Critical Fault-Based Pattern Generation for Screening SDDs

Fang Bao[1], Ke Peng[1], Mahmut Yilmaz[2], Krishnendu Chakrabarty[3], LeRoy Winemberg[4], Mohammad Tehranipoor[1]
[1]ECE Department, University of Connecticut
[2]Advanced Micro Devices, Inc., Sunnyvale, California
[3]ECE Department, Duke University
[4]Freescale Semiconductor

***Abstract*— Testing for small-delay defects (SDDs) becomes necessary as technology further scales. Traditional timing-unaware transition-delay fault (TDF) ATPGs are not adequate for detecting SDDs due to sensitization of short paths. Timing-aware ATPGs suffer from multiple paths sensitization limitation and significant test cost. In this paper, we present a critical fault-based methodology to generate high-quality SDD patterns. By focusing on critical faults, high quality original pattern repository could be generated applicably with n-detect ATPG. Novel pattern evaluation and selection method is presented to further minimize pattern count while maintaining the SDD detection ability. Finally, top-off ATPG is performed to ensure meeting the target fault coverage. Experimental results demonstrate that the proposed critical fault-based method improves long path sensitization efficiency by 2.5X and saves approximately 80% CPU runtime compared with total fault-based method. Comparing with timing-aware ATPG, our pattern set detects equivalent or even more SDDs with significantly reduced pattern count.**

I. Introduction

As manufacturing technology scales, the fabricated chips become more vulnerable to timing-related defects. Semiconductor industry increasingly relies on delay fault test for higher defect coverage. Small-delay defect (SDD) only introduces a small amount of extra delay to the design making it very difficult to detect. A SDD detected on a short path may not fail the test but is highly possible to cause a failure when it gets sensitized on a long path in the filed. Therefore, SDDs require a serious consideration for ensuring product quality and in-field reliability in very deep-submicron regime [1].

Traditional TDF ATPG is developed to target gross delay defects thus has a limited ability in meeting the high SDD test coverage requirement in practice. Commercial timing-aware ATPG tools [2] [3] have been developed to address the deficiencies of the traditional TDF ATPGs. With timing information, the timing-aware ATPG can target a fault along the path with minimum timing slack. However, it lacks ability in sensitizing multiple long paths through a fault. In addition, it consumes large storage and runtime limit in practice. n-detect ATPG can be considered as an alternative for SDD detection [4] [5]. The n-detect ATPG tries to generate patterns to detect each fault n times via different paths. However, it is also limited by large pattern count.

To improve the effectiveness of test patterns for screening SDDs, several techniques have been proposed in literature.

1. Path delay fault (PDF)-based methods [6] [7] [8], which concentrate on identifying a set of critical paths in the circuit. Such methods lack the ability in ensuring required fault coverage and require significant path preprocessing time. As a result, for large industry designs, targeting the top 20% or 30% critical paths will be practically impossible.

2. Transition delay fault (TDF)-based methods, which exploit special algorithms to target TDFs via long paths. For instance, in [9], a model named as late as possible transition fault (ALAPTF) is proposed to launch the transition at the fault site as late as possible to detect transition faults through the least-slack paths. The authors in [10] try to generate *K* longest paths per gate to detect the transition fault. These algorithms suffer from time-consuming path search procedure and high complexity problems.

3. Timing-aware ATPG-based methods have been proposed in [11] [12]. A parameter is set either for dividing faults into timing-aware and non-timing aware categories or for selecting faults with different slack margins in order to improve performance of timing-aware ATPG. However, the reported results show that such methods could not make a significant improvement over timing-aware ATPG. In [13], node fanout is taken into account to differentiate the importance of each fault when using timing-aware ATPG. However, the neglect of post-layout issues impaires its accuracy.

4. Pattern grading and selection method is another alternative for screening SDDs. In [14], the authors grade test patterns based on output deviation metric. However, the output deviation metric suffers from saturation problem in large circuits. In [15], the SDF-based pattern grading and selection procedure is proposed for SDD pattern selection. Both methods require large CPU runtime and hardware resources.

In the previous work [15] [16] [17], we proposed a n-detect based hybrid method to generate patterns for SDD detection. Power supply noise, crosstalk, and process variations are considered in path delay calculation to ensure its efficiency in the presence of uncertainties. However, in these previous works, we generate patterns targeting total faults, which makes the pattern generation very time-consuming. Besides, the generated extremely large original pattern repository requires large memory and time for pattern selection. In this paper, we propose an effecient critical fault (CF)-based pattern generation procedure

for screening SDDs which is especially applicable to large industry designs. The main contributions in the work are: (1) A CF-based n-detect technique is developed to generate orignial pattern repository. Timing-based CF identification is developed to reduce ATPG runtime and hardware resources. n-detect ATPG with high n is used to improve multiple long paths sensitization efficiency. (2) A pattern evaluation metric is developed to represent TDF pattern quality based on its number of sensitized long paths. (3) A pattern selection procedure is developed to keep the most effective patterns and further reduce pattern count. (4) A multiple SDD detection evaluation metric is developed to

[1]The work of F. Bao, K. Peng and M. Tehranipoor was supported in part by NSF under Grant no. ECCS-0823992 and CCF-0811632. [2]The work of M. Yilmaz and K. Chakrabarty was supported in part by NSF under Grant no. ECCS-0823835.

1530-1877/11 $26.00 © 2011 IEEE
DOI 10.1109/ETS.2011.26

IEEE computer society

measure pattern effectiveness in screening SDDs.

The remainder of this paper is organized as follows. Section II analyzes the effectiveness of SDD detection of different pattern sets. Our CF-based pattern generation procedure is presented in Section III. The experiment results are shown in Section IV. We conclude the paper in Section V.

II. Overview

Traditional TDF ATPGs lacks SDD detection efficiency, because it tends to detect faults via short paths. Commercial timing-aware ATPG tools [2] [3] could be used to address SDD detection issue. However, the disadvantages of timing-aware ATPGs are: (i) When the longest path going through a fault, that is chosen first, is not sensitizable, timing-aware ATPG begins back-tracking in an attempt to find a test along some other paths. There is no guarantee that the path finally chosen by the tool is a long path except for the first decision the tool makes. (ii) It consumes much more CPU runtime to calcuate and choose the long paths for fault detection, and it usually results in a larger pattern set. Previous experiments in [15] [18] show that, timing-aware ATPGs may consume over 20X CPU runtime compared with 1-detect ATPG and much larger patten count than 10-detect TDF ATPG. n-detect ATPG provides us with an alternative for screening SDDs [16] [15]. With a large n, n-detect pattern set can sensitize more long paths and detect more SDDs than the timing-aware pattern set [15]. But it suffers from a large pattern count in real practice.

Figure 1 shows the number of all sensitized paths and the number of sensitized long paths for n-detect (n = 1, 3, 5, 7, 10, 15, and 20) pattern sets and timing-aware ATPG pattern sets for the IWLS benchmark wb_conmax [19]. Launch-off-capture (LOC) method is used for pattern generation. It can be seen that: (i) The number of sensitized long paths (paths that are longer than 0.7X clock cycle) by n-detect pattern sets shown in Figure 1(b) is much smaller than the number of all sensitized paths shown in Figure 1(a). The proportion is 2.8% for 1-detect. By n-detect pattern sets, the proportion is 3.4%, 3.7%, 3.6%, 3.4%, 3.5%, and 3.7%, respectively for 3, 5, 7, 10, 15 and 20-detect which implies that a large portion of the faults may be detected n times and result in large pattern generation cost but have no contribution to long path sensitization. Then n-detect efforts on such kind of fault may be managed to transform from "n-detect" to "1-detect", subsequently decreasing the pattern count and CPU runtime. (ii) As n increases, the number of sensitized long paths of n-detect pattern set increases linearly as shown in Figure 1(b). Thus n-detect pattern set, when n is large, is able to detect much more SDDs. (iii) In the experiment, the number of sensitized long paths of timing-aware pattern set (12297) shown in Figure 1(b) is limited to half of 20-detect pattern set (24335) which indicates n-detect potentially has higher ability in sensitizing long paths than timing-aware pattern set when n is large.

III. CF-Based Pattern Generation Procedure

Based on the analysis above, we propose a CF-based pattern generation procedure for screening SDDs aimed at acquiring high test effect with low test cost. In our procedure, n-detect ATPG is firstly applied on critical faults to generate a high-quality original SDD pattern repository.

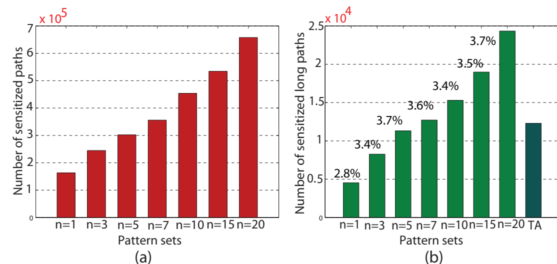

Fig. 1. Number of sensitized paths for different pattern sets in wb_conmax; (a) total sensitized paths by n-detect pattern sets; (b) sensitized long paths by n-detect and timing aware patten sets.

A. Identification of Critical Faults

Using a static timing analysis (STA) tool, we select the timing critical faults before running n-detect ATPG. Post-layout SDF information is used to calculate the slack of each fault. Note that the fault slack reported by STA tool is the minimum slack of a fault, obtained by calculating the longest path running through it. A slack threshold (SL_{thr}) is needed for the timing critical fault selection. All the TDFs with minimum slack equal or smaller than the pre-defined slack threshold will be selected as timing critical faults. In this paper, we consider a fault as timing critical if its minimum slack is equal or smaller than SL_{thr} = $0.3T$, where T is the clock period. It indicates that all the faults on paths that are equal or longer than $0.7T$ will be selected for n-detect pattern generation. In practice, any other slack threshold can be used for timing critical fault selection as well. For example, if we set SL_{thr} very small, only faults on very critical paths will be selected for n-detect ATPG. On the other hand, $SL_{thr} = T$ means all the faults will be selected as timing-critical. It should be noticed that STA tools may not directly provide the exact slack of a fault that is used in our application. In STA, a fault may have a very small slack due to timing constraints but not because it is located on a long path. Such slack value is fine for consideration during design but not precise enough for the identification of critical faults. Therefore, in our procedure, we run one STA iteration to make the slack of a fault precisely correlated to the largest path length running through the fault. The first STA is run to calculate slack inaccuracy introduced by timing constraints for each fault. Then the timing constraints file in the second STA run is modified accordingly to compensate the inaccuracy to make the slack precise. Note that such timing constraints file is only used for critical fault identification but not suitable for circuit design.

Table I presents the efficiency of the critical faults selection in 10-detect on one IWLS circuit ethernet and two industry circuits A and B. TF represents the total number of faults while CF represents the number of critical faults when SL_{thr} is 0.3T. The number in parenthesis shows the ratio between CF and TF. As shown in the table, the number of faults is reduced significantly after critical faults identification. It can be also seen that CF-based procedure decreases CPU runtime and pattern count siginificantly. For industry design, A for example, without CF identification, more than 5 days of 10-detect ATPG time is needed. This makes the TF-based ATPG unacceptable in practice. Our procedure provides an open access to setting slack threshold, thus a slightly loose SL_{thr} could be set to tolerate the delay induced by variable factors such as power supply noise,

process variation, and temperature. Similarly, a slightly tight SL_{thr} could be set for less pattern count and CPU runtime. Note that, if SL_{thr} is changed in this step, the other thresholds used in subsequent procedure should be changed correspondingly.

TABLE I
COMPARISON BETWEEN TF AND CF METHODS IN PATTERN COUNT AND CPU RUNTIME OF 10-DETECT ATPG

Circuit		TF	CF*
ethernet	# of faults	771,840	48,116 (6.2%)
	# of patterns	21,309	9,766 (45.8%)
	CPU(ATPG)	7m24s	2m22s (21.4%)
A	# of faults	3,396,938	377,857 (11.1%)
	# of patterns	>500K	56,784 (<11.36%)
	CPU(ATPG)	>5days	17h30m03s (<14.58%)
B	# of faults	2,828,489	763,379 (27.0%)
	# of patterns	>60K	15,249 (< 25.4%)
	CPU(ATPG)	>2days	5h36m (<11.6%)

* Numbers in parenthesis refer to reduction percentage.

B. Pattern Evaluation and Generation Procedure

By targeting critical faults, our procedure ensures a productive invest on n-detect pattern generation. However, n-detect can not guarantee that each fault is detected via n long paths due to the fact that its main engine is the traditional ATPG. Thus the original n-detect pattern set includes redundant patterns that detect faults via short paths. To remove these inefficient patterns, a pattern evaluation procedure is proposed.

A criteria is needed to evaluate each pattern in the pattern repository according to its SDD detection efficiency so that we can identify the most effective patterns. When a pattern sensitizes a large number of long paths, correspondingly it will detect a large number of SDDs and should be considered as an effective pattern. Otherwise, it would not be considered as an effective pattern. Based on this observation, each pattern is evaluated by its weight calculated using Equation (1).

$$W_{Pi} = \sum_{i=1}^{N} W_{pathi} \qquad (1)$$

where N is the number of sensitized paths and W_{Pi} is the weight of each pattern. A long path threshold LP_{thr} according to the clock cycle is needed for path weight calculation. For all the sensitized paths with length equal or greater than LP_{thr}, W_{pathi} is assigned to 1. Otherwise, W_{pathi} is assigned to 0 to represent a short path. By calculating total weight of sensitized paths, each pattern is tagged with a W_{Pi} and evaluated according to the tag.

Figure 2 shows the flow of sensitized path identification, classification, and evaluation for TDF patterns. For each TDF pattern, we run fault simulation to identify all its detected TDF faults, which are mapped to the topology of the design to identify its sensitized paths. After that, timing data from the SDF file is added to calculate the length of the sensitized paths and classify them to be “short” or “long” with (LP_{thr}). Then pattern weight is calculated for evaluation.

The long path threshold LP_{thr} is complementary to the slack threshold SL_{thr}. In our experiments, faults with slack equal or smaller than $0.3T$ are considered as CFs, or SDDs. Then we set $LP_{thr} = 0.7T$ so that all the faults along the sensitized long paths are SDDs.

After pattern evaluation, we start selection process to remove redundant patterns. In the selection algorithm, the pattern with

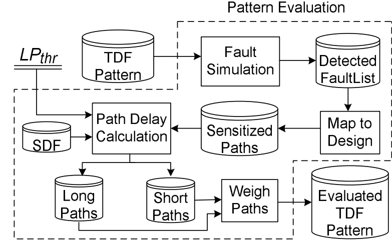

Fig. 2. Patten evaluation procedure for TDF patterns.

the largest weight among all patterns is selected first. Then all the remaining patterns' weights are re-calculated by excluding paths that have already been sensitized. During the pattern weight re-calculation, if a pattern has no unique long path sensitized, it will be removed from the pattern repository to save evaluation runtime. The pattern with the largest weight in the remaining pattern set is selected next. This proposed algorithm returns a selected pattern repository, based on which we can generate top-off pattern set. The complexity of the algorithm is O(N^2M), where N is the number of patterns in pattern set and M is the average number of sensitized paths of a pattern. Actually the runtime almost never hits the theoretic complexity since the original pattern set contains a large amount of patterns that overlap in path sensitization. The pattern count decreases by a large portion after several most effective patterns are selected thus significantly speeds up subsequent selection process.The pattern selection procedure is terminated when the largest weight in the remaining patterns is smaller than the threshold PS_{thr}. The entire flow of the proposed critical fault-based pattern generation is shown in Figure 3. CFs indentification, pattern evaluation and pattern selection are implemented as described earlier and output a effective pattern set for screening SDDs. After that, top-off ATPG is run on the undetected faults to ensure the fault coverage requirement.

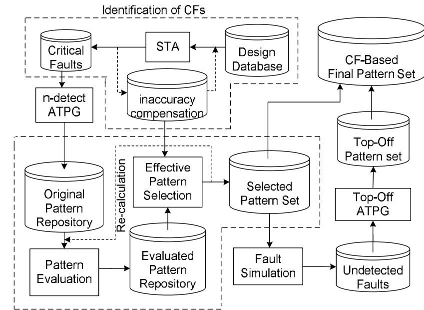

Fig. 3. Critical fault-based pattern generation flow.

C. Multiple SDD Detection Evaluation Technique

In presence of uncertainties such as process variations and on-chip noises, the length of a path can vary to a large extent. As a result, it is desirable to test each critical fault via various long paths, which here is called multiple SDD detection. If a pattern has a high multiple SDD detection ability, it is superior to keep the high test quality in presence of path variability. To check for the multiple SDD detection quality of a pattern, an evaluation metric is proposed and assembled in the final procedure. During long path analysis in pattern evaluation procedure, the number of different long paths running through

critical fault i is extracted as N_{LP_CFi}. Based on N_{LP_CFi}, the average number of sensitized long paths running through detected critical faults and the average number of sensitized long paths running through all critical faults is separately calculated using Equations (2) and (3).

$$N_{LP_DCF} = \sum_i N_{LP_CFi}/N_{DCF} \quad (2)$$

$$N_{LP_TCF} = \sum_i N_{LP_CFi}/N_{TCF} \quad (3)$$

where N_{DCF} and N_{TCF} are the number of detected critical faults and the total number of critical faults. With this evaluation metric, our procedure provides a clear standard to compare different pattern sets' reliability in SDD detection capability.

IV. Experimental Results

In this section, we apply the proposed procedure to nine IWLS benchmarks [19]. The experiments are performed on a Linux x86 server with 8 processors and 24 GB of available memory. Commercial EDA tools are used for logic synthesis, physical design, timing analysis and pattern generation [2].

TABLE II
Benchmarks Characteristics.

Benchmark	# gates	# FFs	# Total cells	# TFs	# CFs	CFs/TFs
tv80	8,353	798	9,151	52,916	15,050	28.4%
wb_dma	8,676	1,423	10,099	56,286	6,119	10.9%
systemcaes	12,570	1,609	14,179	81,018	26,241	32.4%
mem_ctrl	13,408	1,730	15,138	84,986	10,713	12.6%
aes_core	21,515	2,322	23,837	146,402	44,139	30.1%
dma	21,540	2,322	23,862	146,582	27,615	18.8%
ac97_ctrl	24,803	3,059	27,862	150,074	7,519	5.0%
wb_conmax	44,663	4,563	49,226	309,962	53,539	17.3%
ethernet	132,369	11,642	144,011	771,840	48,116	6.2%

Table II shows the charateristics of these benchmarks. Colomn data 2, 3, and 4 are obtained from the post-layout netlists. Columns 5 and 6 show the number of TFs and CFs for each benchmark. All the CFs are selected based on $0.3T$ slack threshold.

A. *Effectiveness in Critical Path Sensitization*

In this subsection, SL_{thr} is set to $0.3T$, i.e., LP_{thr} = $0.7T$. After CF selection, we perform n-detect ATPG ($n = 5, 10, 20$) on both the CF list (CF-n) and the TF list (TF-n), and compare the results in terms of pattern count and long path sensitization. Figure 4 presents the results from two pattern sets generated (1) when 5-detect is run on total faults (TF-5) and (2) when 5-detect is run on critical faults (CF-5). It can be seen that the CF-5 pattern set is much smaller than the TF-5 pattern set (the ratio is shown as P_{CF}/P_{TF} in Figure 4). However, for the long path sensitization (the ratio is shown as LP_{CF}/LP_{TF} in Figure 4), the smaller CF-5 pattern set sensitizes a big portion of long paths as the larger TF-5 pattern set. For wb_dma, the CF-5 pattern set can even sensitize more long paths than the TF-5 pattern set. Except for systemcaes, the long path sensitization ratio between CF-5 pattern set and TF-5 pattern set is much larger than the pattern count ratio. On average, for all benchmarks, 35.9% P_{CF}/P_{TF} can result in 75.5% LP_{CF}/LP_{TF} when n = 5. This predicts that the long path sensitization effectiveness of each pattern in P_{CF} reaches 2.1X of each pattern in P_{TF}.

With an increase in n, CF-n performs better than TF-n in terms of both pattern count ratio and long path sensitization ratio. Figures 5 and 6 present the results from CF-n and TF-n when n = 10 and 20. For most benchmarks, P_{CF}/P_{TF} decreases while LP_{CF}/LP_{TF} increases as n goes up. For systemcaes, the LP_{CF}/LP_{TF} is improved steeply from 55.4% to 94.2%. On average, when n = 10 and 20, 33.2% and 37.8% P_{CF}/P_{TF} brings out 73.1% and 94.0% LP_{CF}/LP_{TF}, respectively. Consequently, each pattern's long path sensitization effectiveness of P_{CF} rises up to 2.2X and 2.5X of P_{TF}. It can be seen that the long path sensitization efficiency of CF-based method significantly increases with n.

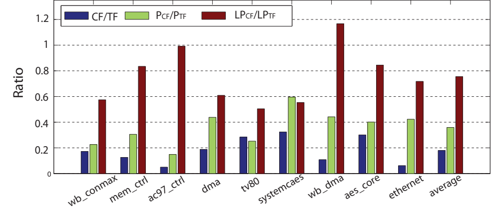

Fig. 4. Comparison between CF-5 and TF-5 pattern sets.

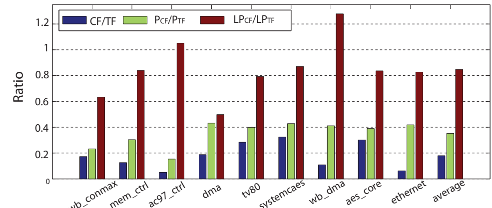

Fig. 5. Comparison between CF-10 and TF-10 pattern sets.

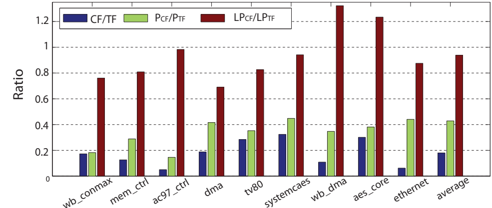

Fig. 6. Comparison between CF-20 and TF-20 pattern sets.

Note that the CF-n method is much faster and requires much less hardware resources than the TF-n method. This leaves us room to increase n so as to increase the number of sensitized long paths. Figure 7 shows an example for comparing the pattern count and number of sensitized long paths obtained from TF-10 and those obtained from CF-n when n = 10, 15, 20 and 30 on ac97_ctrl. It can be seen that even with n = 40, the pattern count generated with CF-40 method is still smaller than that of TF-10 (1218 vs. 1770). However, the number of sensitized long paths increases by approximately 19% (272 to 323).

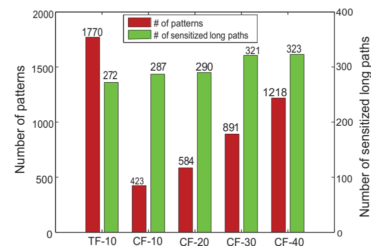

Fig. 7. Comparison between different pattern sets on ac97_ctrl.

TABLE III

CPU RUNTIME COMPARISON BETWEEN TF-BASED AND CF-BASED METHOD

Benchmark	Faultlist	Pat	CPU(ATPG)	CPU(EVA+SEL)	CPU(TOT)
mem_ctrl	TF	11025	41s	1m09s	1m50s
	CF*	3606	13s	8s	21s(81%)
wb_conmax	TF	28354	4m18s	12m05s	16m23s
	CF*	4049	1m07s	2m20s	3m27s(79%)
systemcaes	TF	13039	43s	1m27s	2m10s
	CF*	3507	17s	12s	29s(78%)

* Numbers in parenthesis refer to the saving in CPU runtime.

B. CPU Runtime Analysis

Due to CF identification, each step in our procedure saves CPU runtime. Table III presents CPU runtime of three benchmarks to demonstrate the efficiency of CF-based pattern generation. In this experiment, TF-40 and CF-40 ATPG and pattern selection are performed on the benchmarks. It can be seen that the CF-based method can reduce CPU runtime of n-detect ATPG (shown as *CPU(ATPG)*) by about 60%-75%, and results in a much smaller pattern set, saving huge CPU resources, compared with TF-based method. Note that this CPU runtime includes cost from both n-detect ATPG and top-off ATPG. Consequently, CPU runtime for pattern evaluation and selection (shown as *CPU(EVA+SEL)*) is also reduced by about 60%-80%. As a result, the total CPU runtime (shown as *CPU(TOTAL)*), which is pattern generation runtime plus pattern evaluation and selection runtime, reduces by approximately 80% with our CF-based method. Also, note that our procedure's CPU runtime on CFs is comparable to that of timing-aware ATPG.

C. CF-Based Pattern Generation vs. Timing-Aware ATPG

In this subsection, we compare our final pattern set with two different timing-aware pattern sets generated by (i) timing-aware ATPG on the TF list (shown as TA_T), and (ii) timing-aware ATPG on the CF list (shown as TA_C). In the pattern selection procedure, the pattern selection threshold is $PS_{thr} = 1$, i.e., only the patterns that sensitize at lease one long path can be selected.

Table IV shows the pattern count for five benchmarks. It can be seen that CF-10 pattern sets (shown as "ori.") are comparable to TA_C which are much smaller than TA_T. As n increases from 10 to 40, the CF-n pattern sets become even larger than TA_T. However, after pattern evaluation and selection, the selected pattern sets (shown as "sel.") are much smaller than TA_C. Furthermore, our final pattern sets (shown as "final", which equals to corresponding "sel." plus "topoff") are still much smaller than TA_C. The ascending pattern count sequence is clearly observed as "final", TA_C, and TA_T except for wb_conmax. For wb_conmax, when we selected pattern based on 40-detect pattern set, the selected 1713 pattern set is larger than 1703 pattern set of TA_T. This is because the original pattern repository contains more high quality patterns, considering SDD detection, its efficiency is the best among all pattern sets. This will be seen in Table V and VI.

When comes to the number of sensitized long paths and detected SDDs shown in Tables V and VI, as n increases, our final pattern sets sensitize more long paths and detect more SDDs than TA_T and TA_C for all benchmarks. The efficiency of long path sensitization and SDD detection of our final pattern sets mainly depends on the original CF-n pattern sets. The increasing ratio of SDD detection is not as fast as long paths sensitization. The reason for this is the large number of overlapping SDDs

TABLE IV

NUMBER OF PATTERNS FOR DIFFERENT PATTERN SETS.

Benchmark		CF-10	CF-20	CF-30	CF-40	TA_T	TA_C
ac97_ ctrl	ori.	272	584	891	1218		
	sel.	58	71	67	66		
	topoff	170	181	182	177		
	final	228	253	249	243	593	266
mem_ ctrl	ori.	803	1526	2440	3606		
	sel.	79	117	187	262		
	topoff	412	414	378	339		
	final	491	531	565	611	1,126	813
wb_ conmax	ori.	995	2,062	3,026	4,049		
	sel.	718	1,226	1,510	1,713		
	topoff	302	255	246	237		
	final	1,020	1,481	1,756	1,950	1,703	963
system -caes	ori.	838	1,966	2,625	3,507		
	sel.	308	577	672	809		
	topoff	272	265	263	247		
	final	580	842	935	1,056	1,560	1,002
ethernet	ori.	9,766	20,751	32,740	40,079		
	sel.	1,432	1,472	1,479	1,703		
	topoff	2,360	2,345	2,340	2,295		
	final	3,792	3,817	3,819	3,998	12,065	8,220

TABLE V

NUMBER OF SENSITIZED LONG PATHS FOR DIFFERENT PATTERN SETS.

Benchmark		CF-10	CF-20	CF-30	CF-40	TA_T	TA_C
ac97_ ctrl	sel.	250	290	321	323		
	final	258	295	322	324	262	245
mem_ ctrl	sel.	821	1,232	1,777	2,953		
	final	969	1,410	1,881	3,069	2,228	2,013
wb_ conmax	sel.	7,938	14,800	17,938	19,719		
	final	9,103	15,349	18,304	19,978	12,297	10,296
system -caes	sel.	964	1,761	2,707	3,584		
	final	1,099	1,882	2,804	3,683	3,046	2,884
ethernet	sel.	9,753	10,714	11,035	12,337		
	final	12,821	12,765	13,020	14,316	10,347	10,079

that exist between long paths. For wb_conmax, the number of sensitized long paths of our final pattern set nearly doubles that of TA_C. Correspondingly, the SDD detection ability is 1.31X of TA_C. The table also shows that most of sensitized long paths and detected SDDs of our final pattern sets are contributed by the selected pattern sets (shown as "sel."), which are the subset of original CF-n pattern sets. Take mem_ctrl as an example; when n = 40, 96.2% 2953 out of 3069 sensitized long paths and 98.8% 6510 out of 6590 detected SDDs come from patterns selected from the original pattern set (shown as "sel."). This is because (i) after n-detect ATPG, few long paths are left as unsensitized, and (ii) top-off ATPG, which is timing-unaware 1-detect TDF ATPG, tends to detect faults via the short paths, rather than long paths.

In addition to compare the detected SDD number of different pattern sets which is sort of breadth coverage of SDD, we compare depth coverage of SDD between different pattern sets using proposed multiple SDD detection evaluation technique. Table VII presents a comparison between our CF-n final pattern sets when n = 10, 20, 30, and 40, TA_T, and TA_C pattern sets. It can be seen that N_{LP_TCF} and N_{LP_DCF} of CF-based pattern set increases as n goes up from 10 to 40. CF-20 begins being superior to TA_T and TA_C in detecting critical faults through different long paths. Our CF-based method concentrates more of

TABLE VI

NUMBER OF DETECTED SDDS FOR DIFFERENT PATTERN SETS.

Benchmark		CF-10	CF-20	CF-30	CF-40	TA_T	TA_C
ac97_ ctrl	sel.	2,882	3,003	3,213	3,191		
	final	2,934	3,055	3,222	3,199	2,922	2,857
mem_ ctrl	sel.	4,249	5,384	5,788	6,510		
	final	4,653	5,553	5,942	6,590	7,870	7,665
wb_ conmax	sel.	18,470	22,081	23,440	23,419		
	final	19,293	22,273	23,568	23,526	19,324	17,999
system -caes	sel.	5,337	6,294	7,476	7,983		
	final	5,640	6,552	7,687	8,141	7,465	6,925
ethernet	sel.	35,393	36,916	37,224	39,075		
	final	38,919	40,355	40,556	41,781	37,421	37,331

TABLE VII
NUMBER OF DIFFERENT LONG PATHS SENSITIZED THROUGH THE SAME CF.

Benchmark		CF-10	CF-20	CF-30	CF-40	TA_T	TA_C
ac97_ ctrl	N_{LP_DCF}	2.23	2.50	2.59	2.63	2.27	2.16
	N_{LP_TCF}	0.76	0.89	0.98	0.98	0.78	0.72
wb_ conmax	N_{LP_DCF}	18.25	28.11	32.04	35.23	20.55	17.50
	N_{LP_TCF}	6.06	10.73	12.93	14.11	8.38	6.95
system -caes	N_{LP_DCF}	8.00	9.14	17.07	20.10	6.14	5.14
	N_{LP_TCF}	1.15	1.41	3.77	4.62	0.75	0.52
ethernet	N_{LP_DCF}	16.85	17.80	18.32	21.01	18.43	17.95
	N_{LP_TCF}	12.41	13.40	13.81	16.00	14.14	13.75

n-detect effort on critical faults. Take wb_conmax for example, TA_T and TA_C pattern sets detect each crtical fault 20.55 and 17.50 times through different long paths on average. Our CF-40 pattern set detects each critical fault 35.23 times that is much higher and thus ensures a reliable detection capability to the critical faults via differnet long paths.

D. Trade-off Analysis

The proposed pattern generation flow provides open access to setting variables such as SL_{thr}, LP_{thr}, and PS_{thr}. SL_{thr} and LP_{thr} can be adjusted to treat one fault with different n-detect effort. Changing PS_{thr} can significantly impact the pattern count and SDD detection efficiency.

To meet different requirements such as minimizing the pattern count or maximizing the long path sensitization, we can make different decisions. Figure 8 presents long path sensitization results of the original CF-n pattern sets ($n = 10, 20, 30, 40$) for wb_conmax. The long path senesitization result of TA_T is also shown as a reference. It can be seen that after pattern selection, 10-detect pattern set sensitizes 7938 long paths which is smaller than TA_T. As n increases, the number of sensitized long paths increases considerably. Take CF-20 for instance, to maximize long path sensitization performance, the pattern set at (1226,14800) should be selected since it uses 477 less patterns to sensitize 2,603 more long paths compared with TA_T at (1703, 12297). To minimize pattern count and CPU resources, we can select pattern set at (339,12297) which is only 19.9% of TA_T's 1703 pattern set. Similarly, if we select patterns based on CF-30 or CF-40 pattern set, 223 (13.1% of TA_T) or 190 (11.2% of TA_T) patterns are needed. In fact, all the points, when n=10, between (339,12297) and (1226,14800) could be selected to balance different factors during pattern generation.

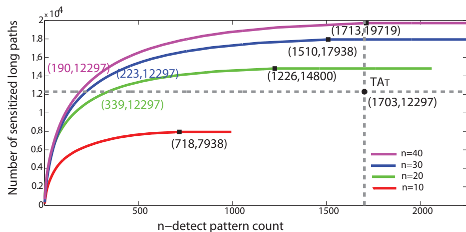

Fig. 8. Tradeoff between pattern count and LP sensitization. The number pair (x,y) shows the pattern count and number of sensitized long paths.

Note that a smaller selected pattern set requires a larger number of topoff patterns to meet the fault coverage requirement. However, the top-off pattern set will not increase significantly since the 1-detect TDF ATPG is capable of detecting faults with minumum runtime and pattern count. Table VIII presents the experimental results on wb_conmax. In this experiment, pattern selection is based on CF-n ($n = 20, 30, 40$) pattern sets, and the pattern selection terminates when the selected patterns sensitize the same long paths as TA_T pattern set. After that, top-off ATPG

TABLE VIII
PATTERN GENERATION RESULTS FOR PATTERN COUNT MINIMIZATION.

n-detect	sel.	topoff	final	# long paths	# TA_T patterns	# TA_T long paths
CF-20	339	343	682	13183		
CF-30	223	369	592	13489		
CF-40	190	406	596	13798	1703	12297

is run. In this case, our procedure needs much less final pattern count (682 for CF-20, 592 for CF-30, and 596 for CF-40) to obtain the same long path sensitization performance as TA_T. Note that, top-off patterns can detect some additional unique long paths fortuitously as well. Comparing Table VIII with Table V for wb_conmax, approximate 2/3 pattern count reduction can be obtained by sacrificing about 6000 sensitized long paths.

V. CONCLUSIONS

This paper proposes a critical fault-based pattern generation method for screening SDDs. With the selected timing-critical faults, the n-detect ATPG can be performed more efficiently to maximize its long path sensitization and SDD detection. An efficient pattern evaluation and selection procedure is proposed to further decrease the pattern count siginificantly while maintaining high SDD detection efficiency. The experimental results show the efficiency of proposed method.

REFERENCES

[1] R. Mattiuzzo, and D. Appello, "Small Delay Defect Testing,"*http://www.tmworld.com/article/CA6660051.html* Test & Measurement World, 2009.
[2] Synopsys Inc., "TetraMAX ATPG User Guide," Synopsys Datasheet, 2010.
[3] Mentor Graphics, "Timing-Aware ATPG_FastScan" Mentor Datasheet, 2008.
[4] M. E. Amyeen, S. Venkataraman, A. Ojha, S. Lee, "Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor", *ITC*, 2004.
[5] Y. Huang, "On N-Detect Pattern Set Optimization," in *Proc. IEEE the 7th International Symposium on Quality Electronic Design (ISQED'06)*, 2006.
[6] J. Lion, A. Krstic, L. Wang, and K.-T. Cheng, "False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation," Design Automation Conference (DAC'02), pp. 566-569, 2002.
[7] L.-C. Wang, J.-J. Liou, and K.-T. Cheng, "Critical Path Selection for Delay Fault Testing based upon A Statistical Timing Model," *IEEE Trans. on Computer Aided Design*, vol. 23, no. 11, pp. 1550-1565, 2004.
[8] R. Tayade and J. A. Abraham, "Critical Path Selection For Delay Test Considering Coupling Noise," in *Proc.IEEE ETS*, 2009.
[9] P. Gupta, and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proc. Int. Test Conf. (ITC'04)*, 2004.
[10] W. Qiu, J. Wang, D. Walker, D. Reddy, L. Xiang, L. Zhou, W. Shi, and H. Balachandran, "K Longest Paths Per Gate (KLPG) Test Generation for Scan Scan-Based Sequential Circuits," in *Proc. IEEE ITC*, pp. 223-231, 2004.
[11] S. Goel, N. Devta-Prasanna and R. Turakhia, "Effective and Efficient Test pattern Generation for Small Delay Defects," *IEEE VTS*, 2009.
[12] P. Cavenaghi, R. Mattiuzzo, S. Bahl and A. Garg, "Incremental Small Delay Defect Methodology," in *IEEE Design Automation Conf.*, 2010.
[13] S. Goel, K. Chakrabarty, M. Yilmaz, K. Peng, and M. Tehranipoor, "Circuit Topology-Based Test Pattern Generation for Small-Delay Defects," *IEEE Asian Test Symposium (ATS'10)*, 2010.
[14] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-Pattern Selection for Screening Small-Delay Defects in Very-Deep Submicrometer Integrated Circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 760-773, 2010.
[15] K. Peng, J. Thibodeau, M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "A Novel Hybrid Method for SDD Pattern Grading and Selection," in *Proc. IEEE VLSI Test Symposium (VTS'10)*, 2010.
[16] K. Peng, M. Yilmaz, M. Tehranipoor, and K. Chakrabarty, "High-Quality Pattern Selection for Screening Small-Delay Defects Considering Process Variations and Crosstalk," in *Proc. IEEE DATE'10*, 2010.
[17] K. Peng, M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "A Noise-Aware Hybrid Method for SDD Pattern Grading and Selection," in *Proc. IEEE Asian Test Symposium (ATS'10)*, 2010.
[18] 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.
[19] IWLS 2005 Benchmarks, "http://iwls.org/iwls2005/benchmarks.html".