Compact Test Pattern Selection for Small Delay Defect

Chia-Yuan Chang, Kuan-Yu Liao, Sheng-Chang Hsu, James Chien-Mo Li, and Jiann-Chyi Rau

***Abstract*—This letter proposes an algorithm that selects a small number of test patterns for small delay defects from a large N-detect test set. This algorithm uses static upper and lower bound analysis to quickly estimate the sensitized path length so that the central processing unit (CPU) time can be reduced. By ignoring easy faults, only a partial fault dictionary, instead of a complete fault dictionary, is built for test pattern selection. Experimental results on large International Test Conference benchmark circuits show that, with very similar quality, the selected test set is 46% smaller and the CPU time is 42% faster than that of timing-aware automated test pattern generation (ATPG). With the proposed selection algorithm, small delay defect test sets are no longer very expensive to apply.**

***Index Terms*—Delay test, fault simulation, test generation.**

I. Introduction

Small delay defects (SDD) are gaining more and more attention in modern nanometer technology due to both increased frequency and shrinking geometry [1]–[3]. Detecting SDD requires both high-quality test patterns that activate and propagate fault effects via long paths. However, it has been demonstrated that timing-aware automatic test pattern generation (ATPG) is much slower and generates longer test sets than traditional timing-unaware ATPG [4]. As an alternative, test pattern selection from a timing-unaware N-detect ATPG test set has been proposed [5]. Although timing-unaware ATPG is faster than timing-aware ATPG, minimum test pattern selection is still time consuming because of the need to calculate path delay of every fault. To make test length short, building a complete fault dictionary for every fault and every pattern takes a lot of time and space. Recently, it has been shown that a majority of faults are easy faults that require no timing-aware ATPG [6].

This letter proposes a novel algorithm to quickly select a small test set for SDD detection. This letter has two contributions. The first contribution of this algorithm is static upper and lower bound (UB/LB) analysis that quickly decides whether a particular pattern drops a fault or not. If the static UB/LB meets certain conditions, then the fault can be dropped or undropped without detailed path length calculation. Second, this algorithm builds only a partial fault dictionary on those faults that have not been dropped before (likely to be hard faults) so that the dictionary size is very small. The reason for this proposal is that, if a test pattern does not detect a hard fault, it is not likely to be selected in the minimum test set. Top-off patterns are added to detect those transition faults that cannot meet the fault dropping criterion. Experimental results on large ITC benchmark circuits show that, on average, our proposed technique reduces runtime by 42% and test size by 46%, with a very similar quality to that of timing-aware ATPG.

The structure of this letter is as follows. Section II describes preliminary background of this research. Section III introduces our proposed technique. Section IV shows the experimental results on ISCAS and ITC benchmark circuits and Section V concludes this letter.

Manuscript received January 4, 2012; revised April 22, 1012, July 27, 2012, and November 26, 2012; accepted December 12, 2012. Date of current version May 15, 2013. This work was supported by the National Science Council, Taiwan, under Grant NSC 100-2628-E-002-015. This paper was recommended by Associate Editor A. E. Gattiker.

C. Chang, K. Liao, S. Hsu, and C. Li are with the National Taiwan University, Taipei 106, Taiwan (e-mail: r98943085@ntu.edu.tw; f97943076@ntu.edu.tw; r00943095@ntu.edu.tw; cmli@cc.ee.ntu.edu.tw).

J. Rau is with Tamkang University, New Taipei City 251, Taiwan (e-mail: jcrau@ee.tku.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2237946

II. Background

A. Past Research

A statistical delay quality model was first proposed to measure the quality of test patterns for SDD detection [3], [7]. Given a delay defect distribution function, a quantitative measure statistical delay quality level (SDQL) is calculated. Their fault simulation, however, required intensive calculation of sensitized path length for each fault and pattern.

Due to a large number of paths, generating a path delay fault test set for SDD detection is very difficult. In practice, high-quality transition fault test patterns have widely been used [8]. Test generation for timing-critical transitional faults was proposed in [9]. As late as possible ATPG has been proposed to detect transition fault via long paths [10]. Because there are too many paths in a circuit, KLPG selects K longest paths per gate when generating patterns [11]. Alternatively, a transition fault ATPG based on SOCRATES with propagation first or activation first heuristics was proposed in [12].

Instead of timing-aware ATPG, timing-unaware ATPG has been applied with some modification [5]. Short paths and intermediate paths are masks to force ATPG to generate patterns along long paths. After test generation, a subset of patterns is then selected. Alternately, selecting patterns from a large N-detect test set is also effective [4]–[6].

Since exact calculation of sensitized path length is slow, output deviation has been used as an alternative measure to select patterns [13]. Layout information can also be considered to take the crosstalk effect into account [14]. Although the output deviation calculation is very fast, it is just an indirect metric for path length.

B. Fault Dropping Criterion

A transition fault is detected if it is excited and the fault effect is propagated to any output(s) along any path(s). Traditionally, a fault is dropped as soon as it is detected by a test pattern. For SDD, a fault is dropped only if it is detected by a test pattern that meets a certain fault dropping criterion. There are different fault dropping criterions presented in the past research. In this letter, we follow the dropping based on slack margin (DSM) proposed in [15]

$$DSM = \frac{PD_f^s - PD_f^a}{T_{TC} - PD_f^a} < \delta \tag{1}$$

where PD_f^s is the structural longest path through fault f and PD_f^a is the actual longest path through fault f sensitized by the test set. T_{TC} is the test clock period, and we can easily see that $T_{TC} \geq PD_f^s \geq PD_f^a$. δ is a user-defined fault dropping threshold between zero and one. δ represents the ratio of the detectable fault size (numerator) to the slack of sensitized path through the fault (denominator). If δ is set to one, which is equivalent to the traditional transition fault dropping criterion, a fault is dropped as soon as it is detected. A smaller δ value means that PD_f^a is closer to PD_f^s, and thus a more stringent fault dropping criterion. In this letter, the default δ is set to 0.6, which is decided based on the tradeoff between test quality and test length in our experiments.

PD_f^s can simply be calculated by static timing analysis (STA), but PD_f^a needs to be dynamically computed for each pattern and each fault. For a transition fault f located at gate output z, PD_f^a is equal to the path delay through z (PD_z), which is the sum of AT_z and PT_z. The former represents the arrival time from circuit inputs (either primary inputs or flip-flops) to z. The latter represents the propagation time from z to circuit outputs (either primary outputs or flip-flops).

AT_z can be calculated in a forward direction from input to net z. When the gate output z changes from a controlled value to a noncontrolled value, AT_z is the latest arrival time of the gate inputs that belong to I_{cn}

$$AT_z = \underset{i \in Icn}{MIN} (AT_i + d_i) \tag{2}$$

where I_{cn} is the set of gat inputs that changes from a controlling to noncontrolling value. d_i is the gate delay from gate input i to output z. However, when the gate output z changes from a noncontrolled value to a controlled value, AT_z is the earliest arrival time of gate inputs in I_{nc} that changes from a noncontrolling value to a controlling value

$$AT_z = \underset{i \in Icn}{MIN} (AT_i + d_i). \tag{3}$$

PT_z can be calculated in a backward direction from the fault detecting outputs to net z. For a single gate, the propagation time from the sensitized gate input i to its gate output z is

$$PT_i = PT_z + d_i^v \tag{4}$$

where d_i^v is the gate delay to propagate the transition v at input i through the gate. v can be either a rising transition or a falling transition.

For a fanout stem without reconvergence, the propagation time is the maximum propagation of all its branches. For a fanout stem with reconvergence, the propagation time is the minimum propagation time of all its reconverging branches. This method takes the reconvergent multiple path sensitization into account.

Given a test pattern p, $PD_f^a(p)$ of a fault at net z is the summation of the arrival time to z and the propagation time from z to all sensitization paths

$$PD_f^a(p) = AT_z(p) + PT_z(p). \tag{5}$$

```
Select(T, FL, δ) // T=test set, FL=fault list, δ = threshold
1:  Static UB/LB Analysis;     // Section III-A
2:  D = BuildDictionary(T, FL, δ);     // Section III-B
3:  T* = SelectPatterns(D, FL, δ);     //Section III-C
4:  // top-off patterns, section III-D
5:  FL* = undropped fault list (undetected faults removed);
6:  D* = BuildDictionary(T-T*, FL*, 1.0);     // set δ =1.0
7:  T** = SelectPatterns(D*, FL*, 1.0);
8:  Ts=T*+T**;
9:  return (Ts);
```

Fig. 1. Overall algorithm.

Given a test set, PD_f^a is the largest $PD_f^a(p)$ among all test patterns that detect fault f

$$PD_f^a = \underset{p \in T_D}{MAX} \left\{ PD_f^d(p) \right\} \tag{6}$$

where T_D is the set of test patterns that detect fault f. Traditionally, for each detected fault f in every test pattern p, PD_f^a and DSM(f, p) have to be dynamically calculated so the runtime is very long.

C. Assumptions

This letter makes the following assumptions. First, when calculating the arrival time AT, we ignore the static hazard. This is a pessimistic assumption because, when static hazards occur, the actual arrival time can be larger than our calculated AT. Therefore, with hazards, the actual PD_f^a can be larger than our calculated PD_f^a, so the actual DSM of a fault can be lower than our estimation. This means that we select test patterns using a more conservative DSM.

Although we use the DSM fault dropping criterion in this letter as an illustration example, it should be noted that our proposed technique does not depend on any particular fault dropping criterion. The same technique can be applied to other fault dropping criteria as well.

III. Proposed Techniques

Fig. 1 shows the overall flow of this algorithm given three inputs: a test set T, a fault list FL, and a DSM threshold δ. First, the upper and low bounds of AT and PT are calculated in the preprocess stage. During the fault simulation, a partial dictionary D is built. Building this dictionary is fast because static UB/LB are used to reduce the number of simulation and path length calculation. A heuristic is used to select a small set, T^*. At the end, top-off patterns (T^{**}) are selected to detect those transition faults that cannot be dropped by the given DSM threshold. This can simply be done by repeating the above process with δ =1. Finally, the selected test set (T^s) is equal to T^* plus T^{**}.

A. Static UB/LB Analysis

Before pattern selection starts, a STA is needed to obtain PD_f^s for each fault. At the same time of STA, a static UB/LB analysis is also performed. For each net z, we calculate its static upper and lower bounds of PT from z to each structurally reachable output. Their definitions are shown as follows.

$PT_{UB}(z, \omega)$ = upper bound of PT from net z to output ω

$PT_{LB}(z, \omega)$ = lower bound of PT from net z to output ω where ω can be either a primary output or a pseudo primary output that is in the fan-out cone of net z. Static PT_{UB} and PT_{LB} can be calculated level by level in a backward direction, from output to input. For a gate input i and gate output z

$$PT_{UB}(i, \omega) = PT_{UB}(z, \omega) + d_i \quad (7)$$

$$PT_{LB}(i, \omega) = PT_{LB}(z, \omega) + d_i \quad (8)$$

where d_i is the gate delay from gate input i to output z. For a fan-out stem z, $PT_{UB}(z, \omega)$ is equal to the maximum $PT_{UB}(z, \omega)$ among all fan-out branches i and $PT_{LB}(z, \omega)$ is equal to the minimum $PT_{LB}(z, \omega)$ among all fan-out branches i

$$PT_{UB}(z, \omega) = \underset{b \in FObranch}{MAX}(PT_{UB}(b, \omega)) \quad (9)$$

$$PT_{LB}(z, \omega) = \underset{b \in FObranch}{MIN}(PT_{LB}(b, \omega)). \quad (10)$$

With static $PT_{UB/LB}$, we can dynamically estimate upper and lower bounds of sensitized path length for a test pattern. Given a fault f at net z and a test pattern p, $PD_{UB}(f, p)$ is $AT(z, p)$ plus the maximum PT_{UB} from z to all detecting PO/PPO. $AT(z, p)$ is the arrival time from launching inputs to net z. According to (2) and (3), $AT(z, p)$ can be calculated in a forward direction during the logic simulation. Launching PI/PPI are those primary inputs or pseudoprimary inputs that create the transition. Detecting PO/PPO are those primary outputs or pseudo primary outputs that detect the fault

$$PD_{UB}(f, p) = AT(z, p) + \underset{\omega \in \text{detecting } PO, PPO}{MAX}\{PT_{UB}(z, \omega)\}. \quad (11)$$

Similarly, $PD_{LB}(f, p)$ can be calculated in the same way. Please note that $PD_{UB}(f, p)$ and $PD_{LB}(f, p)$ are pessimistic since they are based on static structural information

$$PD_{LB}(f, p) = AT(z, p) + \underset{\omega \in \text{detecting } PO, PPO}{MIN}\{PT_{LB}(z, \omega)\}. \quad (12)$$

Dynamic $DSM_{UB}(f, p)$ and $DSM_{LB}(f, p)$ can simply be derived by $PD_{LB}(f, p)$ and $PD_{LB}(f, p)$, respectively. Please note that larger PD corresponds to smaller DSM

$$DSM_{UB} = \frac{PD_f^s - PD_{LB}(f, p)}{T_{TC} - PD_{LB}(f, p)} \quad (13)$$

$$DSM_{LB} = \frac{PD_f^s - PD_{UB}(f, p)}{T_{TC} - PD_{UB}(f, p)}. \quad (14)$$

Comparing $DSM_{UB/LB}(f, p)$ with the fault dropping threshold δ, there are three possible outcomes. If $DSM_{UB}(f, p)$ is smaller than δ, it is known that fault f is definitely dropped by pattern p. If $DSM_{LB}(f, p)$ is greater than or equal to δ, it is known that fault f is definitely not dropped by pattern p. In either case, exact $DSM(f, p)$ calculation is not needed. Otherwise, there is no conclusion as to whether f can be dropped by p or not so an exact DSM calculation is required.

Fig. 2 illustrates an example of the ISCAS S27 benchmark circuit. The gate delay values are shown beside each gate in the figure. Consider a gate output G11 (net z) slow-to-rise transition fault. In static UB/LB analysis, PT_{UB} and PT_{LB} from net z to outputs PO_1 and PO_2 are 2/2, and 3/5, respectively. It can be seen that the structurally longest path through this fault is $PD_f^s = 7$ (G1→G10→G22→PO1).

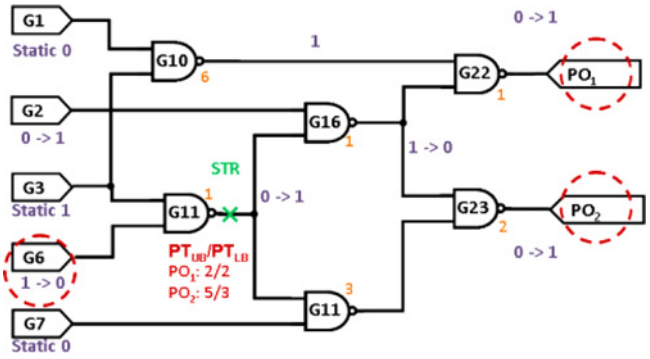

Fig. 2. S27 example.

Now consider a test pattern p, where G6 is the launching PI; PO_1 and PO_2 are both detecting PO. The arrival time of this pattern $AT(z, p)$ is simply 1. Path length sensitized by this pattern can be estimated as: $PD_{UB}(f, p) = 1 + \max(2, 5) = 6$, $PD_{LB}(f, p) = 1 + \min(2, 3) = 3$. Suppose that test clock period $T_{TC} = 8$. Therefore, $DSM_{UB}(f, p) = (7-3)/(8-3) = 0.8$, $DSM_{LB}(f, p) = (7-6)/(8-6) = 0.5$. If the fault dropping threshold $\delta > 0.8$, then f is definitely dropped by p. If $\delta \leq 0.5$, then f is not dropped by p. Otherwise, we would need to calculate the exact DSM for this pattern.

B. Build Dictionary

Fig. 3 shows the algorithm of BuildDictionary function. Initially, end_{undrop} and end_{all} both point to the end of fault list (line 2). For each test pattern p in the test set T, perform a single logic simulation on p (lines 3, 4). Move the pointer $fptr$ to the head of the fault list (FL). Assume that a 64-b simulator is used. Let F_{64} be the set of 64 faults on the right of $fptr$ excited by pattern p—a slow-to-fall/rise fault is excited by a falling/rising transition. Perform a parallel fault simulation on F_{64} (lines 6–8). For each detected fault f, if $DSM_{LB}(f, p)$ is larger than or equal to δ, then f cannot be dropped so no action is taken (lines 9–12). If DSM is smaller than δ, then f can be dropped, so we add pattern p into dictionary D (lines 13–22).

C. Select Patterns

Fig. 5 shows the algorithm to select a minimum test set given the partial dictionary D. Minimum test pattern selection and reducing pattern count without sacrificing test quality are minimum set covering problems (NP-hard), so we propose a two-stage heuristic to find a near optimal test set. In the first stage, essential patterns that drop unique faults in the dictionary are selected. All the faults dropped by the selected patterns are removed from the fault list. In the second stage, a greedy algorithm that iteratively selects the pattern that drops the most number of undropped faults is selected.

D. Top-Off Patterns

After our pattern selection, a small number of faults, especially those on structural long paths but sensitized through short paths that result in large DSM values (large numerator), may not be dropped by any pattern. Therefore, it is important to add top-off patterns to guarantee the transition fault coverage. Top-off patterns can easily be selected by the proposed

```
BuildDictionary(T, FL, δ)
1:  D = EMPTY;
2:  end_undrop=end_all= end of (FL);  //initially all faults are undropped
yet
3:  foreach pattern p in T {
4:      logic simulation (p);
5:      fptr = head (FL);
6:      while (fptr left to end_undrop) {  // undropped fault only
7:          F_64 = 64 excited faults to the right of fptr;
8:          parallel fault simulation (F_64, p);
9:          foreach detected fault f in F_64 {
10:             calculate DSM_LB(f, p)  DSM_UB(f, p);
11:             if ((DSM_LB(f, p) ≥ δ ) // f not dropped by p
12:                 next f;
13:             else if (DSM_UB(f, p) < δ) { //  f dropped by p
14:                 D = AddDicEntry (f, p, D, FL, δ);
15:                 next pattern p;
16:             }
17:             calculate exact DSM(f, p);
18:             if (DSM(f, p) < δ) {
19:                 D = AddDicEntry (f, p, D, FL, δ);
20:                 next pattern p;
21:             }
22:         }
23:         move fptr to the right of last simulated fault;
24:     }
25: return (D);
```

Fig. 3. Build dictionary.

```
AddDicEntry (f, p, D, FL, δ);
1:  insert new entry for p into D;  mark f as dropped by p in D;
2:  fptr+ = next fault right to f;
3:  while (fptr+ left to end_all) {  // simulate all faults
4:      F+_64 = 64 excited faults to the right of fptr+;
5:      parallel fault simulation (F+_64, p);
6:      foreach detected fault f+ in F+_64 {
7:          calculate DSM_UB(f+, p) and DSM_LB(f+, p);
8:          if ((DSM_LB(f+, p) ≥ δ )  // f+ not dropped by p
9:              next f+;
10:         else if (DSM_UB(f+, p) < δ) { //  f+ dropped by p
11:             mark f+ dropped by pattern p in D;
12:             move f+ to end of FL;  update pointers;
13:             next f+;
14:         }
15:         calculate exact DSM(f+, p);
16:         if (DSM(f+, p) <  δ) {  //  f+ dropped by p
17:             mark f+ dropped by pattern p in D;
18:             move f+ to end of FL;  update pointers;
19:         }
20:     }
21:     move fptr to the right of last simulated fault;
22: }
23: return (D);
```

Fig. 4. AddDicEntry.

algorithms with $\delta = 1$. (Please see lines 5–7 in Fig. 1.) For the undropped faults, a new dictionary D^* is built with $\delta = 1.0$. After test pattern selection with $\delta = 1$, a top-off test set T^{**} is selected. The final test set T^s is the SDD test set T^* plus the top-off test set T^{**}. Please note that top-off test pattern selection is performed on testable transition faults; those untestable transition faults have been removed from the fault list (line 5 in Fig. 1).

IV. Experimental Results

To validate the proposed technique, experiments are performed on large ISCAS'89 and ITC'99 benchmark circuits. The circuits are mapped to TSMC 0.13-μm technology and then placed and routed by commercial tools. To make a fair

```
SelectPatterns(D, FL)
1:  T* = EMPTY;
3:  foreach pattern p in D {
4:      if (p drops any unique fault)  // p is essential
5:          T* = T* + p;  // select p
6:          remove faults dropped by p from FL;
7:          remove p from D;
8:      }
9:  }
10: calculate number of dropped faults for each p in D;
11: repeat {
12:     p = pattern that drops most faults in FL;
13:     T* = T* + p;
14:     remove faults dropped by p from FL;
15:     remove p from D;
16:     calculate number of dropped faults for each p in D;
17: } until  no more fault dropped
18: return (T*);
```

Fig. 5. Select patterns.

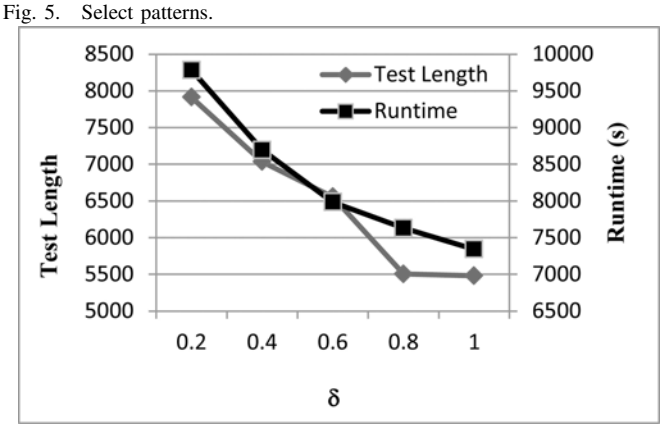

Fig. 6. Results of different δ (b18).

comparison, all the transition fault patterns generated from commercial timing-aware ATPG and commercial N-detect ATPG used the launch-off-capture method.

Table I compares the proposed pattern selection technique with existing commercial tools in terms of test length, pattern quality, and CPU time. There are five columns each under *Commercial TA Patterns* and *Commercial 10-detect Patterns*, which show the test length, DSM fault coverage, delay test coverage (DTC) [15], SDQL, and CPU time. The commercial 10-detect patterns are selected by the proposed technique. There are six columns under *Proposed Patterns*, which show the results of selected patterns. CPU_{sel} shows the required pattern selection time and CPU_{total} shows the required time of commercial 10-detect ATPG plus the proposed pattern selection time. The last two columns in Table I show the results of the proposed technique normalized to commercial timing-aware patterns in terms of test length and CPU time.

The fault dropping criterion δ is equal to 0.6 in the experiment. Our selected patterns are also evaluated by commercial tools in SDQL. The lower the SDQL, the higher the test quality. The SDQL parameters are set as follows: $a = 1.58\times10^{-3}$ $b = 2.1\times10^{-3}$ $\lambda = 4.96\times10^{-6}$. It can be seen that the selected patterns are 32% smaller and the CPU time is 10% larger, on average, as compared to the commercial timing-aware ATPG. On large circuits, such as b17 and b18, timing-aware ATPG runs much slower than 10-detect ATPG; the CPU time of our proposed technique is 42% shorter than timing-aware ATPG.

We did two variations based on the proposed pattern

TABLE I
TEST LENGTH, PATTERN QUALITY, AND CPU TIME ANALYSIS

Ckt	#Gate	Commercial TA Patterns					Commercial *10*-Detect Patterns					Proposed Patterns						Normalize	
		TL	DSM (%)	DTC (%)	SDQL	CPU (s)	TL	DSM (%)	DTC (%)	SDQL	CPU (s)	TL	DSM (%)	DTC (%)	SDQL	CPU_{sel} (s)	CPU_{total} (s)	TL	CPU
s35932	12K	94	86.1	76.5	317.7	39	288	86.9	77.9	297.6	40	106	86.9	76.7	310.0	9	49	1.13	1.26
s38584	22K	1250	90.6	83.8	156.7	120	5081	90.8	84.8	141.3	91	671	90.8	83.3	217.6	86	177	0.54	1.47
s38417	25K	644	97.7	90.8	176.1	82	2677	98.0	92.8	134.6	77	491	98.0	90.3	191.3	54	131	0.76	1.60
b17	34K	6197	79.7	55.1	6930.3	2595	18 532	80.0	56.5	6625.6	1271	3528	80.0	53.3	7139.1	541	1812	0.57	0.70
b18	76K	16 366	82.2	63.0	8482.9	17 230	32 366	82.2	62.6	8736.4	4086	6555	82.2	60.7	9264.9	3899	7985	0.40	0.46
Average	–	4910	87.3	73.9	3212.7	4013	11 789	87.6	74.9	3187.1	1113	2270	87.6	72.8	3424.6	918	2031	0.68	1.10

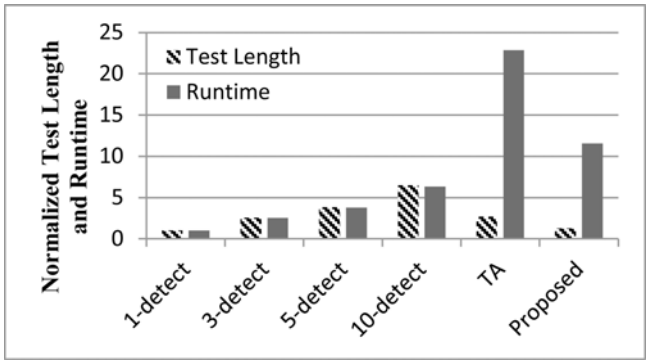

Fig. 7. Normalized test length & CPU time of different patterns (b18).

TABLE II
DIFFERENT TECHNIQUES

Ckt	Complete Dictionary w/o UB/LB Analysis			Complete Dictionary w/ UB/LB Analysis			Partial Dictionary w/ UB/LB Analysis		
	TL	CPU (s)	Mem (MB)	TL	CPU (s)	Mem (MB)	TL	CPU (s)	Mem (MB)
s35932	81	101	390	81	80	390	106	9	390
s38584	591	449	793	591	152	793	671	86	491
s38417	460	280	507	460	764	507	491	54	430
b17	3303	983	1150	3303	780	1150	3528	541	823
b18	6228	8945	3679	6228	5734	3679	6555	3899	2446
Avg	2133	2152	1304	2133	1502	1304	2270	918	916

selection algorithm to see the effectiveness of each component. Table II compares the experimental results. In the first experiment, static UB/LB analysis is turned off and a complete dictionary is built. In the second experiment, static UB/LB is turned on but a complete dictionary is still generated. The last experiment is performed with both static UB/LB analysis and partial dictionary turned on. Tradeoff between test length and memory usage can be observed in the table. Using a complete dictionary results in shorter test length but requires large memory, whereas partial dictionary results in longer test length but smaller memory requirement. It is shown that partial dictionary saves 30% memory and UB/LB analysis reduces approximately 57% CPU time.

Fig. 6 shows the results of SDD pattern selection using different fault dropping criteria for b18 circuit, $\delta = 0.2$–1.0. A smaller δ means a more stringent criterion. $\delta = 1.0$ is equivalent to a traditional transition fault ATPG. It can be observed that CPU time and test length increase as δ decreases. However, the improvement in SDQL quality is not significant when $\delta < 0.6$ so our default δ value is set to 0.6.

Fig. 7 compares the test length and CPU time of various test sets for the b18 circuit. The test length of N-detect transition fault test sets increases almost linearly with N. The test length of a timing-aware test set is 3.2 times larger and the CPU time is 27.3 times larger than $N = 1$. With our proposed selection technique, the test length is only 1.3 times larger and the CPU time is 11.4 times larger than $N = 1$.

V. CONCLUSION

This letter proposed a pattern selection technique for SDD detection to reduce test length as compared to commercial timing-aware ATPG. Static upper and lower bounds were used to analyze the structurally longest and shortest propagation time for each net to the circuit outputs. Dynamic upper and lower bounds were used to dynamically calculate the $DSM_{UB/LB}$, which reduced the CPU time. Moreover, this technique built a partial fault dictionary so that the test set can be compacted efficiently. The partial fault dictionary also resulted in less memory usage and CPU time than that of the complete fault dictionary. Experimental results showed that, on average, with very similar quality, the selected test set is 32% smaller than that of timing-aware ATPG.

REFERENCES

[1] S. Mitra, E. Volkerink, E. J. McCluskey, and S. Eichenberger, "Delay defect screening using process monitor structures," in *Proc. IEEE VLSI Test Symp.*, Apr. 2004, pp. 43–52.
[2] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Meirlevede, "Systematic defects in deep sub-micron technologies," in *Proc. IEEE Int. Test Conf.*, Oct. 2004, pp. 290–298.
[3] Y. Sato, S. Hamada, T. Maeda, A. Takatori, and S. Kajihara, "Evaluation of the statistical delay quality model," in *Proc. IEEE Asian South Pacific Des. Autom. Conf.*, Jan. 2005, pp. 305–310.
[4] 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 Symp.*, Apr. 2010, pp. 45–50.
[5] H. Lee, S. Natarajan, S. Patil, and I. Pomeranz, "Selecting high-quality delay tests for manufacturing test and debug," in *Proc. IEEE Int. Symp. Defect Fault Tolerance Very Large Scale Integr. Syst.*, Oct. 2006, pp. 59–70.
[6] S. K. Goel, N. Devta-Prasanna, and R. P. Turakhia, "Effective and efficient test pattern generation for small delay defect," *Proc. IEEE VLSI Test Symp.*, May. 2009, pp. 111–116.
[7] S. Hamada, T. Maeda, A. Takatori, Y. Noduyama, and Y. Sato, "Recognition of sensitized longest paths in transition delay test," in *Proc. IEEE Int. Test Conf.*, Oct. 2006, pp. 1–6.
[8] Y. Shao, I. Pormeranz, and S. M. Reddy, "On generating high quality tests for transition faults," in *Proc. Asian Test Symp.*, 2002, pp. 1–8.
[9] M. Kassab and J. Rajski, "Test generation for timing-critical transition faults," in *Proc. Asian Test Symp.*, 2007, pp. 493–500.
[10] P. Gupta and M. S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proc. IEEE Int. Test Conf.*, Oct. 2004, pp. 1053–1060.
[11] W. Qiu, J. Wang, D. M. H. Walker, D. Reddy, X. Lu, Z. Li, W. Shi, and H. Balachandran, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," in *Proc. IEEE Int. Test Conf.*, Oct. 2004, pp. 223–231.
[12] S. Kajihara, S. Morishima, A. Takuma, X. Wen, T. Maeda, S.Hamada, and Y. Sato, "A framework of high-quality transition fault ATPG for scan circuits," in *Proc. IEEE Int. Test Conf.*, Oct. 2006, pp. 1–6.
[13] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern grading and pattern selection for small-delay defects," in *Proc. IEEE VLSI Test Symp.*, Apr.–May 2008, pp. 233–239.
[14] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Interconnect aware and layout-oriented test-pattern selection for small delay defects," in *Proc. IEEE Int. Test Conf.*, Oct. 2008, pp. 1–10.
[15] X. Lin, K. Tsai, C. Wang, M. Kassab, J. Rajaski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware ATPG for high quality at-speed testing of small delay defects," in *Proc. Asian Test Symp.*, 2006, pp. 139–146.