High-Quality Pattern Selection for Screening Small-Delay Defects Considering Process Variations and Crosstalk

Ke Peng[1], Mahmut Yilmaz[2], Mohammad Tehranipoor[1], and Krishnendu Chakrabarty[3]
[1]ECE Department, University of Connecticut, {kpeng,tehrani}@engr.uconn.edu
[2]AMD, Sunnyvale, CA, mahmut.yilmaz@amd.com
[3]ECE Department, Duke University, krish@ee.duke.edu

Abstract— **Testing for small-delay defects (SDDs) is necessary to ensure the quality and reliability of high-performance integrated circuits fabricated with the latest technologies. These timing defects can be caused by process variations, crosstalk, and power-supply noise, as well as by physical defects such as resistive opens and shorts. Timing-aware ATPG tools have been developed for SDD detection. However, they only use static timing analysis reports for path-length calculation and neglect important parameters such as process variations, crosstalk, and power-supply noise, which can induce small delays into the circuit and impact the timing of targeted paths. In this paper, we present an efficient pattern evaluation and selection procedure for screening SDDs that are caused by physical defects and by delays added to paths by process variations and crosstalk. In this procedure, the best patterns for SDDs are selected from a large repository test set. Experimental results demonstrate that our method sensitizes more long paths and detects more SDDs with a much smaller pattern count compared with a commercial timing-aware ATPG tool.**

I. Introduction

Detecting timing-related defects has become vital for ensuring product quality in the very deep-submicron regime. Such defects are introduced by imperfect manufacturing processes that lead to resistive opens, resistive shorts, and process variations, as well as some non-physical effects such as crosstalk and power supply noise, which cause chip failures by introducing extra delay to the design. Small delay defect (SDD) is one type of such timing defects. SDDs were not a major concern at higher technology nodes because of their small size relative to the timing margins allowed by the maximum operating frequency of a design. Although the delay introduced by each SDD is small, the overall impact can be significant if the sensitized path is a long/critical path, especially when the technology scales to $65nm$ and below [1] [2] [3]. Therefore, SDDs require serious consideration as we strive to increase test quality and decrease the number of test escapes (i.e., increase in-field reliability), denoted by defective parts per million.

The transition delay fault (TDF) model is widely used in industry for targeting timing defects and has been shown to be effective in improving overall test quality. However, traditional TDF ATPG tools are not useful for detecting SDDs since they tend to detect faults via short paths while a SDD can be effectively detected via long paths. Methods such as n-detect ATPG are capable of sensitizing long paths, i.e. detecting SDDs, since it detects faults via different paths [4]. However, significantly large pattern count for large n limits its usage in real applications. Commercial timing-aware ATPG tools, e.g., latest versions of Synopsys TetraMAX [5] and Mentor Graphics FastScan [6], have been developed to deal with the deficiencies of traditional timing-unaware ATPGs in detecting SDDs. However, timing-aware ATPGs do not take into account important underlying causes such as process variations, crosstalk and power-supply noise [7] [8]. Furthermore, these tools result in a significantly larger pattern count and CPU run time compared to 1-detect timing-unaware TDF ATPG [7].

The complexity of today's ICs and shrinking process technologies have made the design features more probabilistic. Thus, it is necessary to perform statistical timing analysis when evaluating the path length considering process variations. Statistical Static Timing Analysis (SSTA) methods were proposed and SSTA tools were developed to deal with these issues [9] [10]. However, these methods are pattern-independent, i.e., they estimate path length using the delay of components on the path without considering the circuit parametric effects. Note that power supply noise and crosstalk are pattern-dependent effects and they can significantly impact the delay of the components on a path.

A. Related Prior Work

Several techniques have been presented in the past few years for screening SDDs and increasing test quality. The "As Late As Possible Transition Fault (ALAPTF)" model was proposed to launch one or more transition faults as late as possible to detect the faults through the least slack path [11]; this approach requires high CPU run time compared to traditional ATPGs. In [12], the authors proposed a delay fault coverage metric to sensitize the longest paths affecting a TDF fault site. It is based on robust path delay test and attempts to find the longest sensitizable paths passing through the target fault site and generating a *slow-to-rise* or *slow-to-fall* transition. The authors in [2] proposed a hybrid method using 1-detect and timing-aware ATPGs to detect SDDs based on SDF with a reduced pattern count. In [3], a static-timing-analysis based method was proposed to generate and select patterns that sensitize long paths. The authors in [13] proposed a procedure to estimate the maximum path delay with coupling noise considering both logic and timing constrains in delay testing. However, not all the patterns can reach this worst-case scenario in real applications.

A false-path-aware statistical timing analysis framework was proposed in [14]. It selects all logically sensitizable long paths using worst-case statistical timing information and obtains the true timing information of the selected paths. In [15], the authors proposed a dynamic pattern compaction

[1]The work of K. Peng and M. Tehranipoor was supported in part by SRC under Contract No. 1587, and by NSF under Grant No. ECCS-0823992.
[2,3]The work of M. Yilmaz and K. Chakrabarty was supported in part by SRC under Contract No. 1588 and by NSF under Grant No. ECCS-0823835.

method for path delay patterns. The $2K$ longest paths, K paths having the rising transition and K paths having the falling transition, through each fault site were reported for path delay pattern generation. However, with the increase in circuit size, it becomes infeasible to generate $2K$ longest paths for each fault. Furthermore, to meet the robust test generation condition, it is infeasible to generate patterns for all these long paths in the design. The authors in [16] proposed a statistical delay fault coverage model based on the propagation delay of a path and the delay defect size. They assumed independent delay distribution for each gate and derived a Gaussian distribution for path delay according to the Central Limit Theorem (CLT). However, the analysis of test effectiveness in this work requires information on the delay distribution of the path under test, the delay defect size, as well as the delay distribution of the longest path passing through the fault site. Obtaining the delay defect size needs considerable analysis using circuit testability or silicon data. Since the path length is a random variable in statistical timing analysis, it is extremely difficult to find *the longest* path passing through the target fault site.

B. Contributions and Paper Organization

In recent work [7], an output-deviation-based method has been presented to detect SDDs. This method defined gate-delay defect probabilities (DDPs) to model delay variations in a design. Based on the Delay Defect Probability Matrix (DDPM) of each gate, output deviations are calculated, which are used for pattern evaluation and selection. However the main drawback here is that in case of a design with a large number of gates on a path, the output deviation metric saturates and equal output deviations (close to 1) are obtained for both long and intermediate paths. A similar method was developed in [8] to take into account the interconnect contribution to the total delay of sensitized paths. In this paper, we present a different and novel approach to address the above problems. Our main contributions include:

1. We use a probability density function (PDF)-based method rather than DDPM-based method for pattern evaluation; this solved the saturation problem associated with the output deviations.
2. We consider crosstalk effect as a source of SDD in nanometer technology designs. The impact of process variations and crosstalk are taken into account dynamically during the pattern evaluation procedure using PDF-based analysis.
3. The procedure we developed can check the overlap between sensitized paths by various patterns so that we can select the most effective patterns with minimum overlap in terms of sensitized paths. The n-detect pattern set is used as our original pattern repository.
4. Our new pattern set increases the number of sensitized long paths, hence the number of detected SDDs.

The remainder of the paper is organized as follows. Section II presents a preliminary analysis of variation sources inducing SDDs. Pattern evaluation and selection procedure is presented in Section III. To evaluate the efficiency of our proposed method, we implement it on several benchmarks and the results are presented in Section IV. Finally, we conclude the paper in Section V.

II. Analyzing Variation-Induced SDDs

As mentioned earlier, SDDs can be introduced by both physical defects and variations. The physical defects include resistive opens and shorts. The variation-induced SDDs in a circuit are from process variations, crosstalk and power supply noise. In this paper, our procedure targets physical SDDs, as well as SDDs introduced by process variations and crosstalk. Power supply noise effects will be taken into account in future work.

A. Impact of Process Variations on Path Delay

In reality, the parameters of fabricated transistors are not exactly the same as design specifications due to process variations. These variations directly result in deviations in transistor parameters, such as threshold voltage, oxide thickness, W/L ratios, as well as variation in the widths of interconnect wires [17], and impact performance (increase or decrease in gate and interconnect delays) to a large extent in the latest technologies. Due to the impact of process variations, the delay of each path segment (gate and interconnect) is assumed to be a random variable X with mean value μ and standard deviation σ, rather than a precise value. Note that μ and σ are finite values. In this work, we assume that delay variations on path segments are independent from each other. The CLT states that the sum of a sufficiently large number of independent random variables, each with finite mean and variance, will approaches a normal distribution [18]. Let the random variable X_p denote the path delay. According to CLT, we have a Gaussian distribution for X_p, where $X_p = \sum_{i=1}^{N} X_{si}$, and X_{si} is the delay of the i-th segment along the path. X_p has mean $\mu_p = \sum_{i=1}^{N} \mu_{si}$ and standard deviation $\sigma_p = \sqrt{\sum_{i=1}^{N} \sigma_{si}^2}$ for large N, where N is the number of segments along the path, and μ_{si} and σ_{si} are the mean delay and standard deviation for segment i, respectively.

We run Monte-Carlo simulations using 180 nm Cadence Generic Standard Cell Library to obtain the delay distributions for all gates in the library. For each gate, Monte-Carlo simulations were run with:

1) Different input-output combinations,
2) Different output load capacitances,
3) Process-variation parameters:
    - Transistor gate length L: $3\sigma = 15\%$,
    - Transistor gate width W: $3\sigma = 15\%$,
    - Threshold voltage V_{th}: $3\sigma = 15\%$,
    - Gate-oxide thickness t_{ox}: $3\sigma = 3\%$.

We obtain the PDF of each standard gate and create a gate-PDF database considering process variations. The interconnect length is extracted from layout. Different variations between metal layers and vias are taken into consideration when calculating the delay distributions. There are 6 available metal layers in the library. For a metal segment with unit delay, we use 3σ variations $30\%, 30\%, 20\%, 15\%, 10\%, 5\%$, and 5% for Metal 1 to Metal 6, respectively. The $3 * \sigma$ variations for vias in Metal layers 1 through 5 are 50% [19].

B. Impact of Crosstalk on Path Delay

The crosstalk effects introduced by parasitic coupling capacitance between a target net (victim) and its neighboring nets (aggressors) may either speed up or slow down the delays on both victim and aggressor nets, according to the transition direction, transition arrival time, as well as coupling capacitance between the victim and aggressor nets [20]. To take crosstalk effects into account during path delay analysis and pattern selection, we perform various analyses to obtain a realistic model of crosstalk. Since transitions on aggressors and victim have different direction and arrival time, we perform a set of SPICE simulations and analyze their impact on each other. Fig. 1 demonstrates the simulation results on crosstalk effects between two neighboring interconnects with a fixed coupling capacitance. The times $t1$, $t2$, and $t3$ in Fig. 1 represent the break-points of the curve-fitting. The parameter t_{a-v} denotes the arrival time difference between transitions on aggressor and victim nets and $d_{arrival}$ represents the victim net delay considering the impact of arrival time difference. It is seen that when the aggressor and victim nets have the same transition direction (see Fig. 1(a)), the victim net will be speeded up. Otherwise, the victim net will be slowed down (see Fig. 1(b)). Furthermore, the crosstalk effect on the victim net is maximized when the transition arrival time of aggressor and victim nets are almost same ($t_{a-v} \approx 0$).

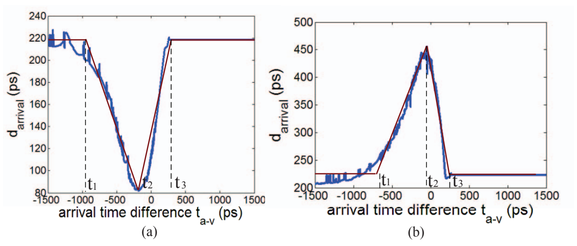

Fig. 1. Impact of aggressor arrival time on victim propagation delay when victim and aggressor nets have (a) same transition direction and (b) opposite transition direction. Coupling capacitance: $0.1pF$.

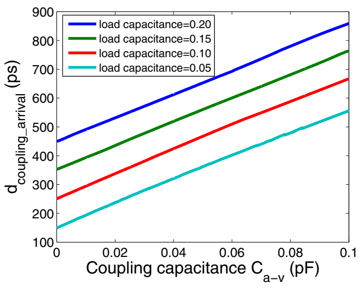

Fig. 2. Impact of coupling capacitance on victim propagation delay with same arrival times, opposite transition direction, and different load capacitances. Load capacitance unit is pF.

We perform another set of simulations and analysis to take into account the impact of coupling capacitance size given a fixed arrival time between transitions on aggressor and victim nets. The simulations were done considering one aggressor for the victim net; superposition is used to consider the total effect when more than one aggressor is used. In Fig. 2 it is shown that the propagation delay on the victim net increases linearly with the coupling capacitance with different load capacitances. $d_{coupling\_arrival}$ denotes the victim net delay considering the impact of coupling capacitance size and C_{a-v} is the coupling capacitance between the aggressor and victim nets. For the same transition direction case, the crosstalk delay decreases linearly.

In the literature on statistical methods, it has been demonstrated that the least squares technique is useful for fitting data to a curve [21]. Instead of solving the equations exactly, the least squares technique tries to minimize the sum of the squares of the residuals. We apply the least squares curve-fitting to the simulation results shown in Fig. 1 and approximate a piecewise function relationship between the interconnect delay and the aggressor-victim alignment as shown in Equation (1).

$$d_{arrival} = \begin{cases} d_{orig}, & 0 \leq t_{a-v} < t_1 \\ a_0 t_{a-v} + a_1, & t_1 \leq t_{a-v} < t_2 \\ b_0 t_{a-v} + b_1, & t_2 \leq t_{a-v} < t_3 \\ d_{orig}, & t_3 \leq t_{a-v} \end{cases} \tag{1}$$

where d_{orig} is the original interconnect delay without considering crosstalk effects. t_{a-v} is the arrival time difference between aggressor and victim nets. a_0 and b_0 are the curve slopes between timing windows $[t_1, t_2]$ and $[t_2, t_3]$ (see Fig. 1), respectively. a_1 is the arrival time delay of the victim net with the conditions $t_{a-v} = 0$ and $t_2 > 0$. Similarly, b_1 is the arrival time delay when $t_{a-v} = 0$ and $t_2 < 0$.

After considering the impact of the transition direction and arrival time, we take the impact of coupling capacitance into account, approximated in Equation (2), which is also obtained from least squares curve fitting.

$$d_{coupling\_arrival} = a' d_{arrival} C_{a-v} \tag{2}$$

where $d_{coupling\_arrival}$ is the propagation delay considering the impact of the transition direction, arrival time and coupling capacitance between the aggressor net and the target victim net. The factor a' is negative for the same transition direction and positive for the opposite transition direction case. Furthermore, these parameters are highly dependent on technology nodes. For different technologies, these parameters are different.

In this work, we assume that for each victim net, crosstalk effects will only impact its mean delay value, rather than its standard deviation. In this way, only the mean delay value of the victim net will be updated when measuring the path delay. For multiple-aggressors case, we consider their impact one by one to update the propagation delay on the victim net and the impact of aggressors on each other is ignored in this work.

III. TDF Pattern Evaluation and Selection

In the proposed procedure, each TDF pattern is evaluated based on the paths it sensitizes. If a TDF pattern sensitizes a large number of long paths in the design, it would be considered an effective pattern. We have developed an in-house tool to list all the sensitized paths for each pattern. During the pattern evaluation and selection procedure, we calculate and evaluate the sensitized paths delay in presence of process variations and crosstalk.

A. Path PDF Analysis

As mentioned earlier, Monte-Carlo simulation is used to obtain the PDFs for each path segment. The gate and interconnect delays are updated after considering process variations and crosstalk effects. With these information, the PDFs of the sensitized paths are then calculated. Each path is evaluated by comparing its length to the clock period. During pattern analysis, we will first obtain the clock period T, and specify a long path threshold (LP_{thr}) based on the clock period to define the *path weight* (W_{path}). The path weight is the probability that the path is longer than the pre-defined long path threshold. Fig. 3 shows an example of path weight definition. Using our method, each sensitized path will have a unique PDF.

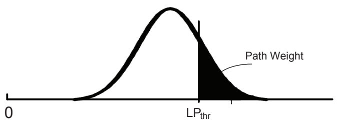

Fig. 3. Path PDF and path weight definition.

After calculating the weight of each sensitized path for $pattern_i$, the weight of $pattern_i$ ($W_{patterni}$) is calculated using Equation $W_{patterni} = \sum_{i=1}^{M_i} W_{pathi}$, where M_i is the total number of sensitized paths by $pattern_i$.

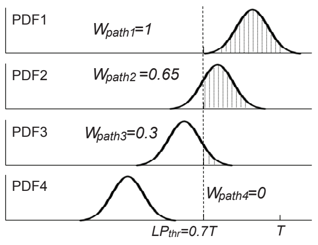

Fig. 4. An example of path and pattern evaluation.

Fig. 4 shows an example of path weight and pattern weight calculation. In this example, assume that $pattern_i$ sensitizes 4 different paths. The PDF of each path is shown in Fig. 4 as $PDF1$, $PDF2$, $PDF3$, and $PDF4$, respectively. Assume that LP_{thr} is $0.7T$. We calculate the weight of these 4 paths, as $W_{path1} = 1$, $W_{path2} = 0.65$, $W_{path3} = 0.3$, and $W_{path4} = 0$. Then the weight of this pattern can be calculated using Equation: $W_{patterni} = 1 + 0.65 + 0.3 + 0 = 1.95$.

B. Pattern Selection

From our analysis and calculation of pattern/path weights in the previous subsection, we conclude that if a pattern has large weight, it is more effective in detecting SDDs. Therefore, we select the patterns with largest weights. However, some of the paths may be sensitized by multiple patterns. In our procedure, if a path has already been detected by the selected patterns, it will not be considered during evaluation of the remaining patterns.

In our pattern selection procedure, the pattern with the largest weight will be the first to be selected. After selecting the pattern, we re-evaluate all the remaining patterns by excluding paths detected by the selected pattern. Then, the pattern with largest weight in the remaining pattern set is selected. This procedure is repeated until some stopping criteria is met, for instance the pattern weight is smaller than a specific threshold. This pattern selection procedure will ensure that the best patterns can be selected from the n-detect pattern set. It can also ensure that there is as little overlap as possible between patterns in terms of sensitized paths. Therefore, it can reduce the pattern count.

```
01: $S \leftarrow$ pattern list of all the patterns, $LP_{thr} \leftarrow$ long path threshold
02: $S = \{P_1, P_2, \cdot, P_N\}$
03: $W(P_i) \leftarrow$ weight of pattern $i$, $P_i$ based on $LP_{thr}$
04: $P_s = NULL$
05:     for ($i = 1, \cdots, N$)
06:     {
07:         for ($j = i, \cdots, N$)
08:         {
09:             update $W(P_j)$ by removing paths detected by $P_s$
10:         }
11:         $P_{max} = MAX\{P_j, \cdots, P_N\}$
12:         if ($W(P_i) < W(P_{max})$)
13:         {
14:             exchange $P_i$ and $P_{max}$ in pattern list $S$
15:         }
16:         $P_s = P_i$
17:     }
18: Return sorted pattern list for pattern selection.
```

Fig. 5. Pattern sorting algorithm.

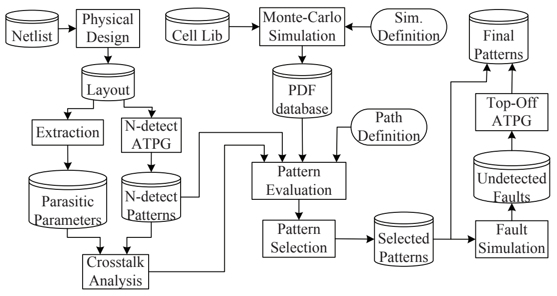

Fig. 6. Flow diagram of pattern generation, evaluation and selection.

The pattern-sorting algorithm is shown in Fig. 5. In this algorithm, each pattern is evaluated by the not already sensitized long paths, i.e., those that are not sensitized by the previously selected patterns. This algorithm will return a pattern list with decreasing order according to the test efficiency of patterns with which we can select the best patterns in the n-detect pattern set.

Assume that N patterns are used by the above algorithm. Also assume that a maximum of M paths are sensitized by a pattern and a maximum of K segments exist on a sensitized path in the target circuit. The worst-case time complexity of the pattern sorting algorithm is $O(N^2MK)$ where $N >> M$ and $N >> K$ for large designs.

IV. Experimental Results

The complete pattern evaluation and selection flow is shown in Fig. 6. We use commercial EDA tools for circuit synthesis, physical design, parasitic parameter extraction, as well as pattern generation. The programs for performing crosstalk calculation, pattern evaluation and selection were implemented using C/C++. We performed our experiments on Linux x86 servers with 8 processors and 24 GB of available memory. Five IWLS benchmarks and two ISCAS benchmarks were used in our experiments. After pattern selection, top-off ATPG is run to meet the fault coverage requirement for TDF fault model. As a result, the final pattern set of our procedure is the selected pattern set plus the top-off ATPG pattern set.

A. Pattern Selection Efficiency Analysis

In this subsection, we will present experimental results for validating the pattern selection procedure. The validation is done by calculating the total number of sensitized unique long paths for the selected patterns. In our experiments, we set the long path threshold $LP_{thr} = 0.7T$ (T is clock period), which is similar to the example in Section III-A. We consider a path to be long if its weight is larger than 0.5, i.e., its mean value is larger than the long path threshold.

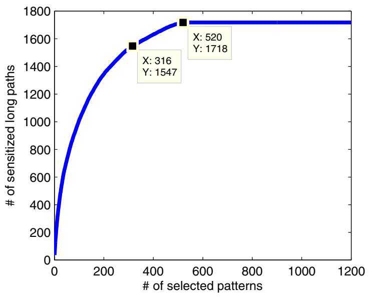

Fig. 7. Long path sensitization by the selected patterns for tv80.

Fig. 7 presents the relation between the number of selected patterns and sensitized unique long paths by the selected pattern set on IWLS benchmark tv80. For the tv80 benchmark, pattern count of n-detect pattern set (n = 10) is 11,072, and the number of sensitized long paths is 1,718, with long path threshold LP_{thr} = 0.7T. From the experimental results in Fig. 7, we can see that only 520 (4.7% of the total pattern count 11,072) patterns are needed to detect all the long paths sensitized by the 10-detect pattern set, or 316 (2.8% of the total pattern count 11,072) patterns are needed to detect 1,547 (or 90.0% of) long paths sensitized by the 10-detect pattern set. From these results, we can see that our pattern grading and selection procedure is very efficient for selecting patterns.

B. Pattern Set Comparison

In our pattern selection procedure, a pattern weight threshold is set as the termination criterion. For example, the pattern weight threshold used in our procedure is $W_{pattern} = 1$, i.e., only the patterns with weight larger than 1 can be selected. Changing this threshold can impact the total number of selected patterns.

Tables I and II show the results for the number of sensitized unique long paths and detected unique SDDs. In general, n-detect and timing-aware pattern sets are expected to perform better in sensitizing unique long paths and detecting unique SDDs compared to 1-detect timing-unaware ATPG. This is indicated by the results shown in Columns 2, 3, and 4 in both tables. Column 5 presents the number of sensitized unique long paths of our selected pattern set. Column 6 presents the number of unique long paths sensitized by top-off ATPG pattern set, not sensitized by the selected pattern set. Top-off patterns are generated using 1-detect (n=1) timing-unaware ATPG. Column 7 presents the total number of sensitized unique long paths for our final pattern set, i.e., the selected patterns plus top-off ATPG patterns. From the results in Table I, we can see that timing-aware ATPG and 10-detect ATPG pattern sets always detect significantly higher number of long paths than 1-detect pattern set except for S13207 benchmark. On the other hand, timing-aware ATPG is not as effective as 10-detect ATPG in long path sensitization for these circuits.

TABLE I
NUMBER OF SENSITIZED LONG PATHS FOR DIFFERENT PATTERN SETS.

Benchmark	1-detect	10-detect	t.aware	selected	top-off	sel+top-off
tv80	976	1,718	1,578	1,623	613	2,236
ac97_ctrl	343	400	360	373	92	465
mem_ctrl	237	2,739	1,168	2,707	186	2,893
systemcaes	772	2,110	1,763	1,940	28	1,968
wb_dma	1,036	1,890	1,209	1,881	98	1,979
s13207	143	155	136	152	25	177
s9234	172	309	203	305	3	308

TABLE II
NUMBER OF DETECTED SDDS FOR DIFFERENT PATTERN SETS.

Benchmark	1-detect	10-detect	t.aware	selected	top-off	sel+top-off
tv80	17,420	32,856	34,199	31,080	10,171	41,251
ac97_ctrl	3,825	4,704	4,183	4,376	1,007	5,383
mem_ctrl	4,614	53,650	22,156	53,024	3,416	56,440
systemcaes	13,366	38,380	32,988	35,334	440	35,774
wb_dma	13,842	26,701	20,131	26,594	1,299	27,893
s13207	2,167	2,629	1,881	2,682	306	2,988
s9234	2,128	3,724	2,442	3,679	44	3,723

TABLE III
COMPARISON BETWEEN THE NUMBER OF PATTERNS.

Benchmark	1-detect	10-detect	t.aware	selected	top-off	sel+top-off
tv80	1,435	11,072	17,107	393	924	1,317
ac97_ctrl	1,032	6,393	4,087	126	834	960
mem_ctrl	1,595	13,142	6,577	352	1,032	1,384
systemcaes	591	3,686	5,590	800	30	830
wb_dma	483	3,600	4,460	131	354	485
s13207	810	6,712	1,108	36	775	811
s9234	343	2,763	428	64	271	335

However, our pattern set is more efficient than 10-detect ATPG pattern set in terms of long paths sensitization, except for S9234 and systemcaes benchmarks.

Table II presents the number of SDDs for different pattern sets. Since SDDs are TDFs on the long paths, if a pattern detects many long paths, then it can also detect many SDDs. Table III presents the number of patterns for 1-detect, 10-detect, timing-aware ATPG, and our pattern set. These patterns are used for obtaining the number of sensitized long paths and detected SDDs as shown in Tables I and II. From these results, we can see that timing-aware ATPG results in large pattern count compared to 1-detect set for large IWLS benchmarks. For some cases, e.g., tv80 and wb_dma benchmarks, its pattern count is even larger than the 10-detect pattern set. For all cases, our pattern set would result in a significantly smaller number of patterns compared to 10-detect and timing-aware pattern sets. In short, our pattern set can detect a large number of long paths with pattern count close to 1-detect pattern set.

C. Long Path Threshold Analysis

Long path threshold LP_{thr} is an important parameter for our procedure. If the long path threshold changes, the path weight calculation threshold will change accordingly. Although it will not impact the effectiveness of our selected patterns, it may impact the number of selected patterns and number of detected long paths and SDDs. If the long path threshold increases, the number of selected patterns decreases and the number of top-off ATPG patterns increases to meet the fault coverage requirement. On the other hand, if we reduce the long path

TABLE IV

LONG PATH THRESHOLD IMPACT ON PATTERN SELECTION FOR TV80.

LP_{thr}	# of sel patterns	# of top-off patterns	Total # patterns	# of LPs	# of SDDs
0.7T	393	924	1,317	2,236	82,502
0.8T	60	1,279	1,339	1,765	63,428

TABLE V

CPU RUNTIME OF DIFFERENT PATTERN SETS FOR TV80.

n-detect	1-detect	3-detect	5-detect	8-detect	10-detect
# patterns	1435	3,589	5,775	8,931	11,072
CPU (PV)	48s	2m17s	4m2s	6m32s	8m3s
CPU (PV+Xtalk)	18m57s	48m28s	1h19m2s	2h1m59s	2h30m3s

threshold, the number of selected patterns increases and top-off ATPG pattern count decreases. The number of sensitized long paths and SDDs will also change accordingly.

The results in Subsection IV-B is only for a fixed long path threshold LP_{thr} (0.7T). Table IV presents the results for two different long path thresholds (0.7T to 0.8T) for tv80 benchmark. When the long path threshold increases, the weight of each pattern decreases and the number of selected pattern decreases as well.

D. CPU Runtime Analysis

Table V presents the CPU runtime of implementing our method on n-detect pattern sets (n=1, 3, 5, 8, and 10) for the tv80 benchmark. It can be seen that as n increases, the pattern count increases. The CPU runtime of the pattern evaluation and selection procedure also increases with the pattern count when considering (i) only process variations (Row 3 in Table V) and (ii) process variations and crosstalk (Row 4 in Table V). Furthermore, CPU runtime increases significantly when considering crosstalk effects. This is because during crosstalk calculation, for each net on the path, the procedure extracts (i) all its neighboring nets with coupling capacitances from the layout database, (ii) the arrival time, and (iii) transition directions on the neighboring nets (if any) for each pattern. This makes the pattern selection procedure much more complex.

Note that the timing-aware TDF ATPG on the tv80 benchmark takes about *1 hour 2 minutes*, and the CPU time of n-detect timing-unaware TDF ATPG (n=1, 3, 5, 8, and 10) on this benchmark is less than *2 minutes*. As seen from the table, our pattern evaluation and selection procedure consumes a considerably lower CPU time when only process variations is considered. The top-off ATPG is quite fast and consumes a negligible CPU time. When taking crosstalk into consideration, the CPU time of evaluating 5-detect pattern set is close to that of timing-aware ATPG. Even though our method is quite fast, comparing it with timing-aware ATPG does not seem fair since our method takes extra features into account during pattern generation. However, we believe that our method's CPU runtime can be further reduced by better programming, optimizing the data structures and algorithms.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel pattern evaluation and selection procedure for screening small delay defects. The proposed procedure takes into account process variations and crosstalk to evaluate their impact on path delay. An n-detect pattern set was used to provide the initial large pattern set. Our procedure was able to efficiently identify high-quality patterns in an n-detect pattern set that sensitize a large number of long paths. The method was evaluated using several ISCAS and IWLS benchmarks and the results demonstrated its effectiveness for reducing the pattern count and significantly increasing the number of sensitized long paths.

As future work, we plan to perform the following simulations and analysis:

1. Continue simulations and analysis using SPICE to validate the accuracy of our developed models.
2. Perform correlation analysis between path PDF and long path enumeration using Monte-Carlo simulations.
3. Improve the sorting algorithm to reduce the complexity. The current CPU runtime of our procedure is comparable to, and in some cases more than timing-aware ATPG.

REFERENCES

[1] R. Mattiuzzo, D. Appello C. Allsup, "Small Delay Defect Testing," *http://www.tmworld.com/article/CA6660051.html* Test & Measurement World, 2009.

[2] S. Goel, N. Devta-Prasanna and R. Turakhia, "Effective and Efficient Test pattern Generation for Small Delay Defects," in *Proc. IEEE VLSI Test Symposium (VTS'09)*, 2009.

[3] N. Ahmed, M. Tehranipoor and V. Jayaram, "Timing-Based Delay Test for Screening Small Delay Defects," in *Proc. IEEE Design Automation Conf.*, pp. 320-325, 2006.

[4] M. E. Amyeen, S. Venkataraman, A. Ojha, S. Lee, "Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor", in *Proc. IEEE International Test Conference (ITC'04)*, pp. 669-678, 2004.

[5] Synopsys Inc., "SOLD Y-2007, Volumes 1, 2, 3," Synopsys Inc.,Oct., 2007.

[6] Mentor Graphics, "Understanding how to run timing-aware ATPG," Application Note,2006.

[7] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-Pattern Grading and Pattern Selection for Small-Delay Defects," in *Proc. IEEE VLSI Test Symposium (VTS'08)*, 2008.

[8] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Interconnect-Aware and Layout-Oriented Test-Pattern Selection for Small-Delay Defects," in *Proc. IEEE Int. Test Conference (ITC'08)*, 2008.

[9] A. B. Kahng, B. Liu, and X. Xu, "Statistical Timing Analysis in the Presence of Signal-Integrity Effects," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 10, October 2007.

[10] R.-B. Lin, and M.-C. Wu, "A New Statistical Approach to Timing Analysis of VLSI Circuits," in *IEEE DAC2001*, 1997.

[11] P. Gupta, and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proc. Int. Test Conf. (ITC'04)*, pp. 1053-1060, 2004.

[12] A. K. Majhi, V. D. Agrawal, J. Jacob, L. M. Patnaik, "Line Coverage of Path Delay Faults," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 5, pp. 610-614, 2000.

[13] R. Tayade, J. A. Abraham, "Critical Path Selection For Delay Test Considering Coupling Noise," in *Proc. IEEE European Test Symposium.*, pp. 119-124, 2008.

[14] J. Lion, 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," in *Proc. Design Automation Conference (DAC'02)*, pp. 566-569, 2002.

[15] Z. Wang, D. M. H. Walker, "Dynamic Compaction for High Quality Delay Test," in *Proc. 26th IEEE VLSI Test Symposium (VTS'08)*, 2008.

[16] E. S. Park, M. R. Mercer, T. W. Williams, "Statistical Delay Fault Coverage and Defect Level for Delay Faults," in *Proc. IEEE International Test Conference (ITC'88)*, 1988.

[17] J. M. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits, A Design Perspective (Second Edition)," Prentice Hall Publishers, 2003.

[18] L. B. Koralov, Y. G. Sinai, "Theory of Probability and Random Processes," Second Edition, springer.com. ISBN: 978-3-540-25484-3.

[19] ITRS 2008, "http://www.itrs.net/Links/2008ITRS/Home2008.htm."

[20] W. Chen, S. Gupta, and M. Breuer, "Analytic Models for Crosstalk Delay and Pulse Analysis Undernon-Ideal Inputs," in *Proc. IEEE International Test Conference (ITC'97)*, pp. 808-818, 1997.

[21] G. W. Recktenwald, "Numerical Methods with MATLAB: Implementations and Applications," Prentice-Hall, 2000.