p are coefficients for process variation variables.

Let P = {P_1, P_2, ..., P_n} be a set of testable paths through a fault site in the circuit, and let the delay of each path P_i be $D_i(x)$=d_{i0}+ $d_{i1}x_1$+ $\cdots$ +$d_{ip}x_p$, for i=1, ..., n. The range of all process variation variables is defined as $G \subset \Re^p$, where $G=\{(x_1, \ldots, x_p) \mid l_j \le x_j \le h_j, j=1, \ldots, p\}$, and l_j and h_j are the lower bound and upper bound of x_j respectively. For each path P_q in P, if there exists $x' \in G$ such that:

$$D_q(x') \ge D_1(x'), D_2(x'), \ldots, D_n(x'), \quad (3)$$

then from the definition, P_q is a longest path. Formally, we have:

Theorem 1. For each path P_q in P, define an $n \times p$ matrix E_q, where each entry $e_{ij}=d_{qj}-d_{ij}$, for $j=1, \ldots, p$, and an $n \times 1$ vector C with each entry $c_i=d_{i0}-d_{q0}$. Then path P_q is a longest path if and only if there exists a solution in G satisfying the following inequalities:

$$E_q x \ge C. \quad (4)$$

Therefore, verifying whether a path is a longest path or not is equivalent to checking whether the set of inequalities (4) can be satisfied or not. This is known as the feasibility problem of linear programming (LP). When the dimension is fixed, which in our case means p is fixed, LP can be solved in time $O(n)$ [17]. However, the constant factor within the time complexity is exponential with p, resulting in high costs for large p. To further reduce the running time, we now present two heuristics that will be executed before using LP. Heuristic 1 prunes redundant paths. Heuristic 2 identifies longest paths. For those *undetermined* paths that are neither pruned nor identified, LP will be used. Usually after performing Heuristic 1 and Heuristic 2, only a small fraction of paths are undetermined. Therefore the time cost of LP is greatly reduced.

Consider an inequality $\sum e_{ij}x_j \ge c_i$ defined in (4). For any process variable x_k, we can rewrite the inequality as

$$e_{ik}x_k \ge c_i - \textstyle\sum_{j\ne k} e_{ij}x_j. \quad (5)$$

Now define the *loose* range of x_k as:

$$\cap_{1\le i\le n} \{x_k \mid e_{ik}x_k \ge c_i - \max(\textstyle\sum_{j\ne k} e_{ij}x_j)\}, \quad (6)$$

and the *tight* range of x_k as:

$$\cap_{1\le i\le n} \{x_k \mid e_{ik}x_k \ge c_i - \min(\textstyle\sum_{j\ne k} e_{ij}x_j)\}. \quad (7)$$

Therefore, we have:

Theorem 2. Path P_q is redundant if there exists a process variable such that the loose range of this variable is empty.

Proof. We prove it by contradiction. Assume path P_q is a longest path under such condition. According to Theorem 1, there exists $x'=[x_1', \ldots, x_p']$ that satisfies $E_q x' \ge C$. Thus, for a variable x_k' in x', $e_{ik}x_k' \ge c_i - \sum_{j\ne k} e_{ij}x_j' \ge c_i - \max(\sum_{j\ne k} e_{ij}x_j')$. Therefore the loose range of x_k' is not empty. Similarly, the loose range of every variable in x' is not empty. This contradicts the assumption that there exists at least one variable whose loose range is empty. □

Theorem 3. Path P_q is a longest path if there exists a process variable such that the tight range of this variable is not empty.

Proof. We prove it by contradiction. Assume path P_q is redundant under such condition, then for any solution x in G, $E_q x \ge C$ cannot be satisfied. Thus, for a variable x_k, there exists at least one row m, such that $e_{mk}x_k < c_m - \sum_{k\ne j} e_{mj}x_j < c_m - \min(\sum_{k\ne j} e_{mj}x_j)$. Thus the tight range of x_k is empty. Similarly, the tight range of every process variable is empty. This contradicts the condition that there exists at least one variable whose tight range is not empty. □

Pseudo codes of Heuristic 1 in Fig. 3 and Heuristic 2 in Fig. 4 are drawn based on Theorem 2 and Theorem 3 respectively. The time complexity of each heuristic is $O(np^2)$.

```
Heuristic 1: Prune redundant paths
Input: Matrix $E_q$, vector $C$, integer $q$
Output: $P_q$ is "redundant" or "undetermined"
1 For $k=1:p$
2     $s_l=l_k$; $s_h=h_k$
3     For $i=1:n$ $(i\ne q)$
4         For $j=1:p$ $(j\ne k)$
5             If $e_{ij}>0$, then $x_j=h_j$
6             Else $x_j=l_j$
7         If $e_{ik}=0$ and $c_i-\sum_{j\ne k}e_{ij}x_j>0$, then return "redundant"
8         If $e_{ik}>0$, then $s_l=\max(s_l, (c_i-\sum_{j\ne k}e_{ij}x_j)/e_{ik})$
9         If $e_{ik}<0$, then $s_h=\min(s_h, (c_i-\sum_{j\ne k}e_{ij}x_j)/e_{ik})$
10        If $s_l>s_h$, then return "redundant"
11 Return "undetermined"
```

Fig. 3. Pseudo codes of Heuristic 1.

```
Heuristic 2: Identify longest paths
Input: Matrix $E_q$, vector $C$, integer $q$
Output: $P_q$ is "longest" or "undetermined"
1 For $k=1:p$
2     $s_l=l_k$; $s_h=h_k$
3     For $i=1:n$ $(i\ne q)$
4         For $j=1:p$ $(j\ne k)$
5             If $e_{ij}>0$, then $x_j=l_j$
6             Else $x_j=h_j$
7         If $e_{ik}=0$, then continue
8         If $e_{ik}>0$, then $s_l=\max(s_l, (c_i-\sum_{j\ne k}e_{ij}x_j)/e_{ik})$
9         If $e_{ik}<0$, then $s_h=\min(s_h, (c_i-\sum_{j\ne k}e_{ij}x_j)/e_{ik})$
10        If $s_l\le s_h$, then return "longest"
11 Return "undetermined"
```

Fig. 4. Pseudo codes of Heuristic 2.

IV. Longest Path Generation

Given a set of testable paths, we generate the set of longest paths by pruning redundant paths from it. Testable paths are generated by the algorithm in [8] and are ranked in the order of non-increasing nominal delays. The path with the largest nominal delay has index 0, and the path with the second largest nominal delay has index 1, etc. For each fault site, we first request a batch of K longest paths, indexed from 0 to $K-1$. Then the path-pruning algorithms are applied to prune all redundant paths. Finally, the probability that a path in the next batch could be longest is estimated. If the probability is less than a specified criteria value, for example 0.1%, the procedure stops. Otherwise, we request the next batch of paths from the path generator, indexed from K to $2K-1$. The above procedure is repeated until the stop criterion is satisfied. The flowchart of longest path generation is shown in Fig. 5.

To estimate the probability that longest paths could exist in the next batch of paths, we consider the distribution of the already generated longest paths versus path indexes for all fault sites. Let $f(z)$ be the percentage of fault sites where the path with index z is a longest path. Because of the non-increasing order of the nominal path delay and path delay correlations, paths with greater indexes are less likely to be longest paths. Thus, $f(z)$ decreases with the increasing of index z, and can be modeled by a rational function:

$$f(z)=\frac{1}{1+az+bz^{2}}, \tag{8}$$

where parameters a and b can be computed by performing curve fitting on the distribution of already generated longest paths. Although $f(z)$ changes when a new batch is considered, experiments show that $f(z)$ varies little when a proper K is used. To minimize the cost of path pruning, K cannot be too large. On the other hand, K cannot be too small; otherwise, the rational function is inaccurate. In our experiments we use K=50 and the number of longest paths that escape our stop criteria is very small.

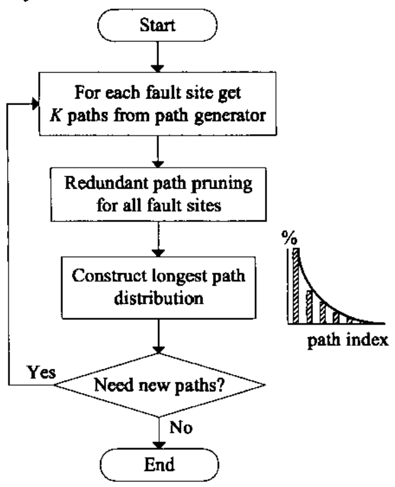
Start
For each fault site get K paths from path generator
Redundant path pruning for all fault sites
Construct longest path distribution
%
path index
Need new paths?
Yes
No
End

Fig. 5. Flowchart of longest path generation.

Case studies show that the rational function model matches well with the distribution of longest paths in real circuits. Fig. 6 gives the actual longest path distribution versus path indexes and $f(z)$ in an ISCAS85 circuit, where the x-axis indicates the path index and the y-axis indicates values of $f(z)$. In this case, a = 0.158 and b = 0.670. $f(0)$ is 100% because the path with index 0 has the maximum delay under the nominal process variation.

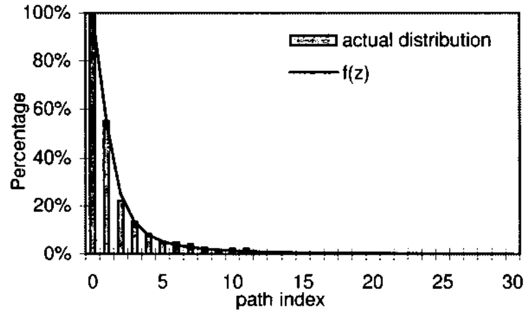

Fig. 6. $f(z)$ and actual distribution of longest paths versus path indexes.

Because a longest path through a fault site is very likely to be a longest path through another fault site in the circuit, a path collapsing procedure is performed to discard the shared paths among all fault sites in the circuit, after the longest path set of each fault site is generated. The procedure can be implemented in linear time in terms of the number of paths. The collapsed path set is the longest path set that covers all delay fault sites in the circuit and must be tested.

V. Experiment Results

We apply our method to ISCAS85 circuits using a 2.4GHz Pentium 4 with 512MB memory. Cadence Silicon Ensemble™ is used for circuit layout generation and parasitic extraction under TSMC 180nm 1.8V 5-metal layer technology. Systematic process variation variables considered in our paper are variations of the transistor gate length, the metal layer width, the metal layer thickness and the inter-layer-dielectrics (ILD) thickness. There are totally 16 variables for the 5-metal layer technology. We apply the following manufacturing ranges of these variables: gate length ±5%, metal width ±5%, metal thickness ±20%, and ILD thickness ±40%. Under such variation ranges, the worst-case variation of a path delay is ±10% in our experimental circuits.

We first compare the performance of our new method with the min-max method, which is the best previous method for the problem [9]. Considering path structural correlation, the min-max method first identifies shared gates between different paths and eliminates the delay of shared gates from path delays. Then min-max comparison is performed on remaining delays. In our new method, path structural correlation is implicitly considered during the formation of delay inequalities. In addition, process correlations are handled by using the same set of process variables in delay functions. Therefore, the new method is able to identify and prune more redundant paths than the min-max method.

In the experiment, we consider a delay fault at the output of each cell in the circuit. A path generator [8] is used to provide critical and testable paths in the batch of 50. The number of longest paths generated by the two methods is shown in Table I.

TABLE I
Performance comparison between the min-max method and the new method

Circuit	# of gates	Min-max method			New method		
		# of longest paths	# of paths for test	Time (s)	# of longest paths	# of paths for test	Time (s)
c432	160	4259	1099	21.8	329	140	0.2
c499	202	14026	2654	32.2	560	223	0.9
c880	383	12017	2505	164.5	516	191	0.2
c1355	546	187444	32231	1872.8	2335	725	3.0
c1908	880	163796	37641	1824.4	1203	400	0.7
c2670	1269	45492	8191	210.1	2309	847	0.8
c3540	1669	153683	29938	1823.9	2189	782	1.3
c5315	2307	129903	24147	1040.9	5072	1882	1.9
c6288	2416	-	-	>3 hr	9552	2507	12.2
c7552	3513	60959	14254	124.7	5604	2051	5.8

In the table, column "# of longest paths" is the sum of the number of longest paths for all fault sites in each circuit, column "# of paths for test" is the number of longest paths after collapsing. We also show the running time of both methods in column "Time (s)", where the running time of the path generator for each method is not included. We do not list the result of the min-max method for circuit c6288 because its time cost is too high. As shown in the table, the number of longest paths selected by the new method is 1%~15% of the results by the min-max method. The ratio is the same for the number of paths after collapsing. In addition, the new method is 20-3000 times faster than the min-max method.

To compare the efficiency between the two methods, in Fig. 7 and Fig. 8, we show path distribution in circuit c5315 using the min-max method and the new method respectively. The x-axis indicates path indexes provided by the path generator, and the y-axis indicates the percentage of fault sites where the path with its x-axis index is a longest path. Clearly, the percentage for the path with index 0 is 100%. As shown in Fig. 7, using the min-max method, the distribution tends to 0 after more than 300 paths are provided for each fault site, while the distribution tends to 0 after only 10 paths in Fig. 8. The significant difference indicates that considering both process correlation and path structural correlation can detect much more redundant paths than the method considering path structural correlation only.

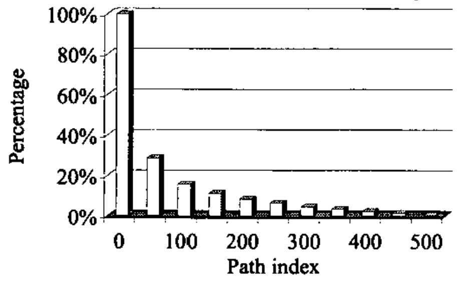

Fig. 7. Longest path distribution versus path indexes in c5315 using the min-max method.

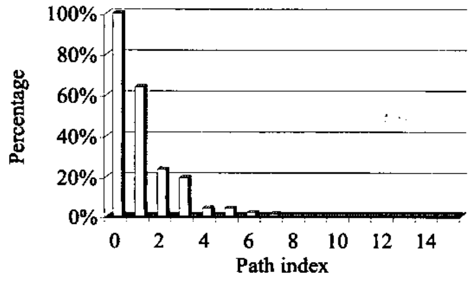

Fig. 8. Longest path distribution versus path indexes in c5315 using the new method.

To compare delay test cost between the two methods, we compare the average longest path set size after path collapsing, and show results in Table II. For each ISCAS85 circuit, the average size for all fault sites are presented. Obviously, the average size using the min-max method is much greater than the new method, which means using the min-max method will result in much more cost in delay test. For the new method, the average size is around 1.0. That means on average only one path is needed to detect the smallest fault for each fault site. Such cost can be comparable to the cost of traditional delay test, where process variation is not considered.

As an example, the distribution of the longest path set size before path collapsing for all fault sites in circuit c5315 is shown in Fig. 9. The distribution indicates that for all paths passing through a fault site, most of them are redundant and only a few paths are longest and necessary to be tested.

TABLE II
Average size of longest path sets for delay test obtained by the min-max method and the new method

Circuit	c432	c499	c880	c1355	c1908
Min-max method	6.9	13.1	6.5	59.0	42.8
New method	0.9	1.1	0.5	1.3	0.5
Circuit	c2670	c3540	c5315	c6288	c7552
Min-max method	6.5	17.9	10.5	-	4.1
New method	0.7	0.5	0.8	1.0	0.6

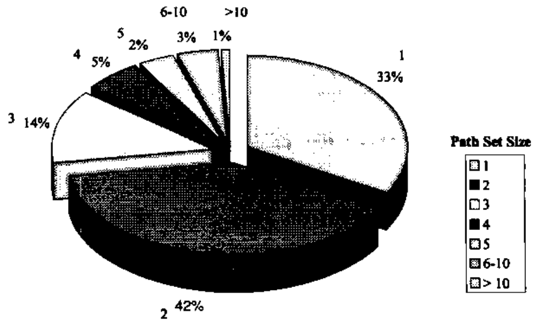

Fig. 9. Longest path set size distribution in c5315.

Finally we compare the performance of our path pruning heuristics and the linear programming (LP) approach in Table III. It is shown that Heuristic 1 and Heuristic 2 can reduce the path set by 95%, and the LP algorithm can reduce the path set by 5% further from the result of Heuristic 1 and Heuristic 2. After performing Heuristic 1 and Heuristic 2, some paths are redundant and pruned, some are longest and kept, and some are undetermined. In Table III we list numbers of these paths and their percentages in the paths provided by the path generator. Undetermined paths must be resolved by LP. Since only a small fraction (less than 2%) of paths are undetermined, the execution time of the expensive LP is greatly reduced. Nevertheless, the running time of LP can be considerable if more than 1% of paths are undetermined after performing Heuristic 1 and Heuristic 2.

VI. Conclusions

In this paper we present a novel and efficient method to find the set of longest paths for delay fault test under process variation. For the first time, we consider both path structural correlation and process correlation, and process variation in both devices and interconnect. Previous researchers considered the process variation only in devices [7] or ignored the process correlation [9]. Two heuristics are proposed to prune redundant paths. Experimental results show the heuristics are very efficient and effective. Our method can significantly reduce the number of paths and test patterns for delay fault test, compared with previous best method. Experiments on ISCAS85 circuits show that the new method reduces the number of paths for test to 1%-15% of the results using the min-max method [9], without decreasing the fault coverage in delay test. In addition, the new method runs 20-3000 times faster than the min-max method, mainly because the min-max method examines far more paths.

We currently only consider systematic process variation and assume there is no intra-die process variation in the paper. However, our method can be extended to consider intra-die variations if the delay effect of intra-die process variation is modeled as a linear function, for example, the principal component analysis based method proposed by Chang *et al.* [18], and the multi-level partition algorithm proposed by Agarwal *et al.* [19]. It will be interesting to see the result of path reduction when intra-die variations are considered.

References

[1] G. L. Smith, "Model the delay faults based upon path," *ITC* 1985, pp. 342-349.

[2] W. N. Li, S. M. Reddy and S. K. Sahni, "On path selection in combinational logic circuits," *IEEE Trans. CAD*, 8(1), Jan. 1989, pp. 56-63.

[3] Y. Shao, S. M. Reddy, I. Pomeranz and S. Kajihara, "On selecting testable paths in scan designs," *IEEE European Test Workshop*, 2002, pp. 53-58.

[4] M. Sharma and J. H. Patel, "Finding a small set of longest testable paths that cover every gate," *ITC* 2002, pp. 974-982.

[5] M. Sivaraman and A. J. Strojwas, "Path delay fault diagnosis and coverage – a metric and an estimation technique," *IEEE Trans. CAD*, 20(3), Mar. 2001 pp. 440-457.

[6] T. W. Williams, B. Underwood and M. R. Mercer, "The interdependence between delay optimization of synthesized networks and testing," *DAC* 1991, pp. 87-92.

[7] G. M. Luong and D. M. H. Walker, "Test generation for global delay faults," *ITC* 1996, pp. 433-442.

[8] W. Qiu and D. M. H. Walker, "An efficient algorithm for finding the K longest testable paths through each gate in a combinational circuit," *ITC* 2003, pp. 592-601.

[9] S. Tani, M. Teramoto, T. Fukazawa and K. Matsuhiro, "Efficient path selection for delay testing based on partial path evaluation," *VTS* 1998, pp. 188-193.

[10] J. J. Liou, A. Krstic, L. C. Wang and K. T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," *DAC* 2002, pp. 566-569.

[11] B. Stine, D. Boning and J. Chung, "Analysis and decomposition of spatial variation in integrated circuit process and devices," *IEEE Trans. Semiconductor Manufacturing*, 10(1), 1997, pp. 24-41.

[12] S. R. Nassif, "Modeling and analysis of manufacturing variations," *CICC* 2001, pp. 223 –228.

[13] A. Gattiker, S. Nassif, R. Dinakar and C. Long, "Timing yield estimation from static timing analysis," *ISQED* 2001, pp. 437-442.

[14] B. Choi and D. M. H. Walker, "Timing analysis of combinational circuits including capacitive coupling and statistical process variation," *VTS* 2000, pp. 49-54.

[15] X. Lu, Z. Li, W. Qiu, D. M. H. Walker and W. Shi, "PARADE: PARAmetric Delay Evaluation under Process Variation," *ISQED* 2004 (to appear).

[16] A. R. Alvarez, B. L. Abdi, D. L. Young, H. D. Weed, J. Teplik and E. R. Herald, "Applications of statistical design and response surface methods to computed aided VLSI device design," *IEEE Trans. CAD*, 7(2), Feb. 1988, pp. 272-287.

[17] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *J. ACM*, 31(1), Jan. 1984, pp. 114-127.

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

[19] A. Agarwal, D. Blaauw and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," *ICCAD* 2003, pp. 271 –276.

TABLE III
Performance of path pruning heuristics and linear programming

Circuit	Heuristic 1 and Heuristic 2								Linear programming			
	# of paths from generator	Paths pruned		Paths longest		Paths undetermined		Time (s)	# of paths from H1&H2	Paths pruned		Time (s)
		#	%	#	%	#	%			#	%	
c432	8000	7671	95.89	310	3.88	19	0.24	0.06	329	0	0.00	0.11
c499	10100	9516	94.22	452	4.48	132	1.31	0.11	584	24	4.11	0.82
c880	19150	18634	97.31	510	2.66	6	0.03	0.17	516	0	0.00	0.04
c1355	27300	24946	91.38	1923	7.04	431	1.58	0.38	2354	19	0.81	2.66
c1908	44000	42795	97.26	1176	2.67	29	0.07	0.50	1205	2	0.17	0.18
c2670	63450	61141	96.36	2285	3.60	24	0.04	0.67	2309	0	0.00	0.16
c3540	83450	81258	97.37	2151	2.58	41	0.05	1.07	2192	3	0.14	0.22
c5315	115350	110274	95.60	4948	4.29	128	0.11	1.17	5076	4	0.08	0.76
c6288	120800	110690	91.63	8820	7.30	1290	1.07	5.67	10110	558	5.52	6.51
c7552	175650	170046	96.81	5217	2.97	387	0.22	3.72	5604	0	0.00	2.05