First IEEE Int. Workshop on Reliability Aware System Design and Test (RASDAT 2010), pp. 46-51, Jan. 2010.

Optimizing Delay Test Quality with a Limited Number of Test Set

Michiko Inoue† Akira Taketani Tomokazu Yoneda† Hiroshi Iwata
Hideo Fujiwara†
Graduate School of Information Science, Nara Institute of Science and Technology
† Japan Science and Technology Agency, CREST

Abstract

To obtain high defect coverage for current nanometer VLSI design, timing-aware ATPGs are being developed to detect small delay faults. However, timing-aware test generation results in a large test set compared with test generation targeting traditional fault models such as stuck-at-fault or transition fault. The statistical delay quality level (SDQL) is proposed to evaluate test quality for small delay defects, and it is adopted as a measure for timing-aware test generation in several EDA tools. However, SDQL requires a high computation cost to evaluate a test set. In this paper, we address a problem to get a limited number of test set with high delay test quality based on SDQL. In the proposed method, we first generate a base test set and then efficiently select a test set with high delay test quality under a constraint on the number of test patterns. Experimental results demonstrate that our method can obtain a small test set with high quality SDQL within a reasonable time.

1 Introduction

Nanometer VLSI design is facing various problems on dependability. Resistive-open defects occur more often than before, and they lead to small delay faults[4]. In addition, further scaling down of transistors induces transistor aging problem[7]. To overcome these problem, testing that detects small delay faults or defects has an important role[1, 2, 3, 9].

The statistical delay quality model (SDQM) is proposed to evaluate test quality for small delay defects[5, 6], and it is adopted as a measure for timing-aware test generation in several EDA tools[2, 8, 9]. The SDQM has careful consideration into statistical delay defect distribution and evaluates not only test pattern quality but also quality of fabrication, design and test timing. Statistical delay quality level (SDQL) is a delay test metric based on SDQM. Though it is promising as a delay test quality measure, timing-aware ATPG and fault simulation based on SDQL tend to take long CPU time. In addition, a test set based on SDQL becomes large compared with test sets targeting other fault models. Therefore, it is important to obtain small test set with high delay test quality within a reasonable time.

Yilmaz et al. proposed test set pattern grading and selection method for small delay defects[10]. They avoided time-consuming timing-aware ATPG and evaluated delay test quality using the number of sensitized long paths, where a long path is a path with at least 70% of the clock period. They proposed a test set selection method based on their delay test metric.

In this paper, we address the problem to generate a small test set with high delay test quality measure. We adopt SDQL as a delay test quality. In our test generation flow, we first generate a test set called *a base test set* using exiting ATPG and select a limited number of test patterns so that their delay test quality are maximized. We proposed a test set selection method from a given base test set. The proposed method selects test patterns based on the lengths of sensitized paths, those are correlated with SDQL but can be efficiently evaluated. The proposed method avoids to apply time-consuming SDQL evaluation repeatedly, and therefore, selects a test set within a reasonable time. In the experiments, we demonstrate the efficiency of the proposed test set selection method, and also evaluate ATPG methods as base test set generators.

The rest of the paper is organized as follows. We introduce SDQM and SDQL in Section 2, and then propose a test set selection method in Section 3. Experimental results are given to evaluate the proposed test set selection method and ATPG methods for base test set generation in Section 4. Finally, Section 5 concludes this paper.

2 Statistical Delay Quality Model (SDQM)

In this section, we introduce statistical delay quality model (SDQM) and statistical delay quality level (SDQL) proposed by Sato et al.[5, 6]. The SDQM is proposed to

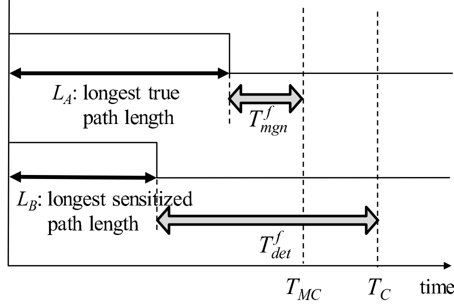

Figure 1. Timing relations for the two types of paths.

evaluate test quality based on a delay defect distribution function which is derived from fabrication process. The SDQL is a delay test quality metric that shows the amount of delay defects that should be detected but cannot be detected by a given test set.

The SDQM considers rising and falling delay faults on each of input and output pins of each gate. Though the number of faults is the same as transition faults, a delay defect size is associated with each fault. Figure 1 shows a concept of delay defect sizes that should be detected and can be detected by a given test set. Let f be a fault, and let L_A and L_B be the lengths of the longest true path passing through f and the longest path passing through f that is actually sensitized by the given test set, respectively. The true path is defined as a path that is designed to keep timing constraints. Let T_{MC} and T_C be system clock timing and test timing, respectively. The difference $T^f_{mgn} = T_{MC} - L_A$ is the minimum delay defect size that can affect system behavior and therefore should be detected. The difference $T^f_{det} = T_C - L_B$ is the minimum delay defect size that can be actually detected by the given test set.

The SDQL for a given test set is defined as follow, where N is the total number of faults and $F(s)$ is a delay defect distribution function.

$$SDQL = \sum_{f \in N} \int_{T^f_{mgn}}^{T^f_{det}} F(s)ds \qquad (1)$$

It shows the total delay test quality of the chip based on the delay defect test escapes. A shadow area in Fig.2 shows an amount of delay defect for one fault escaped during test. The SDQL is the total amount of such test escapes for the total faults. Therefore, smaller SDQL means better delay test quality.

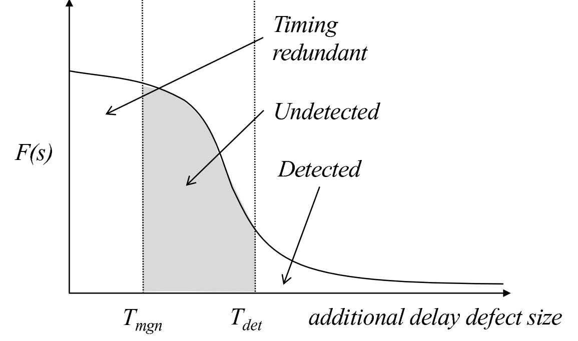

Figure 2. SDQL for one fault.

3 Test Set Selection Method

In this paper, we propose a method to generate a test set with the minimum SDQL under a constraint on the number of test patterns. We solve the problem by the following two steps.

1. Generate a test set T_{base} according to some criteria.
2. Select a limited number of test patterns with the minimum SDQL from T_{base}.

A test set T_{base} generated in Step 1 is called *a base test set*, and it is generated by existing ATPG tools in this paper. Therefore, we first consider how to select a test set with the minimum SDQL from a given test set T_{base}.

3.1 Simple Greedy Method

We can consider a simple greedy method as a heuristic to solve to select a test set with the minimum SDQL. We call the method SimpleGreedy.

SimpleGreedy
T: a given test set
P: the number of selected test patterns
T': a selected test set
1. Set $T' = \emptyset$
2. Repeat Steps 3 and 4 for P times
3. For each $t \in T$, calculate SDQL for a test set $T' + \{t\}$
4. For $T' + \{t\}$ with the minimum SDQL
   Set $T' = T' + \{t\}, T = T - \{t\}$

The above SimpleGreedy requires fault simulation $|T_{base}| - (i-1)$ times to obtain SDQL for each test set $T' + \{t\}$ in the i-th iteration. Therefore, fault simulation is applied $\sum_{i=1}^{P}(|T_{base}| - (i-1)) = \frac{1}{2}P(2|T_{base}| - P + 1)$ times. In general, fault simulation to obtain SDQL is time-consuming. The reason is as follows. In fault simulation for

Table 1. CPU time of fault simulation

circuit	#faults	#patterns	fault sim. CPU time(s)	
			transition	SDQL
b04	2,620	124	0.03	0.27
b12	4,794	698	0.14	3.29
b13	1,267	97	0.01	0.08
b15	26,428	1,793	1.98	86.30
b17	80,612	5,745	18.86	390.70
b18	223,312	13,030	185.07	5,501.43
b19	433,410	24,058	719.72	19,992.62
b20	46,538	3,894	12.87	1,504.55

SDQL, it cannot be accelerated by fault dropping like fault simulation for fault coverage. In case of fault simulation for fault coverage, once some fault is detected by some test pattern, the fault does not need to be cared by remaining test patterns. However, in fault simulation for SDQL, we have to consider not only detected or not but also the length of a sensitized path, and a fault can be dropped only when some test pattern detect the fault by the longest true path passing through the fault. Therefore, fault simulation for SDQL treats many faults for every test pattern and calculates the integration provided in Equation (1) many times. Table 1 shows CPU times required for fault simulation to obtain SDQL for ITC benchmark circuits, where we used TetraMAX with Small Delay Defect Test mode (Synopsys) and SunFireX4100 with AMD Opteron256 3.0GHz and 15GB memory (Sun Microsystems). The table compares CPU times between fault simulations for fault coverage for transition faults and SDQL. From the table, we can find that fault simulation for SDQL takes long computation time, and therefore, SimpleGreedy is impractical for large circuits.

3.2 Proposed Method

SimpleGreedy uses SDQL values to select a test pattern in each iteration, and therefore needs to apply time-consuming fault simulation. In contrast, the proposed test set selection method uses the length of the longest sensitized path for selection. The lengths of the longest sensitized paths for all the faults for a test set can be easily found without fault simulation, once we obtain the length of the longest sensitized path for each fault and each test pattern. The proposed method first obtains, for each test pattern, an SDQL value and the lengths of the longest paths sensitized by the test pattern for all the faults. Though this first step needs fault simulation for SDQL, we do not need further fault simulation to select test patterns.

First, we explain how to find the lengths of the longest paths for a test set without fault simulation. Assume that we already selected a set T' of $i-1$ test patterns and the lengths of the longest paths sensitized by T' for all the faults are known. We also know the lengths of the longest paths sensitized by each test pattern t for all the faults. Let $l_f^{T'}$ and l_f^t be the length of the longest path sensitized by T' and t for a fault f, respectively. It is obvious that the length of the longest path sensitized by $T' + \{t\}$ is $\max(l_f^{T'}, l_f^t)$, and hence, we can easily obtain the longest sensitized path lengths for $T' + \{t\}$.

In the proposed method, we use the sum of the longest sensitized path lengths for all the faults instead of a SDQL value. Though we do not directly evaluate SDQL values in each iteration like SimpleGreedy, the increase of the longest sensitized path length for some fault f implies the decrease of T_{det}^f. From Equation (1), it implies the decrease of SDQL. Let $L_{T'}$ denote the sum of the longest sensitized path lengths for test set T'. The sum $L_{T'+\{t\}}$ is obtained as follows.

$$L_{T'+\{t\}} = L_{T'} + \sum_{f \in N} \max(l_f^t - l_f^{T'}, 0) \quad (2)$$

Let us define $Gain_{T',t}$ as follows.

$$\begin{aligned} Gain_{T',t} &= L_{T'+\{t\}} - L_{T'} \\ &= \sum_{f \in N} \max(l_f^t - l_f^{T'}, 0) \end{aligned} \quad (3)$$

The proposed method selects a test pattern t with the largest $Gain_{T',t}$ as the i-th test pattern. The outline of the proposed method is as follows.

TestSetSelection
T: a given test set
P: the number of selected test patterns
T': a selected test set
1. Set $T' = \emptyset$
2. For each test pattern $t \in T$,
   apply fault simulation
   and obtain an SDQL value and l_f^t for each f.
3. Select t with the minimum SDQL, and
   set $T = \{t\}$.
4. Repeat Steps 5 and 6 for $P-1$ times
5. For each $t \in T$, calculate $Gain_{T',t}$
6. For t with the maximum $Gain_{T',t}$
   Set $T' = T' + \{t\}, T = T - \{t\}$

In the proposed method, we apply fault simulation for SDQL for a test set with one test pattern only $|T_{base}|$ times, and therefore, it can select a test set much faster than SimpleGreedy.

4 Experiments

We made experiments to evaluate the proposed test set selection method and also to analyze test generation methods suitable for base test sets. The experiment environment is the same as described in Section 3.

4.1 Evaluation of Selection Method

To evaluate the test quality of the proposed method, we compared the method with other selection methods. In the experiment, we generated base test sets using timing-aware ATPG (TetraMAX with Small Delay Defect Test mode). Table 2 shows the test generation results for ITC benchmark circuits, where TGT, FC and FE show test generation time, fault coverage, and fault efficiency, respectively. We can provide a delay defect distribution function $F(s)$ to TetraMAX in the form described as Equation (4).

$$F(s) = A \cdot e^{-Bs} + C \quad (4)$$

In this experiment, we set $A = 1, C = 0$, and set B so that $F(T_{MC}) = 0.1$ holds. The values of B are shown in the column "B in $F(s)$" in Table 2.

For comparison, we prepared two different selection methods: (1) select the first P test patterns generated by ATPG and (2) select a test pattern with the most gain for fault coverage as the i-th pattern. For the second selection method, called a *coverage based* method, we slightly modified Steps 4 and 5 in the proposed method so that it evaluates fault coverage of $T' + \{t\}$, or the number of faults detected by $T' + \{t\}$, instead of $Gain_{T',t}$. This evaluation is also possible without fault simulation, once we apply fault simulation for each test pattern at the beginning.

We applied the three test set selection methods for the base test sets in Table 2. We varied the number P of selected test patterns, and Table 3 shows CPU times for the proposed method and the coverage based method for $P = |T_{base}|$ where T_{base} is a base test set. That is, Table 3 shows upper bounds of the selection methods for any number P. The columns "fsim", "other" and "total" show CPU times for fault simulation, the other computation, and total computation, respectively. The proposed method takes longer CPU time than the coverage based method, since the proposed method applies fault simulation for SDQL at the beginning. However, the total CPU time is less than double of fault simulation time for the whole circuit shown in Table 1 except for a few small circuits.

Figures 3 and 4 shows relations between the number of selected patterns and SDQL values for the three methods. These figures shows SDQL improvement that is the improvement of SDQL value from the SDQL value for an empty test set. The SDQL value $SDQL_0$ for an empty test set means the total amount of delay defects that should be detected. The $SDQL_0$ and the SDQL improvement are given as follows.

$$SDQL_0 = \sum_{f \in N} \int_{T^f_{mgn}}^{\infty} F(s)ds \quad (5)$$

$$SDQL\ improvement = SDQL_0 - SDQL \quad (6)$$

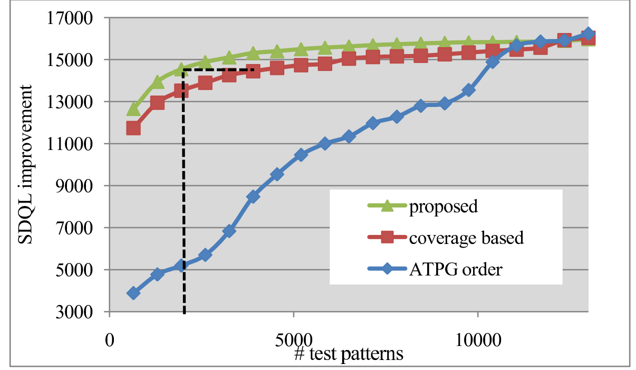

Figure 3. SDQL and selection method for b18.

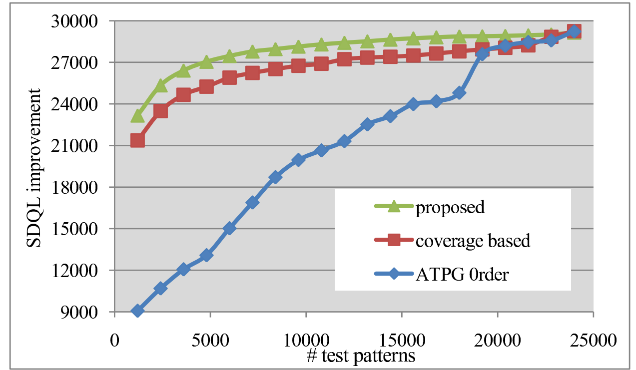

Figure 4. SDQL and selection method for b19.

In Fig.2, $SDQL_0$ corresponds to the area of "Undetected" and "Detected", and the SDQL improvement corresponds to the area of "Detected". Therefore, the SDQL improvement shows the total amount of detected delay defects.

From Fig.3 and Fig.4, we can find that both the proposed and coverage based methods efficiently improve SDQL compared with the simple selection based on ATPG order, and the proposed method can work most efficiently. Though it might seem that difference between the proposed method and the coverage based method is slightly little, there is a big difference between them. For example, if we try to obtain the same SDQL as obtained by the proposed method for $P = 2000$ for b18 (Fig.3), we need around double size of test set using the coverage based method.

4.2 ATPG for Base Test Set

Since the delay test quality of selected test sets depends on base test sets, in order to find a suitable base test set, we

Table 2. Timing-aware test generation result.

circuit	#gates	#FFs	#faults	#patterns	TGT(s)	FC(%)	FE(%)	SDQL	B in $F(s)$
b04	1,025	66	2,620	124	2.20	67.37	85.57	126.66	3.14
b12	1,730	121	4,794	698	6.71	88.40	91.20	51.27	4.22
b13	643	51	1,262	97	0.45	72.98	84.79	16.42	4.08
b14	8,460	215	22,904	1,325	2,429.06	84.27	86.74	8,497.64	0.70
b15	8,983	417	26,428	1,793	439.21	79.00	84.21	1,993.73	1.19
b17	27,766	1,317	80,612	5,745	2,193.66	85.96	88.17	5,654.23	1.19
b18	79,401	3,020	223,312	13,030	32,898.94	80.82	83.22	32,453.11	0.71
b19	152,599	6,042	433,410	24,058	103,917.61	81.24	83.04	63,921.51	0.71
b20	17,546	430	46,538	3,894	14,542.42	94.13	95.39	15,001.30	0.71

Table 3. CPU time of test set selection.

circuit	proposed			coverage based		
	fsim	other	total	fsim	other	total
b04	0.85	0.13	0.98	0.40	0.09	0.49
b12	5.20	0.94	6.14	1.50	0.65	2.15
b13	0.49	0.09	0.58	0.29	0.05	0.34
b14	190.34	5.55	195.89	25.23	2.92	28.15
b15	73.23	7.11	80.34	17.46	3.26	20.72
b17	537.21	65.72	602.93	175.17	23.57	198.74
b18	5,168.63	390.59	5,559.22	1,191.56	120.45	1,312.01
b19	24,727.96	1,828.55	26,556.51	4,807.91	534.31	5,342.22
b20	1,071.01	23.55	1,094.56	162.51	10.72	173.23

compared several base test sets generated by different test generation methods. In the experiment, we applied timing-aware ATPG and n-detect ATPGs for transition faults for $n = 1, 2, 4$. Table 4 shows the results on base test set generation and test set selection. Though the timing-aware ATPG needs longer test generation time than ATPG for transition faults, it results in better SDQL and fault coverage. Moreover, the timing-aware ATPG generates less test patterns than 2-detect and 4-detect ATPGs.

We selected test sets using the proposed selection method for four types of base test sets. Table 4 shows CPU times to select all the test patterns in the base test set. Figures 5 and 6 show the SDQL improvements from SDQL value for an empty test set for ITC benchmark b18 and B19. The SDQL values for an empty test set are 48,716.09 for b18 and 93,339.22 for b19. In the figures, "timing-aware(select)" and "timing-aware(ATPG)" means the results on the proposed test set selection and the simple selection using ATPG order, respectively. From these figures, we can find that timing-aware ATPG is suitable to obtain high delay test quality.

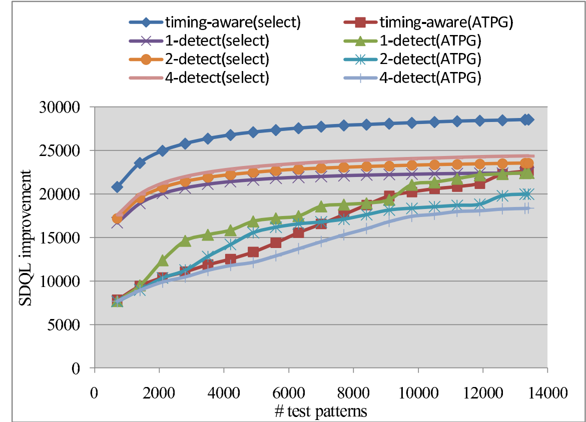

Figure 5. SDQL and base test set for b18.

5 Conclusions

In this paper, we consider how to obtain high delay test quality with a limited number of test set, and proposed a test set selection method from a given base test set. The proposed method selects test sets with small number of test patterns and high delay test quality based on SDQL within a reasonable time. Furthermore, we examined suitable test generation for base test sets, and found the timing-aware test generation can lead to high delay test quality.

Table 4. Test set selection for various base test sets.

circuit	b18							b19						
	base test set				selection			base test set				selection		
ATPG method	TGT (m)	#tp	SDQL	FC (%)	CPU time (m) fsim	other	total	TGT (m)	#tp	SDQL	FC (%)	CPU time (m) fsim	other	total
timing-aware	548.3	13,030	32,742	80.82	86.1	6.5	92.7	1732.0	240,58	64,190	81.24	412.1	30.5	442.6
1-detect	8.3	7,494	36,816	74.83	51.0	3.8	54.8	27.9	13,403	70,944	75.26	210.3	14.6	225.0
2-detect	14.1	13,755	36,045	75.51	103.4	7.0	110.4	50.4	24,789	69,528	76.06	328.1	26.7	354.9
4-detect	25.1	25,932	35,429	76.27	196.3	13.7	210.0	88.7	46,878	68,440	76.73	680.2	55.8	736.0

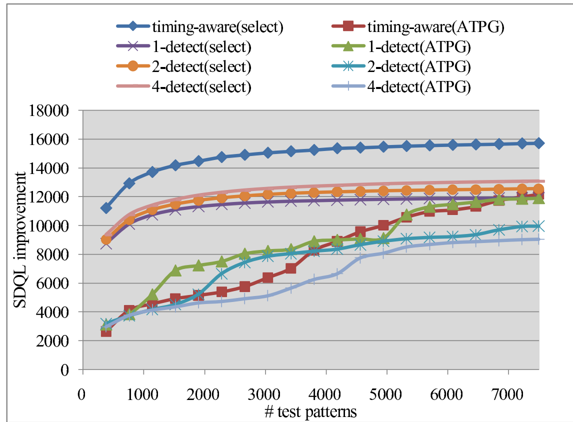

Figure 6. SDQL and base test set for b19.

Acknowledgment

The authors would like to thank Prof. Seiji Kajihara and Prof. Yasuo Sato with Kyusyu Institute of Technology and Prof. Satoshi Ohtake with Nara Institute of Science and Technology for their variable comments to this work.

References

[1] N. Ahmed, M. Tehranipoor, and V. Jayaram. Timing-based delay test for screening small delay defects. In *Proceedings of the 43rd annual Design Automation Conference*, pages 320–325, New York, NY, USA, 2006. ACM.

[2] X. Lin, K.-H. Tsai, C. Wang, M. Kassab, J. Rajski, 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 *ATS '06: Proceedings of the 15th Asian Test Symposium*, pages 139–146, Washington, DC, USA, 2006. IEEE Computer Society.

[3] R. Mattiuzzo, D. Appello, and A. Chris. Small-delay-defect testing. *EDN*, July 2009.

[4] P. Nigh and A. Gattiker. Test method evaluation experiments & data. In *Proceedings of International Test Conference*, pages 454–463, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[5] Y. Sato, S. Hamada, T. Maeda, A. Takatori, and S. Kajihara. Evaluation of the statistical delay quality model. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 305–310, New York, NY, USA, 2005. ACM.

[6] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara. Invisible delay quality ? sdqm. model lights up what could not be seen. In *Proceedings of the International Test Conference 2005*, page 47.1, 2005.

[7] Y. Sato, S. Kajihara, Y. Miura, T. Yoneda, S. Ohtake, M. Inoue, and H. Fujiwara. A circuit failure prediction mechanism (DART) for high field reliability. In *Proceedings of International Conference on ASIC, IEEE*, pages 581–584, 2009.

[8] Synopsys. *TetraMAX ATPG User Guide, Version C-2009.06-SP2*, September 2009.

[9] A. Uzzaman, M. Tegethoff, B. Li, K. M. Cauley, S. Hamada, and Y. Sato. Not all delay tests are the same - SDQL model shows true-time. In *Proceedings of the 15th Asian Test Symposium*, pages 147–152, Washington, DC, USA, 2006. IEEE Computer Society.

[10] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. Test-pattern grading and pattern selection for small-delay defects. In *Proceedings of VLSI Test Symposium*, pages 233–239, Los Alamitos, CA, USA, 2008. IEEE Computer Society.