IPSJ Transactions on System LSI Design Methodology Vol. 1 104–115 (Aug. 2008)

Regular Paper

Estimation of Delay Test Quality and Its Application to Test Generation

Seiji Kajihara,[†1] Shohei Morishima,[†1]
Masahiro Yamamoto,[†1] Xiaoqing Wen,[†1]
Masayasu Fukunaga,[†2] Kazumi Hatayama[†2]
and Takashi Aikyo[†2]

As a method to evaluate delay test quality of test patterns, SDQM (Statistical Delay Quality Model) has been proposed for transition faults. In order to derive better test quality by SDQM, the following two things are important: for each transition fault, (1) to find out the accurate length of the longest sensitizable paths along which the fault is activated and propagated, and (2) to generate a test pattern that detects the fault through as long paths as possible. In this paper, we propose a method to calculate the length of the potentially sensitizable longest path for detection of a transition fault. In addition, we develop a procedure to extract path information that helps high quality transition ATPG. Experimental results show that the proposed method improves SDQL (Statistical Delay Quality Level) by not only accurate calculation of the longest sensitizable paths but also detection of faults through longer paths.

1. Introduction

In deep-submicron VLSIs, faults that affect timing behavior of logic circuits more likely occur due to various reasons such as a resistive open/short, circuit noise, or process variation [1),2)]. Even for the increase of small delay, the circuits often would be affected. Therefore, conventional stuck-at fault testing is not sufficient and delay testing is becoming more and more important [3),4)].

When generating test patterns for delay testing, some delay fault models, which are transition fault, segment fault and path delay fault, are known [5)]. Among them the transition fault model is widely accepted. While detecting a delay fault requires a two pattern test, the first pattern to detect a transition fault is required only to set a logic value to the line where the fault is assumed, and the second pattern is required to make a transition at the line and propagates the value of the line to a primary output or a pseudo primary output that is a flip-flop. Since the second pattern is a similar to a test pattern to detect a stuck-at fault, ATPG and fault simulation for transition faults can be done by extending tools for stuck-at faults easily.

In general, test generation or fault simulation for transition faults does not take delay size caused by a defect into consideration. If a transition fault was counted as "detected" in fault simulation, the fault would be detectable necessarily with delay size larger than test timing. However the detection of faults with small delay is not guaranteed. Because the detectable delay size depends on generated test patterns, fault coverage does not indicate delay test quality well.

In order to evaluate transition delay quality of test patterns with considering probability of delay defect size, SDQM (Statistical Delay Quality Model) [6),7)] has been proposed instead of conventional fault coverage. The evaluation with SDQM is based on two delay sizes for each transition fault: the minimum delay size detectable by the test patterns and the minimum delay size that might affect circuit behavior. The former one is calculated by timed fault simulation for the test patterns. The later one is calculated from the delay of the longest sensitizable paths i.e. longest path length. Therefore, to derive better test quality by SDQM, there are two requirements as follows: for each transition fault, (1) to generate a test pattern that detects the fault through as long paths as possible, and (2) to find out the length of the longest sensitizable path along which the fault is activated and propagated. In Ref. 8), a statistical fault coverage metric has been proposed for path delay faults. The metric takes defects which cause transition faults into consideration.

Test generation methods to detect transition faults through long paths have been proposed in Refs. 9)–13). However, there is no efficient method developed to find out the longest sensitizable paths for transition faults. Note that, for path delay faults, there are many algorithms to find the (potentially) testable longest paths [14)–17)]. While detection of a path delay fault is based on single path sensitization, detection of a transition fault is based on multiple path sensitization.

†1 Kyushu Institute of Technology
†2 Semiconductor Technology Academic Research Center

In addition, conditions of path sensitization for transition faults are different between fault activation paths and fault propagation paths. Therefore finding out the longest sensitizable paths for transition faults is more complex than that for path delay faults.

In this paper, we propose a method to compute the length of the potentially sensitizable longest path for each transition fault. The method avoids selecting unsensitizable paths which are easily identified by necessary assignments [18] for detecting the transition fault. The path information calculated by the proposed method is optionally used in high quality transition ATPG. Experimental results show that the proposed method improves SDQL (Statistical Delay Quality Level) by not only accurate calculation of the longest sensitizable paths but also detection of faults through longer paths.

This paper is organized as follows. In Section 2, we explain SDQM and assumptions used in this work. In Section 3, we propose an algorithm to calculate upper bound of the length of the longest sensitizable path for each transition faults. In Section 4, we show a procedure to extract path information that helps transition ATPG. In Section 5, we give some experimental results for benchmark circuits, and in Section 6 we conclude this paper.

2. Preliminaries

2.1 SDQM

The statistical delay quality model (SDQM) [6),7)] has been proposed for the evaluation of delay test quality. The SDQM is generated by first assuming a delay defect distribution that is based on the actual defect probability in a fabrication process, and then investigating the sensitized transition paths and calculating their delay lengths. Detectable delay defect sizes are defined as the difference between the test timing and the path lengths. Finally, the probability of detecting small delay defects is calculated by multiplying the occurrence probability for each defect size. The calculated value is called the statistical delay quality level (SDQL). Note that, in this paper, test quality of test patters means the ability of fault detection and it is evaluated with SDQL.

We explain the SDQM using **Fig. 1** and **Fig. 2**. Assume that the delay of the longest sensitizable path that detects a transition fault is 5 ns and the delay of

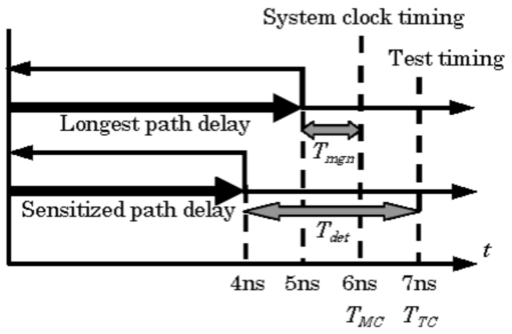

Fig. 1 Slack and detectable delay size.

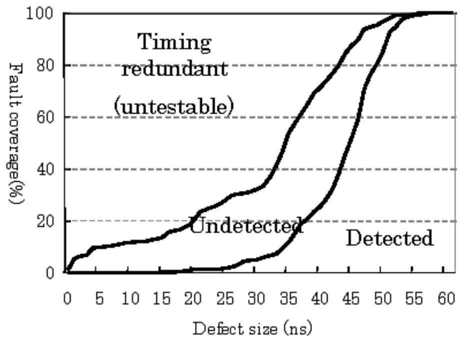

Fig. 2 Example of SDQL graph.

the sensitized path through which the fault is detected by generated test patterns is 4 ns. Also suppose the system clock timing T_{MC} is 6 ns and the test timing T_{TC} is 7 ns. For the fault, minimum detectable delay size T_{det} is 3 ns. Define the difference between the delay size of longest sensitizable path and T_{MC} as T_{mgn}. If delay size of a fault is less than T_{mgn}, the fault is untestable. In this case, T_{mgn} is 1 ns. So if delay size is greater than 1 and less than 3, the fault remains undetected. Depending on the delay size (defect size) for transition faults, fault coverage is changed, i.e., the percentages of untestable faults, detected faults and undetected faults are changed. Figure 2 shows an example of fault coverage graph. If the area indicating undetected is small, it means test quality of test patterns is high. For defect size s, SDQL is calculated as follows:

$$\sum_{K=1}^{2N} \int_{T_{mgn}}^{T_{det}} F(s)ds$$

where N is the number of circuit lines, i.e., $2N$ is the number of assumed transition faults and $F(s)$ is the probability of small delay defects of size s. The SDQL corresponds to the area indicating "undetected" when the distribution probability of delay size is uniform. Note that this work introduces the following assumptions in calculating SDQL:

- $F(s)$ is not varied depending on the location of lines.
- The delay of every gate is 0.2 ns under the unit delay model.
- T_{MC} and T_{mgn} are the same as the structurally longest path delay of the circuit.
- T_{det} is calculated by a timed fault simulator.

2.2 Assumption of Test Application

Delay testing requires application of two patterns at-speed. Even for delay testing, however, scan designs are still required and scan function are used for test application because test patterns with high fault coverage cannot be generated without scan function. There are two well-known methods to apply two-pattern tests for scan circuits. One is called LoC (Launch-off-Capture, or broad-side) method[19] and the other one is called LoS (Launch-off-Shift, or skewed-load)[20]. The difference of two methods is how to apply the second pattern of a two-pattern test. While LoC sets the second pattern by normal operation using a capture clock, LoS sets it by scan shift operation using a scan clock. Since LoS has more variations of second patterns, LoS can derive higher fault coverage than LoC. On the other hand, LoS requires more complex physical design than LoC for which standard scan design is enough. Therefore, in this work, we assume LoC as a test application method.

In ATPG for a scan circuit, an output of a scan flip-flop is regarded as a pseudo primary input (PPI), and an input of a scan flip-flop is regarded as a pseudo primary output (PPO). In this work, we assume the following test conditions as well as the work in 11):

- PPIs of the first patterns are controllable and PPOs of the second patterns are observable,
- Primary input values cannot be changed between the first pattern and the second pattern, that is, the second pattern takes the same primary input values as the first pattern. This assumption is introduced because of the difficulty of input change at speed by a LSI tester.
- Primary outputs are not observable between the first pattern and the second pattern.

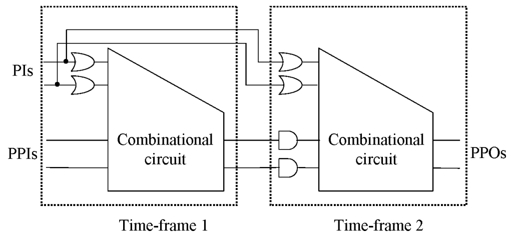

Fig. 3 Netlist transformation for restricted LoC.

In order to employ a combinational ATPG under these test conditions, we use a time expansion model of the circuit-under-test. The netlist of the circuit is transformed such that lines and gates which never reach to any pseudo primary output are removed, and each primary input is connected between the first time frame and second time frame as illustrated in **Fig. 3**[11]. By using such a modified netlist, we need not to change programs of combinational ATPG and fault simulation for the restricted LoC testing.

3. Longest path length for a transition fault

3.1 Overview

In Refs. 6), 7), T_{mgn} (explained in Fig. 1) for each fault was calculated by regarding a structurally longest path through the fault site as the longest path for the fault. However, the structurally longest path is not necessarily sensitizable, and in such a case delay test quality evaluated with SDQM would be pessimistic. For example, a circuit in **Fig. 4**, the structurally longest path for a rising transition fault at the output of FF_1 is path FF_1-A-B-E-F-FF_2. However, fault propagation along this path is blocked at gate F. The longest path to detect

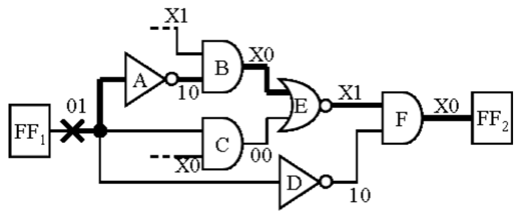

Fig. 4 Fault undetectable through the longest path.

the fault is path FF_1-D-F-FF_2. The length of the structurally longest path through a fault site is an upper bound of the length of the actual longest path to detect the fault. Although there is no known algorithm that efficiently finds the actual longest path length, the better upper bound of the longest path length makes test quality with SDQM be more accurate. We propose a method to find a better upper bound of the longest path length to detect a fault. In the rest of the paper, we refer to upper bound of the longest path length as the longest path length simply. Similarly, the longest path in this paper means the potentially sensitizable longest path.

For each fault, the proposed method calculates the path length after computing necessary assignments to detect the fault. The necessary assignments give constraints when computing possible transitions at lines on fault activation paths and possible faulty values at lines on fault propagation paths. A possible transition at a line is a transition that can be propagated to the faulty line from the line. Each line between the faulty line and PPIs takes one of “rising”, “falling”, “both rising and falling”, and “stable” as a possible transition. A possible faulty value at a line is one of “D only”, “D only”, “both D and D”, and “nothing”. Note that D and D are logic values of 5-valued logic used in ATPG [21]. Necessary assignments in the time-frame 2 can give constraints on the possible faulty values, but necessary assignments in the time-frame 1 are ignored to determine the possible faulty values. The possible transitions and the possible faulty values are determined such that there is no conflict with necessary assignments. For example, if a necessary assignment of a line on fault activation paths is 0 at the time-frame 1, the possible transition of the line cannot include “falling”.

The longest path length of the activation side and the propagation side are calculated separately. The length of activation paths is calculated as the maximum level from pseudo primary inputs to each line. The length of propagation paths are calculated as the maximum level from the fault site to each line. Note that the maximum levels are associated with rising transition and falling transition, respectively. And unlike path delay faults, multiple path sensitization is covered.

3.2 Procedure

The procedure of the proposed method is summarized as follows. For each fault,

(1) Compute necessary assignments for the fault.
(2) Calculate possible transitions at each line which is reachable to the fault site. This calculation is done toward pseudo primary inputs from the fault site.
(3) Calculate maximum logic level of each line which is reachable to the fault site. This calculation is done toward the fault site from pseudo primary inputs. The calculated logic level at the fault site is the length of the potentially sensitizable longest activation path.
(4) Calculate possible transitions at each line which is reachable from the fault site. This calculation is done toward pseudo primary outputs from the fault site.
(5) For each line which is reachable from the fault site, calculate possible faulty values D and/or D and maximum logic level from the fault site. This calculation is done toward pseudo primary outputs from the fault site.
(6) The maximum value among the calculated logic levels at pseudo primary outputs is the length of the potentially sensitizable longest propagation path.

Figures 5, **6** and **7** give an example of calculation of the longest activation path length where a falling transition fault is assumed at the output of gate G. At each line in Fig. 5, the two values like 10 or 1X show necessary assignments at the time-frame 1 and the time-frame 2, respectively. Assume that logic value 1 of the output of Gate A at the time frame 2 and the output of FF_4 at the time frame 1 are external constraints too. This calculation is done at Step 1 of the above procedure. The possible transition that can affect the transition at the fault site or the possible faulty value is described as “*r*”, “*f*”, “*b*”, or “*s*” in

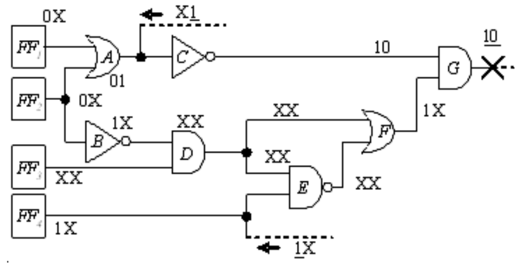

Fig. 5 Calculation of necessary assignments.

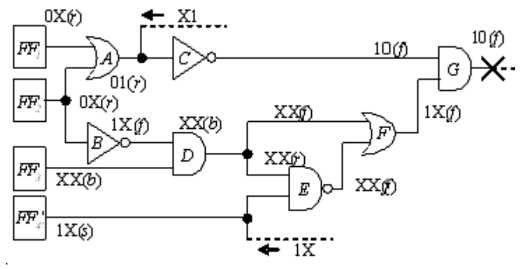

Fig. 6 Calculation of possible transitions.

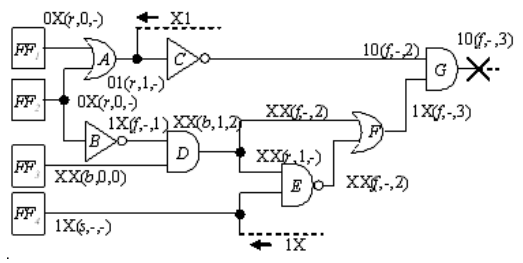

Fig. 7 Calculation of the maximum logic levels.

the parenthesis in Fig. 6. "*r*" means "rising". "*f*" means "falling". "*b*" means "both rising and falling". "*s*" means "stable" that are calculated at Step 2 of the procedure. Possible transition at the fault site is uniquely determined from the assigned value. Possible transition at each line which is reachable to the fault site is calculated with the implication of the assigned values and possible transitions at the output of the line. This calculation is done toward pseudo primary inputs from the fault site. For AND gates or OR gates, possible transition of an input is determined based on the possible transition of the gate output. However when the transition is inconsistent with the assigned value (e.g. value 1X for transition "*r*") possible transition becomes "*s*". Similarly, for NAND NOR, NOT gates, possible transition of an input is determined based on the opposite possible transition of the output as long as the transition is not inconsistent with the assigned value. Possible transition at a fanout stem is determined from those at its fanout branches. When both possible transitions "*r*" and "*f*" are included at its fanout branches, possible transition at the stem becomes "*b*". In Fig. 6, the output of FF_4 becomes "*s*" because assigned values 1X and the possible transition "r" at the fanout branch.

Two integers added to the parenthesis in Fig. 7 mean the maximum logic levels associated with "*r*" and "*f*", respectively. These are calculation at Step 3 of the above procedure. As a result, the structurally longest fault activation path is FF_2-*B*-*D*-*E*-*F*-*G*. From the maximum level at the gate output of "*G*", it is found that the length of the longest activation path is 3.

At Step 2 and Step 3, the potentially sensitizable longest activation paths are calculated. On the other hand, at Steps 4 to 6, the potentially sensitizable longest fault propagation paths arecalculated. These processes are similar to each other. Step 4 corresponds to Step 2, and Steps 5 and 6 correspond to step 3. By concatenating the longest activation paths and the longest propagation paths, the longest path for fault detection can be found.

3.3 Heuristics to Find Necessary Assignments

Since the proposed method computes the longest path such that there are no inconsistency with necessary assignments and transitions, the more necessary assignments are found, the better the upper bound of the longest path length is derived. There are some techniques to find more necessary assignments that have been developed as heuristics in ATPG [22)–24)]. The proposed method employs static learning [22)] and blockage learning [24)]. These techniques are done as a pre-processing and collect information that allows implication and fault propagation to assign more logic values.

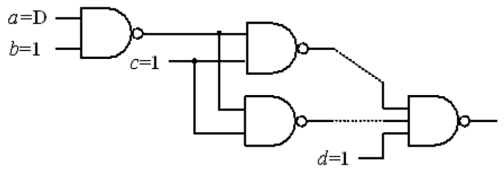

Fig. 8 Example of unique sensitization.

In calculating necessary assignments for each fault at Step 1 of the above procedure, unique sensitization[21),22)] is also helpful. There are two kinds of assignments by unique sensitization. When every fault propagation path passes a gate, logic values of the gate inputs may be assigned, as b=1 and d=1 in **Fig. 8**. Finding such a case is not time-consuming, and we refer to this type of heuristics as $HL1$. Another type of unique sensitization is shown as assignment c=1 in Fig. 6 too, where assignment of a logic value to a fanout is necessarily not to block any propagation. We refer to this type of heuristics as $HL2$, which is time-consuming.

SDQL is calculated from the longest path length for each fault and generated test patterns. We can calculate the longest path length independently of ATPG, i.e., it can run with ATPG in parallel. Therefore a time-consuming procedure as well as ATPG would be acceptable, if information on the longest path is not needed in ATPG.

4. Application to Transition ATPG

4.1 Path Information for ATPG

The procedure described above aimed at computing only the length of the longest path. When computing the longest path length for a fault, the longest activation path and the longest propagation path are found. Though the paths are not necessarily sensitizable, if the fault is detected through the paths, a test pattern with the best test quality for the fault would be generated. So we prepare procedures to extract the longest fault activation path and the longest fault propagation path, and to generate a test pattern that sensitizes the extracted paths. The test generation procedure consists of internal value assignments to sensitize the paths and justification for the assignments. And if the extracted paths is used in ATPG, the calculation of the longest paths has to be completed before test generation (cannot do them in parallel).

Path information consists of a sequence of lines indicating a path and logic values to control the path. When we extract path information for ATPG, not only assignments to on-path inputs but also assignments to off-path inputs are sometimes necessary. Note that, for a path P, if line l_i is included in P, l_i is referred to as an on-path input of P. A line l_j is referred to as an off-path input associated with on-path input l_i of P if l_j is not an on-input but it drives a gate G that is also driven by l_i. The assignments to off-path inputs are needed to block early arrival of a transition to fault activation paths or early arrival of a fault effect to fault propagation paths.

4.2 Example

For a circuit in **Fig. 9**, suppose a falling transition fault at the output of gate D. The notations in Fig. 9 are given with a similar manner to Fig. 5, except that logic values with under lines were X in necessary assignments for the faults. The longest fault activation path FF_2-B-C-D and the longest fault propagation path D-F-I-J-FF_4 are described in bold lines. In addition, as the fault propagation path D-G-J-FF_4 is sensitized simultaneously, it is described in bold lines too. Information on the longest paths is extracted based on the maximum level, and passed to ATPG. In order to guarantee to cause a falling transition at the gate output of D through three gates from PPIs, a transition at the gate output of A must not come to gate D. Therefore 1 with an underline at the gate output of A is determined and passed to ATPG too. Similarly, logic values with under lines are determined and passed to ATPG to ensure the sensitization of the longest paths. As a result, path information is composed of paths shown as bold lines and underlined logic values.

5. Experimental Results

We implemented the proposed method on a workstation (CPU: Itanium2 1.6 GHz, Memory: 16 GB OS: RedHat Linux 2.6) using C language and applied it for ISCAS'89 benchmark circuits and an industrial circuit i_145 with 145,472 gates. **Table 1** shows results on upper bounds of the longest path length. In this experiment we prepared two programs with different heuristics "$HL1$" and "$HL2$" to find necessary assignments, as described in Section 3.3. Since the

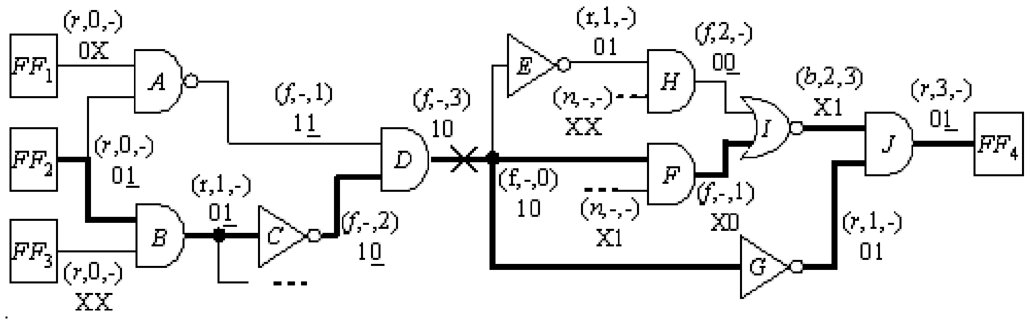

Fig. 9 Extraction of path information for ATPG.

Table 1 Results on average of the longest path length.

Circuits	Faults	Ave. of Length			Ave. of Length top 30%			Time (sec)	
		Str	HL1	HL2	Str	HL1	HL2	HL1	HL2
s420	384	9.14	6.58	6.42	12.00	8.15	8.15	0.02	0.02
s1423	2250	30.96	26.25	25.84	55.73	46.80	46.30	0.26	1.02
s5378	5108	13.13	13.09	12.98	17.60	17.53	17.31	0.59	2.02
s9234	10636	28.46	26.13	25.96	46.64	41.46	41.21	2.38	397
s13207	14812	23.76	21.52	20.89	49.74	43.34	41.43	4.22	541
s15850	17628	29.62	28.20	27.72	60.35	57.07	55.94	3.87	1061
s35932	53980	19.28	17.40	17.38	28.00	24.38	24.35	419	3487
s38417	49040	21.48	21.00	20.99	32.89	32.09	32.08	5.36	1917
s38584	52274	13.70	12.68	12.48	24.95	22.36	21.86	166	9333
i_145	488434	11.11	10.82	10.80	19.11	18.68	18.63	248	61983

longest path length is computed for each transition fault, we compared average of the obtained longest path length with the structural longest path length. In Table 1, the second column shows the number of transition faults. The 3rd to 5th columns give results of the average path length. The column "*Str*" shows the structural longest path length and the columns "$HL1$" and "$HL2$" show our results. For a fault whose structural longest path length is small, the effect of making the longest path length accurate might be little. Therefore, in the 6th to 8th columns, we show average path lengths for faults whose structural longest path length includes in the top 30%. CPU times of $HL1$ and $HL2$ are also given.

The proposed method $HL1$ and $HL2$ could find the longest path length more accurately than the structural longest path length. **Figure 10** visualizes upper bounds of fault coverage for each delay size caused by transition faults in circuit s1423. We could observe that $HL1$ improves upper bound significantly as well as $HL2$. $HL2$ gives better improvement than $HL1$, but CPU time of $HL2$ is much longer. Therefore, if the calculation of the longest path length is performed with ATPG in parallel, $HL2$ may be acceptable. However, if it should be done before ATPG, $HL1$ is recommended. The proposed method is effective for faults with long structural longest paths.

Table 2 shows results of ATPG using the longest path information as described in Section 4. In order to show the effectiveness of the proposed method, we implemented four ATPG and compared their results. In the Table, "TR" means simple transition ATPG without considering path length, and "TA" means timing-aware ATPG proposed in Ref. 11). Because i_145 was too large to run the timing-aware ATPG, we describe as "NA" in the table. ATPGs "$HL1+TR$" and "$HL1+TA$" try to generate test patterns for the extracted paths by $HL1$ first, then apply TR and TA, respectively, if test generation fails for the extracted paths. The columns "Ave. of Length" shows average length of paths through which faults are detected. The columns "$\#ofTests$" show the number of generated two-pattern tests, and the column "time" shows CPU time in second. The number of test patterns is increased when each fault is detected through a longer path. This is because the number of faults detected by one test pattern is decreased. It means that test cost and test quality are still trade-off. **Fig. 11** visualizes fault coverages of generated test patterns for s1423. If TA is used, high quality test patterns are generated, but it's time consuming. Because, by combining $HL1$ with TR or TA, test patterns with higher test quality could be generated easily.

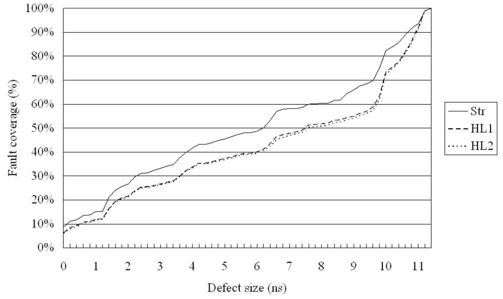

Fig. 10 Upper bound of fault coverage.

In **Table 3**, we show how test quality calculated by SDQL was improved. The columns "$HL1$" are evaluation results for test patterns "TR" using $HL1$ described in Section 3. Even though ATPG is not changed (path information is not used), $HL1$ contributed to the improvement of SDQL more than 40% on average. But if path information derived by $HL1$ was used in ATPG, more improvement of SDQL can be achieved. Also it can be seen that the improvements of SDQL given in Table 3 are much larger than those of the average length of fault detection paths given in Table 2. Such a thing is caused by the reason why a small delay defect more likely occurs than large one.

Table 2 ATPG results on average path length for detected faults.

	Ave. of Length				# of Tests				Time (sec)			
Circuits	TR	TA	HL1+TR	HL1+TA	TR	TA	HL1+TR	HL1+TA	TR	TA	HL1+TR	HL1+TA
s420	6.24	6.33	6.296	6.33	25	21	28	29	0.02	1.02	2.02	3.02
s1423	11.22	16.96	14.78	18.91	139	206	148	242	0.73	127	0.57	85.3
s5378	11.98	12.30	12.24	12.39	339	403	379	448	2.93	788	4.38	820
s9234	22.28	23.53	22.62	23.74	857	953	839	1095	27.0	38818	51.6	35808
s13207	20.01	20.29	20.10	20.31	812	934	838	984	27.5	2695	41.7	2230
s15850	19.34	21.03	19.53	21.18	659	891	703	959	27.0	163798	45.8	128828
s35932	16.60	16.70	16.67	16.70	225	219	763	229	105	2199	845	3355
s38417	19.58	20.34	19.86	20.44	2271	3376	2605	4037	148	13552	553	16632
s38584	11.03	11.26	11.06	11.28	2028	2468	2111	2707	369	102919	1068	108183
i_145	10.29	NA	10.55	NA	12277	NA	15782	NA	8165	NA	17924	NA

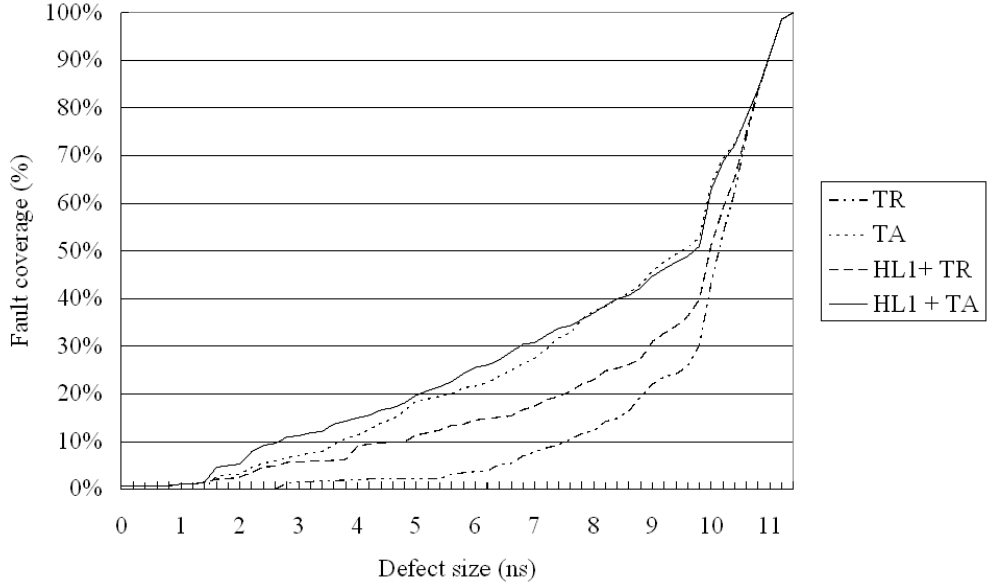

Fig. 11 fault coverage of generated test patterns.

Table 3 Improvements of SDQL.

Circuits	SDQL (ppm)				Ratio		
	TR	HL1	HL1+TR	HL1+TA	HL1	HL1+TR	HL1+TA
s420	0.508	0.033	0.021	0.019	93.59%	95.83%	96.16%
s1423	1.592	1.239	1.103	1.028	22.15%	30.69%	35.44%
s5378	1.650	1.636	1.510	1.448	0.87%	8.48%	12.27%
s9234	7.116	6.837	6.700	4.840	3.93%	5.84%	31.99%
s13207	7.754	2.140	1.914	1.905	72.40%	75.32%	75.43%
s15850	7.361	5.424	5.396	5.125	26.32%	26.70%	30.38%
s35932	120.005	31.731	29.826	29.408	73.56%	75.15%	75.49%
s38417	2.620	1.321	1.090	0.735	49.58%	58.40%	71.94%
s38584	4.593	2.408	2.365	2.229	47.58%	48.50%	51.47%
i_145	12.426	10.572	4.378	NA	14.92%	64.77%	NA
Average					40.49%	47.21%	53.40%

6. Conclusion

In this paper we proposed a method to compute the length of the potentially sensitizable longest paths for detection of transition faults. In addition, we gave a procedure to extract path information that helps high quality transition ATPG. Experimental results showed that the proposed method improves SDQL by not only accurate calculation of the longest sensitizable paths but also detection of faults through longer paths.

References

1) Cheng, K.-T., Dey, S., Rodgers, M. and Roy, K.: Test Challenges for Deep Sub-Micron Technologies, *Proc. Design Automation Conf.*, pp.142–149 (2000).
2) Chen, L.-C., Gupta, S.K. and Breuer, M.A.: High Quality Robust Tests for Path Delay Faults, *Proc. VLSI Test Symp.*, pp.88–93 (1997).
3) Kruseman, B., Majhi, A.K., Gronthoud, G. and Eichenberger, S.: On hazard-free patterns for fine-delay fault testing, *Proc. International Test Conference*, pp.213–222 (2004).
4) Mitra, S., Volkerink, E., McCluskey, E. and Eichenberger, S.: Delay defect screening using process monitor structures, *Proc. VLSI Test Symposium*, pp.43–52 (2004).
5) Bushnell, M.L. and Agrawal, V.D.: *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*, Kluwer Academic Publishers (2000).
6) Sato, Y., Hamada, S., Maeda, T., Takatori, A. and Kajihara, S.: Evaluation of the statistical delay quality model, *Proc. Asian and South Pacific Design Automation Conference*, pp.305–310 (2005).
7) Sato, Y., Hamada, S., Maeda, T., A.T., Nozuyama, Y. and Kajihara, S.: Invisible delay quality — SDQM model lights up what could not be seen, *Proc. International Test Conference*, p.47.1 (2005).
8) Qiu, W., Lu, X., Wang, J., Li, Z., Walker, D. and Shi, W.: A statistical fault coverage metric for realistic path delay faults, *Proc. VLSI Test Symposium*, pp.37–42 (2004).
9) Shao, Y., Pomeranz, I. and Reddy, S.: On Ggenerating High Quality Tests for Transition Faults, *Proc. Asian Test Symposium*, pp.1–8 (2002).
10) Yang, K., Cheng, K.-T. and Wang, L.-C.: TranGen: A SAT-Based ATPG for Path-Oriented Transition Faults, *Proc. Asian and South Pacific Design Automation Conference*, pp.92–97 (2004).
11) Kajihara, S., Morishima, S., Takuma, A., Wen, X., Maeda, T., Hamada, S. and Sato, Y.: A Framework of High-Quality Transition Fault ATPG for Scan Circuits, *Proc. Int'l Test Conf.*, p.2.1 (2006).
12) Lin, X., Tsai, K.-H., Wang, C., Kassab, M., Sato, Y., Hamada, S. and Aikyo, T.: Timing-Aware ATPG for High Quality At-speed Testing of Small Delay Defects, *Proc. Asian Test Symposium*, pp.139–146 (2006).
13) Uzzaman, A., Tegethoff, M., Li, B., K.M., Hamada, S. and Sato, Y.: Not all Delay Tests Are the Same — SDQL Model Shows True-Time, *Proc. Asian Test Symposium*, pp.147–152 (2006).
14) Li, W.-N., Reddy, S.M. and Sahni, S.K.: On Path Selection in Combinational Logic Circuits, *IEEE Trans. CAD*, Vol.8, No.9, pp.56–63 (1989).
15) Murakami, A., Kajihara, S., Sasao, T., Pomeranz, I. and Reddy, S.M.: Selection of Potentially Testable Path Delay Faults for Test Generation, *Proc. Int'l Test Conf.*, pp.376–384 (2000).
16) Sharma, M. and Patel, J.H.: Finding a Small Set of Longest Testable Paths that Cover Every Gate, *Proc. Int'l Test Conf.*, pp.974–982 (2002).
17) Qiu, W. and Walker, D.M.H.: An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit, *Proc. Int'l Test Conf.*, pp.592–601 (2003).
18) Rajski, J. and Cox, H.: A Method to Calculate Necessary Assignmentsin Algorithmic Test Pattern Generation, *Proc. Int'l Test Conf.*, pp.25–34 (1990).
19) Savir, J.: On broad-side delay testing, *Proc. VLSI Test Symposium*, pp.284–290 (1994).
20) Savir, J.: Skewed-Load Transition Test: Part I, Calculus, *Proc Int'l Test Conf.*, pp.705–713 (1992).
21) Abramovici, M., Breuer, M.A. and Friedman, A.D.: *Digital Systems Testing and Testable Design*, IEEE Press, Piscataway, New Jersey (1990).
22) Schulz, M., Trischler, E. and Sarfert, T.: SOCRATES: A Highly Efficient Automatic Test Pattern Generation System, *IEEE Trans. on CAD*, pp.126–137 (1988).
23) Fujiwara, H. and Shimono, T.: On the Acceleration on Test Generation Algorithms, *IEEE Trans. on Comp.*, Vol.C-32, pp.1137–1144 (1983).
24) Wang, C., Pomeranz, I. and Reddy, S.: REDI: An Efficient Fault Oriented Procedure to Identify Redundant Faults in Combinational Circuits, *Proc. Int'l Conf. on CAD*, pp.370–374 (2001).

(Received December 25, 2007)
(Revised March 17, 2008)
(Accepted May 1, 2008)
(Released August 27, 2008)

(Recommended by Associate Editor: *Makoto Sugihara*)

Seiji Kajihara received the B.S. and M.S. degrees from Hiroshima University, Japan, and the Ph.D. degree from Osaka University, Japan, in 1987, 1989, and 1992, respectively. From 1992 to 1995, he worked with the Department of Applied Physics, Osaka University, as an Assistant Professor. In 1996, he joined the Department of Computer Science and Electronics of Kyushu Institute of Technology, Japan, where he is a Professor currently. His research interest includes test generation, delay testing, and design for testability. He received the Young Engineer Award from IEICE in 1997, the Yamashita SIG Research Award from IPSJ in 2002, and the Best Paper Award from IEICE in 2005. He is a member of the IEEE, the IEICE, and the IPSJ.

Shohei Morishima received his B.E. degrees in Computer Science and Electronics, and M.E. degrees in Creation Informatics Program, from Kyushu Institute of Technology, Japan, in 2005 and 2007, respectively. Currently he is working in Toshiba Corp., Japan. His research interest includes delay testing and test generation.

Masahiro Yamamoto received his B.E. in Computer Science and Electronics from Kyushu Institute of Technology, Japan, in 2006. Currently he is studying towards the M.E. degree in the Creation Informatics Program at Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology. His research interest includes delay testing and design for testability.

Xiaoqing Wen received the B.E. degree from Tsinghua University, Beijing China, in 1986, the M.E. degree from Hiroshima University, Hiroshima, Japan, in 1990, and the Ph.D. degree from Osaka University, Osaka, Japan, in 1993. From 1993 to 1997, he was a Lecturer at Akita University. He was a Visiting Researcher at University of Wisconsin, Madison, U.S.A., from Oct. 1995 to March 1996. He joined SynTest Technologies, Inc., USA, in 1998, and served as its CTO until 2003. In 2004, he joined the Kyushu Institute of Technology, Iizuka, Japan, where he is currently a Professor. His research interests include VLSI test, diagnosis, and testable design. He is a senior IEEE member, a IEICE member, and a REAJ member.

Masayasu Fukunaga received the B.S and M.E. degrees from Meiji University, Kawasaki, Japan in 2000 and 2002 respectively, and the Ph.D. degree in computer science and system engineering from Kyushu Institute of Technology, Iizuka, Japan in 2005. He joined Fujitsu Limited in 2005. From 2006 to 2008, he was a researcher in Semiconductor Technology Academic Research Center. He is currently with Fujitsu Microelectronics Limited. His research interests include delay testing and design for testability.

Kazumi Hatayama received his B.S., M.S. and Ph.D. degrees in applied mathematics and physics from Kyoto University, Kyoto, Japan, in 1976, 1978 and 1982, respectively. He is now a Team Leader in Test & Diagnosis Group in Semiconductor Technology Academic Research Center. He is Asia Pacific Regional Chair of Test Technology Technical Council, IEEE Computer Society, Program Chair of ATS'08, IP Track Co-Chair of VTS'08 and a PC member of ITC and other conferences. He is a senior member of IEEE and a member of IEICE, IPSJ, ORSJ and REAJ. His research interests include DFT, BIST, ATPG, fault diagnosis and fault tolerant computing.

Takashi Aikyo recieved the B.E. degree in electronics from Shibaura Institute of Technology, in 1978, and M.E. degree in electronic and communications engineering from Waseda University, in 1980. He joined Fujitsu Limted in 1980, and he was working on development of computer-aided design tools in LSI testing area. He is currently a senior manager at the Semiconductor Technology Academic Reserch Center, and he is also a Ph.D. student at the Graduate School of Science and Engineering, Ehime University. His reserch interests include design for testability, and fault diagnosis. He is a member of IEICE and IEEE Computer society.