IPSJ Transactions on System LSI Design Methodology Vol. 3 283–291 (Aug. 2010)

Regular Paper

On Delay Test Quality for Test Cubes

Shinji Oku,[†1] Seiji Kajihara,[†1,†2] Yasuo Sato,[†1,†2]
Kohei Miyase[†1,†2] and Xiaoqing Wen[†1,†2]

This paper proposes a method to compute delay values in 3-valued fault simulation for test cubes which are test patterns with unspecified values (Xs). Because the detectable delay size of each fault by a test cube is not fixed before assigning logic values to the Xs in the test cube, the proposed method only computes a range of the detectable delay values of the test patterns covered by the test cubes. By using the proposed method, we derive the lowest and the highest test quality of test patterns covered by the test cubes. Furthermore, we also propose a GA (genetic algorithm)-based method to generate fully specified test patterns with high test quality from test cubes. Experimental results for benchmark circuits show the effectiveness of the proposed methods.

1. Introduction

In deep sub-micron circuits, small delay faults caused by various defects such as resistive opens, resistive bridges or circuit noises, are increasing[1)]. Therefore it is difficult to keep high test quality by the conventional stuck-at fault testing, and delay testing is getting more and more important.

As a fault model for delay testing, the transition delay fault model is often used[2)]. However, a major problem of the transition fault model is that delay sizes of transition faults are not taken into consideration. That is, even though a fault is judged as detectable by fault simulation, it may not be detected because of its too small delay size. Therefore simple fault coverage for the transition fault model is not enough.

Sato, et al. proposed the SDQM (Statistical Delay Quality Model) as a method to evaluate delay test quality of test patterns considering quality of manufacturing process, delay margin of the design, accuracy of test timing, and test pattern quality for transition faults[3),4)]. The SDQL (Statistical Delay Quality Level) was also proposed as a quantified value of the SDQM. The SDQL is calculated for fully specified test patterns in which a logic value 0 or 1 is assigned to all bits[5)]. This is because signal propagation delay, which is necessary to calculate the SDQL, is not fixed unless unspecified values (Xs) are assigned to fixed values.

Recently test cubes, which are test patterns with unspecified values, are preferably utilized for solving various testing problems such as test compression[6),7)], X-masking for output compression[8)], test power reduction[9),10)], or improvement of test quality[11)]. Although the stuck-at fault coverage can be calculated easily for test cubes, it is difficult to evaluate delay test quality for transition faults using the SDQM before assigning logic values to the Xs in the cubes because no delay calculation method for transition faults has been established yet.

Therefore, if a proper method to calculate the SDQL for test cubes is established, we are able to obtain the range of the lowest and the highest delay test quality of the test cubes. Using the information, we are able to assign logic values to Xs in a balance between test quality, test compaction ratio and low test power, etc. In addition, we are also able to evaluate delay test quality, when test cubes include unknown values that come from outputs of SRAM, analog modules, or non-scan FFs, etc.

In this paper, we propose a novel method to estimate signal delay in the transition fault simulation for test cubes, and try to find the range of the SDQL of test patterns obtained by assigning arbitrary logic values to the Xs in the test cubes. By using the method, we can derive the lowest and the highest test quality of test patterns covered by the test cubes. Therefore when one tries to improve delay test quality of test cubes, a possible target of improvement would be clear and test quality would be easily consistent with other test compression or test power reduction requirements.

In order to calculate the SDQL, two types of delay values should be computed: one is arrival times of input transitions at a faulty site, and another is an arrival time of the faulty value at each observable point. The proposed method estimates both the maximum and the minimum values of these arrival times, and then calculates the range of the SDQL of test patterns covered by the test cubes. In experiments, we compare the range of the SDQL obtained by the proposed

†1 Kyushu Institute of Technology
†2 JST, CREST

© 2010 Information Processing Society of Japan

method with that of fully specified test patterns obtained from test cubes. Note that this work targets full-scan sequential circuits and employs a unit delay model in delay calculation. An X-filling method, which is based on by using a GA (Genetic Algorithm), is also shown to derive test patterns with high delay test quality from test cubes.

This paper is organized as follows. In Section 2, we explain the concept of SDQM and assumptions used in this work. In Section 3, we explain the problem to calculate delay values for test cubes. In Section 4, we propose methods of delay calculation in fault simulation for transition faults. In Section 5, we explain the X-filling method for high delay test quality. In Section 6, we introduce some experimental results for benchmark circuits, and finally we conclude the paper in Section 7.

2. Preliminaries

2.1 The Statistical Delay Quality Model

The statistical delay quality model (SDQM)[3),4)] was proposed for the evaluation of delay test quality for given test patterns. The model assumes a delay defect distribution that is based on the actual defect probability in a fabrication process, and then investigates the sensitized transition paths and calculates their delay lengths. Detectable delay defect sizes are defined as the difference between the test timing and the tested path lengths. Therefore, the probability of missing small delay defects is calculated by multiplying the occurrence probability for each defect size. The value calculated as a defect level is called the statistical delay quality level (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 the sensitized path through which the fault is detected by the 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 $3\,ns$ $(7\,ns - 4\,ns)$. 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 for any test pattern. In this case, T_{mgn} is $1\,ns$ $(6\,ns - 5\,ns)$. So if a delay size is greater than 1 and less than 3, the fault remains undetected. Depending on the delay

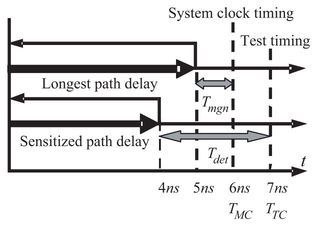

Fig. 1 Slack and detectable delay size.

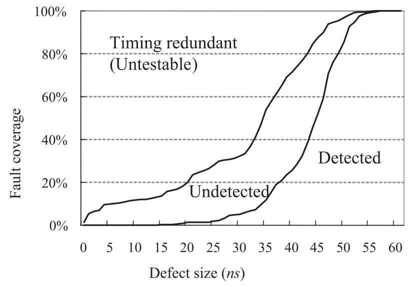

Fig. 2 Example of fault coverage.

size (defect size), its fault coverage varies significantly, i.e., the percentages of untestable faults, detected faults and undetected faults vary. Figure 2 shows an example of those fault coverage graph. If the area indicating undetected faults is small, it means test quality of test patterns is high. Let $T_{det}(f_k)$ and $T_{mgn}(f_k)$ be T_{det} and T_{mgn} for fault f_k. SDQL is calculated as follows:

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

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 faults when the distribution probability of delay size is uniform. Note that this work introduces the following assumptions in calculating the 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.

© 2010 Information Processing Society of Japan

- T_{MC} and T_{mgn} of each fault are able to be calculated using the structurally longest path delay of the circuit.
- T_{det} is calculated by a timed fault simulator.

While T_{MC} and T_{mgn} are test pattern independent, T_{det} must be calculated for each given test pattern. T_{det} is calculated from two types of delay values: (1) an arrival time of a transition at a faulty site from flip-flops, (2) arrival times of the faulty value at observable points from the faulty site. The sum of these two delay values is the sensitized path delay, and T_{det} is the difference between the sensitized path delay and test timing T_{TC}. In this paper, to derive the SDQL for test cubes, we propose a method to compute these values in 3-valued fault simulation. From the maximum and the minimum values of sensitized path delay for test patterns covered by the test cubes, we can obtain the range of T_{det}.

2.2 Test Application

Delay testing requires application of two successive patterns at test-timing. Even for delay testing, however, scan designs are still required and scan function is used for test application because test patterns with high fault coverage can hardly be achieved without scan designs. In this work, we assume LoC (Launch-off-Capture, or broad- side) method as a test application method of two-pattern test [12]. LoC sets the first pattern by scan shift operation, and sets the second pattern by normal operation using a capture clock. 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 Ref. 13):

- 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 take the same primary input values as the first pattern.
- Primary outputs are not observable.

In order to employ a combinational ATPG under these test conditions, we use a time expansion model of the circuit. The net-list of the circuit is transformed such that all 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** [13].

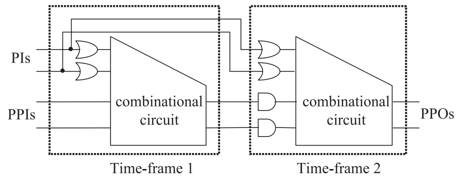

Fig. 3 Netlist transformation [13].

3. Delay Calculation in 3-valued Simulation

We explain a difference of delay calculation between 2-valued simulation and 3-valued simulation using an example. In case of 2-valued simulation, logic values of all lines are fixed, and for every line it is fixed whether a signal transition occurs, or not. Therefore the arrival time of a transition at each line is uniquely determined as shown in **Fig. 4**. In the Figure, the numbers in parenthesis of each line means the arrival time of transition at the line. When transitions from a controlling value to its non-controlling value are propagated on a gate, the arrival time at the gate output is determined from the latest arrival time at the gate inputs. When transitions from a non-controlling value to its controlling value are propagated on a gate, the arrival time at the gate output is determined from the earliest arrival time at the gate inputs. In case of Fig. 4, the transition occurring at the output of FF_1 arrives at the input of FF_5 at delay time 3.

In **Fig. 5**, we consider a case of 3-valued simulation. Suppose that flip-flop FF_1 has an unspecified value X at the first vector. If logic value 0 is assigned to the X, a transition occurs at the output of FF_1 and it reaches to FF_5 at delay time 3. On the other hand, if logic value 1 is assigned to the X, a transition which occurs at the output of FF_4 reaches to FF_5 at delay time 1. Thus when unspecified values are included in test patterns, the arrival times of transition are not uniquely determined, and every time a transition passes a gate the number of arrival times may increase. In general when the number of Xs is n in a test cube, $2n$ arrival times at a line are possible at the maximum. Since it is meaningless for evaluation of test cubes to compute all the possible arrival times, we compute the minimum value and the maximum value of the arrival times. Similarly, for the

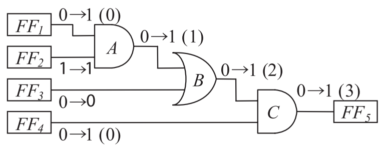

Fig. 4 Delay calculation in 2-valued simulation.

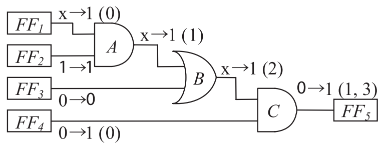

Fig. 5 Delay calculation in 3-valued simulation.

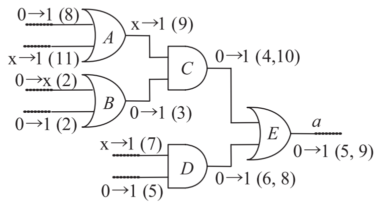

Fig. 6 Calculation of arrival time of transition.

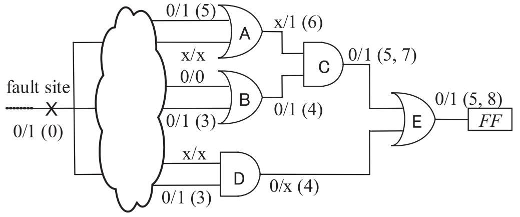

Fig. 7 Calculation of arrival time of faulty value.

calculation of the arrival time of a faulty value at observable points, we compute only the minimum value and the maximum value.

4. Delay Calculation for a Test Cube

4.1 Arrival Time of Transition

We first explain the calculation of the arrival time of transition at a line using a sub-circuit in **Fig. 6**. Suppose that signal transitions propagate to line a which is the output of the sub-circuit. Since Xs exist on paths to a, the arrival time of the transition cannot be fixed as described in Section 3. At the outputs of gates C and D, two arrival times are possible. We calculate the minimum and maximum value of the arrival time, respectively. For example, if an X arriving at gate input D is 0, the arrival time at the output of gate D is 8. And if an X arriving at gate input D is 1, the arrival time at the output of gate D is 6. At line a, since a transition from a non-controlling value to its controlling value passes gate E, the arrival time at a is determined from the earliest arrival time at the gate inputs. Since the minimum value of the earliest arrival time at the gate inputs is 4, the minimum value of the arrival time at a becomes 5. Similarly, the maximum value of the arrival time becomes 9.

4.2 Arrival Time of Faulty Value

Next, we explain the calculation of the arrival time of faulty values at observable points which are scan flip-flops. In **Fig. 7**, suppose that a transition fault occurs at the input line of the sub-circuit and the faulty effect is propagated to FF_1. In the Figure, two logic values added to each line indicate "fault-free value/faulty value" of the second vector, and the numbers in parentheses mean the range of the arrival time of faulty values. The calculation of the arrival time of faulty values starts from the fault site. Therefore the arrival time at the fault site is 0. The arrival time is calculated only for lines which have a possibility to be different between the fault-free value and the faulty value. Note that, in calculation of the arrival time of faulty values, we don't have to check the transitions of logic values on the fault propagation paths.

When an X value exists on a fault propagation path, the arrival time of faulty values cannot be fixed. However, the minimum and maximum values of the arrival times are calculated as well as the calculation of arrival time of transitions. In Fig. 7, because an X appears at gate A, the minimum and maximum arrival times are calculated at gate C. At gate D, the faulty value of the gate output is an X. Because the value may be different from the fault-free value, the arrival time at the gate output must be calculated.

When more than one fault effect propagates through a gate and the faulty

© 2010 Information Processing Society of Japan

value is a controlling value of the gate, the arrival time of the faulty value at the gate output is determined from the latest arrival time at the gate inputs. On the other hand when the faulty value is a non-controlling value of the gate, the arrival time at the gate output is determined from the earliest arrival time at the gate inputs. At gate E in Fig. 7, since the faulty value is a controlling value, the arrival time at the output of E is determined from the earliest arrival time at the gate inputs. Since the minimum value of the earliest arrival time at the gate inputs is 4, the minimum value of the arrival time at the gate output is 5. Similarly, the maximum value of the arrival time is 8.

4.3 Calculation of T_{det}

From the calculated arrival times of transition and faulty values, sensitized path delay which is the sum of these two delay values is calculated. Then, as shown in Fig. 1, T_{det} is calculated as the difference between the sensitized path delay and test timing T_{TC}. Since the arrival times have a range, T_{det} has a range too.

While it is not guaranteed that there is a test pattern that brings the maximum arrival times, it is guaranteed that there is no test pattern that brings the arrival time exceeding the maximum arrival times. Therefore the range of T_{det} gives an upper bound and a lower bound of T_{det} for test patterns covered by the test cubes.

5. X-filling for High Delay Test Quality

5.1 Test Generation Targeting Small Delay

In Section 3 and Section 4, we explained the calculation method of two types of delay values. Using the method, we estimate the range of the SDQL of test patterns covered by the test cubes.

Even if we get the lower bound of the SDQL (that is, highest quality), the method to generate the test patterns with the highest quality is not known yet. Therefore, in this section, we propose a novel X-filling method to generate test patterns with high quality for delay test. In the method we employ a GA-based (Genetic Algorithm based) method to assign logic values to Xs, because the GA is easy to implement and CPU time is easily controllable.

5.2 GA-based X-filling

The GA is the well-known approach that emulates evolution of animate beings. In **Fig. 8**, we show the flow of the proposed test generation method that uses the GA. The details of the flow are as follows:

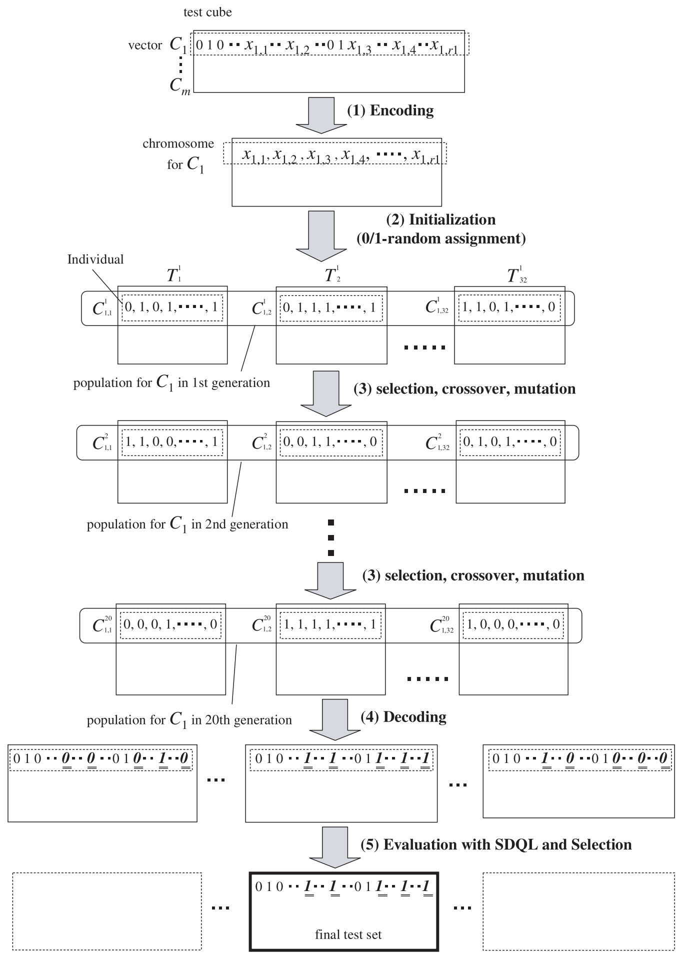
test cube
vector C_1 0 1 0 ·· $x_{1,1}$ ·· $x_{1,2}$ ·· 0 1 $x_{1,3}$ ·· $x_{1,4}$ ·· $x_{1,r1}$
C_m
(1) Encoding
chromosome for C_1
$x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}, \cdots, x_{1,r1}$
(2) Initialization (0/1-random assignment)
Individual
T_1^1 T_2^1 T_{32}^1
$C_{1,1}^1$ 0, 1, 0, 1,····, 1
$C_{1,2}^1$ 0, 1, 1, 1,····, 1
$C_{1,32}^1$ 1, 1, 0, 1,····, 0
population for C_1 in 1st generation
(3) selection, crossover, mutation
$C_{1,1}^2$ 1, 1, 0, 0,····, 1
$C_{1,2}^2$ 0, 0, 1, 1,····, 0
$C_{1,32}^2$ 0, 1, 0, 1,····, 0
population for C_1 in 2nd generation
(3) selection, crossover, mutation
$C_{1,1}^{20}$ 0, 0, 0, 1,····, 0
$C_{1,2}^{20}$ 1, 1, 1, 1,····, 1
$C_{1,32}^{20}$ 1, 0, 0, 0,····, 0
population for C_1 in 20th generation
(4) Decoding
0 1 0 ·· 0 ·· 0 ·· 0 1 0 ·· 1 ·· 0
0 1 0 ·· 1 ·· 1 ·· 0 1 1 ·· 1 ·· 1
0 1 0 ·· 1 ·· 0 ·· 0 1 0 ·· 0 ·· 0
(5) Evaluation with SDQL and Selection
0 1 0 ·· 1 ·· 1 ·· 0 1 1 ·· 1 ·· 1
final test set

Fig. 8 Flow of X-filling.

© 2010 Information Processing Society of Japan

(1) Encoding
Each set of Xs derived from a given test cube C_i is defined as the *chromosome* for C_i ($i = 1$ to m), where m is the number of given test cubes.

(2) Initialization (0/1-random assignment)
Logic values of 0 or 1 are randomly assigned to the Xs in the chromosome for C_i, and this is defined as an *individual*. 32 individuals in the 1st generation $C^1_{i,1}, C^1_{i,2}, \cdots, C^1_{i,32}$ are generated. $C^k_{i,j}$ means a j-th individual for test cube C_i in the k-th generation. The set of these individuals is defined as the *population* for C_i. A fully specified test set T_j ($j = 1$ to 32) in the k-th generation consists of m test patterns obtained by decoding individuals $C^k_{1,j}, C^k_{2,j}, \cdots, C^k_{m,j}$.

(3) Evolution for the next generation using *Selection*, *Crossover*, and *Mutation*
The GA is applied for each individual for C_i in the k-th generation. The fitness value of each individual is calculated from the longest path length to detect a fault with the vector that corresponds to the individual. This is because if a test pattern detects a fault through a longer path, test quality measured with the SDQL becomes better. The processes of Selection, Crossover, and Mutation are then applied for individuals in the population as follows.

3-A) 4 individuals that have the large fitness values are selected as individuals $C^{k+1}_{i,1}, C^{k+1}_{i,2}, \cdots, C^{k+1}_{i,4}$ in the next generation.

3-B) Further 4 individuals except ones selected at 3-A) are randomly picked up and the individual with the largest fitness value is selected from them as a parent. In the same way, another individual which has not been selected as a parent or an individual in the next generation is selected as a parent.

3-C) For the selected parents, Crossover and Mutation are applied. In Crossover, that is the two-point crossover, both parents are divided into three parts of bits as shown in **Fig. 9**, and the middle parts are exchanged each other. They are called the *children* of the parents. Then, Mutation is applied for each bit of the children. In Mutation, each bit is inverted at the random probability of 0.1%. After Mutation, the children are determined as individuals in the next generation. These steps are repeated until 28 (= 32 − 4) children $C^{k+1}_{i,5}, C^{k+1}_{i,6}, \cdots, C^{k+1}_{i,32}$ are determined at last.

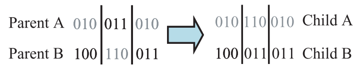

Fig. 9 Two-point crossover.

3-D) The steps of $3-A$, $3-B$ and $3-C$ are applied for each individual for C_i ($i = 1$ to m) in each generation. Evolution for the next generation using Selection, Crossover, and Mutation is repeated 20 times (20 generations).

(4) Decoding
All the individuals in the last generation are converted to the sets of test vectors (i.e. 0/1-assigned test cubes), and 32 test sets are obtained finally.

(5) Evaluation and selection with SDQL
A SDQL value of each test set is calculated and the test set with the lowest SDQL (the highest test quality) is selected as the final solution.

6. Experimental Results

We implemented the proposed methods on a PC (Celeron 2.67 GHz, 512 MB, RedHat Linux 2.6) using C language and applied for full-scan versions of ISCAS'89 and ITC'99 benchmark circuits. The test cubes used were obtained by applying test relaxation [14] for fully specified test patterns for transition faults generated by an in-house ATPG tool. **Table 1** gives the number of the test cubes, the percentage of unspecified values in the test cubes, CPU time to compute the maximum and the minimum delay times, and CPU time to compute delay time for fully specified test patterns after filling logic value 0 to all Xs in the test cubes (0-filling). The proposed method calculates the maximum and the minimum delay times separately.

In **Table 2**, we show the normalized values of SDQL computed from the maximum delay, the minimum delay, 0-fill, random-fill, and the original test patterns before test relaxation, where the reference value is the SDQL computed from the

Table 1 Statistics of test cubes and CPU time.

			CPU time [sec]			
circuit	test cubes	Xs[%]	Upper bound	Lower bound	0-fill	GA
b15s	1141	45.1	39	42	37	6496
b17s	1250	43.0	120	137	117	26181
b20s	989	29.5	28	32	27	6940
b21s	923	28.5	27	30	26	7291
b22s	1123	37.2	49	57	48	13635
s35932	337	92.0	77	80	74	839
s38417	270	46.9	4.5	5.5	4.4	2491
s38584	412	73.7	32	36	32	2801

Table 2 Normalized SDQL.

circuit	Upper bound	original	0-fill	random	Lower bound
b15s	1	0.9662	0.9658	0.9636	0.9299
b17s	1	0.9590	0.9569	0.9533	0.9243
b20s	1	0.9769	0.9765	0.9752	0.9355
b21s	1	0.9692	0.9686	0.9685	0.9337
b22s	1	0.9726	0.9720	0.9716	0.9255
s35932	1	0.9808	0.9768	0.9902	0.9385
s38417	1	0.8679	0.8610	0.8577	0.7202
s38584	1	0.9831	0.9829	0.9824	0.8415

minimum delay. The smaller the SDQL value is, the higher the delay test quality is. The SDQL obtained from the minimum delay means the upper bound of the SDQL (delay test quality guaranteed by the test cubes). In the experiment, there was no case that the SDQL obtained from the test cubes is higher than that of the minimum delay or lower than that of the maximum delay.

In **Tables 3** and **4**, we give the SDQLs of test patterns obtained by the proposed GA-based test generation method. Table 3 shows the average SDQLs of test sets obtained from the populations in four generations, 1st, 5th, 10th, 20th and 50th generations. The results for toth generation is shown for reference. Table 4 shows the best SDQL of test sets in each generation. These values are the normalized value computed from the upper bound of the SDQL. In the Tables, the values of "Lower bound" are shown again. From these tables, we can find that the SDQL improves as the generation increases. However, the improvement is small. It means that either the lower bound of the SDQL is very far from the real minimum SDQL of test patterns covered by the test cubes, or there is room for the improvement in the X-filling method using GA algorithms. The best values in Table 4 are close to the average values in Table 3. In addition, for some circuits, the random-fill derives better SDQLs than the GA-based method. It means that the simulation-based X-filling may have the limitation of the test quality improvement. In this case, more deterministic methods like timing-aware ATPG would be necessary to drastically improve SDQLs by X-filling.

Table 3 Average SDQLs in four generations.

circuit	GEN1	GEN5	GEN10	GEN20	GEN50	Lower bound
b15s	0.9653	0.9651	0.9650	0.9650	0.9649	0.9299
b17s	0.9607	0.9600	0.9596	0.9594	0.9591	0.9243
b20s	0.9768	0.9780	0.9777	0.9774	0.9769	0.9355
b21s	0.9682	0.9681	0.9680	0.9678	0.9674	0.9337
b22s	0.9719	0.9719	0.9718	0.9717	0.9723	0.9255
s35932	0.9709	0.9699	0.9700	0.9694	0.9687	0.9385
s38417	0.8611	0.8556	0.8488	0.8378	0.8317	0.7202
s38584	0.9826	0.9826	0.9823	0.9820	0.9817	0.8415

Table 4 Best SDQLs in four generations.

circuit	GEN1	GEN5	GEN10	GEN20	GEN50	Lower bound
b15s	0.9651	0.9646	0.9646	0.9646	0.9645	0.9299
b17s	0.9605	0.9587	0.9586	0.9584	0.9582	0.9243
b20s	0.9760	0.9776	0.9771	0.9768	0.9768	0.9355
b21s	0.9679	0.9676	0.9675	0.9673	0.9671	0.9337
b22s	0.9711	0.9708	0.9714	0.9712	0.9719	0.9255
s35932	0.9698	0.9683	0.9688	0.9687	0.9683	0.9385
s38417	0.8590	0.8467	0.8399	0.8331	0.8289	0.7202
s38584	0.9824	0.9823	0.9820	0.9818	0.9816	0.8415

7. Conclusions

This paper proposed a method to compute signal delay in fault simulation for test cubes and to find the range of SDQL of test patterns covered by the test cubes. By using the proposed method, we could derive the lower bound and the upper bound of delay test quality of test patterns covered by the test cube. In addition, this paper proposed an X-filling method to generate test patterns from the test cubes. By using the proposed method, we could generate test patterns

© 2010 Information Processing Society of Japan

with the better test quality than the test pattern assigned logic values to Xs at random. However, test quality of the test patterns obtained by the proposed method was still far from the high test quality for the test cubes. In the future work we will investigate a more accurate procedure to find a sensitizable path with maximum delay for each fault. Also we need to develop more effective X-filling methods to derive high quality test patterns from test cubes.

References

1) Cheng, K.-T., Dey, S., Rodgers, M. and Roy, K.: Test Challenges for Deep Sub-Micron Technologies, *Design Automation Conf.*, pp.142–149 (June 2000).
2) Waicukauski, J.A., Lindbloom, E., Rosen, B.K. and Iyengar, V.S.: Transition fault simulation, *IEEE Design & Test of Computers*, pp.32–38 (April 1987).
3) Sato, Y., Hamada, S., Maeda, T., Takatori, A. and Kajihara, S.: Evaluation of the statistical delay quality model, *ASP-DAC.*, pp.305–310 (2005).
4) Sato, Y., Hamada, S., Maeda, T., Takatori, A., Nozuyama, Y. and Kajihara, S.: Invisible delay quality – SDQM model lights up what could not be seen, *Int. Test Conf.*, 47.1 (Nov. 2005).
5) Hamada, S., Maeda, T., Takatori, A., Noduyama, Y. and Sato, Y.: Recognition of sensitized longest paths in transition delay test, *Int. Test Conf.*, Paper 11.1 (Oct. 2006).
6) Rajski, J., Tyszer, J., Kassab, M., Mukherjee, N., Thompson, R. and Tsai, K.H.: Embedded deterministic test for low cost manufacturing test, *Int. Test Conf.*, pp.301–310 (2002).
7) Koenemann, B., Barnhart, C., Keller, B., Snethen, T., Farnsworth, O. and Wheater, D.: A smart BIST variant guaranteed encoding, *Asian Test Sympo.*, pp.325–330 (2001).
8) Mitra, S. and Kim, K.S.: X-COMPACT An efficient rsponse compaction technique for test cost reduction, *Int. Test Conf.*, pp. 311–320 (2002).
9) Sankaralingam, R., Oruganti, R.R. and Touba, N.A.: Static Compaction Techniques to Control Scan Vector Power Dissipation, *VLSI Test Sympo.*, pp.35–40 (2000).
10) Wen, X., Miyase, K., Suzuki, T., Kajihara, S., Ohsumi, Y., Saluja, K.K.: Critical-Path-Aware X-Filling for Effective IR-Drop Reduction in At-Speed Scan Testing, *Design Automation Conf.*, pp.527–532 (June 2007).
11) Miyase, K., Terashima, K., Kajihara, S., Wen, X. and Reddy, S.M.: On improving defect coverage of stuck-at fault tests, *Asian Test Symp.*, pp.216–221 (2005).
12) Savir, J.: On broad-side delay testing, *VLSI Test Sympo.*, pp.284–290 (1994).
13) 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, *Int'l Test Conf.*, Paper 2.1 (Oct. 2006).
14) Miyase, K. and Kajihara, S.: XID: Don't Care Identification of Test Patterns for Combinational Circuits, *IEEE Trans. Computer-Aided Design of ICs & Systems*, Vol.23, No.2, pp.321–326 (2004).

(Received December 1, 2009)
(Revised February 26, 2010)
(Accepted April 14, 2010)
(Released August 16, 2010)

(Recommended by Associate Editor: *Takashi Sato*)

Shinji Oku was born in 1986. He received his B.E. and M.E. degrees from Kyushu Institute of Technology, in 2008 and 2010, respectively. His research interest includes delay testing for logic circuits.

Seiji Kajihara was born in 1965. He received his B.S. and M.S. degrees from Hiroshima University, Japan, and 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. Dr. Kajihara is a member of IEEE, IEICE, and IPSJ. He serves on the editorial board of the Journal of Electronic Testing: Theory and Applications.

© 2010 Information Processing Society of Japan

Yasuo Sato was born in 1953. He received his B.S. degree in 1976 and M.S. degree in 1978 in mathematics from the University of Tokyo. He received his Ph.D. in 2005 in engineering from Tokyo Metropolitan University. He joined Hitachi, Ltd. in 1978, and began working in computer-aided design. He had been a senior manager of Test Methodology Group in Semiconductor Technology Academic Research Center (STARC) from 2003 to 2005. He had been a Chief Engineer in Hitachi Micro Device Division from 2006 to 2008. He joined Kyusyu Institute of Technology, Iizuka, Japan, in 2009, where he is currently a Professor. His research interests include reliability-aware testing, design for testability, delay testing, defect-based testing, and fault diagnosis. He is a member of IEEE and IEICE.

Kohei Miyase was born in 1976. He received his Ph.D. degree in Computer Science and Systems Engineering from Kyushu Institute of Technology, Japan 2005. From 2005 to 2007, he was a Researcher in Innovation Plaza Fukuoka, Japan Science and Technology Agency, Japan. In 2007, he joined the Department of Computer Science and Electronics of Kyushu Institute of Technology, Japan, where he is an Assistant Professor currently. His research interests include test compression, design for testability, low power test, and fault diagnosis. He is a member of IEEE, IEICE, and IPSJ.

Xiaoqing Wen was born in 1965. He received his B.E. degree from Tsinghua University, Beijing, China in 1986, M.E. degree from Hiroshima University, Hiroshima, Japan in 1990, and 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 from October 1995 to March 1996. He joined SynTest Technologies, Inc., Sunnyvale, CA, in 1998 and served as its chief technology officer until 2003. In 2004, he joined the Kyushu Institute of Technology, Iizuka, Japan, where he is currently a Professor. His research interests include design, test, and diagnosis of integrated circuits. He currently holds 15 U.S. Patents, 2 Japan Patents, and 22 pending U.S. and Japan Patents in logic built-in self-test, test compression, and low power test generation. He received the 2008 IEICE-ISS Best Paper Award. He is a senior member of IEEE, a member of IEICE, and a member of REAJ.

© 2010 Information Processing Society of Japan