IEICE TRANS. INF. & SYST., VOL.E93–D, NO.1 JANUARY 2010

PAPER *Special Section on Test, Diagnosis and Verification of SOCs*

A Fault Dependent Test Generation Method for State-Observable FSMs to Increase Defect Coverage under the Test Length Constraint*

Ryoichi INOUE[†], ***Nonmember***, **Toshinori HOSOKAWA**[††], ***Member***, ***and*** **Hideo FUJIWARA**[†††], ***Fellow***

SUMMARY Since scan testing is not based on the function of the circuit, but rather the structure, it is considered to be both a form of over testing and under testing. Moreover, it is important to test VLSIs using the given function. Since the functional specifications are described explicitly in the FSMs, high test quality is expected by performing logical fault testing and timing fault testing. This paper proposes a fault-dependent test generation method to detect specified fault models completely and to increase defect coverage as much as possible under the test length constraint. We present experimental results for MCNC'91 benchmark circuits to evaluate bridging fault coverage, transition fault coverage, and statistical delay quality level and to show the effectiveness of the proposed test generation method compared with a stuck-at fault-dependent test generation method.

key words: *state-observable FSMs, logical fault testing, timing fault testing, fault sensitization coverage, n-detection*

1. Introduction

In recent years, very large scale integrated circuit (VLSI) testing has become increasingly important because of the rapidly increasing number of gates on VLSIs and the growing complexity of VLSIs due to advances in semiconductor technology. Currently, scan testing for the stuck-at fault model [1], [2] is one of the most popular test methods for VLSIs. However, it has been reported that scan testing for the stuck-at fault model may not detect defective VLSIs [3], and that delay testing and at-speed functional testing can effectively improve test quality [4]. Scan testing is based on the structure of the circuit rather than its function and the test pattern can be generated with this method. During scan testing, the states of the circuits are turned into invalid states [5] by the shift operation during the testing in order to detect faults. Invalid states occur when test patterns contain values for the state register that cannot be stored as state transitions after the reset state. Due to this, scan testing is considered as form of over testing, hence, yield loss of VLSIs may occur. Moreover, this testing method detects faults through the process of shifting-in test vectors, operating on normal mode for the combinational circuit part, and shifting-out. Thus, faults are not detected by performing sequential operations of the circuits. With this, scan testing is also considered as a form of under testing. Therefore, the test quality deteriorates and outflow of defective VLSIs into the market may occur.

VLSI design methodologies using hardware description languages have been adopted to reduce VLSI design time. VLSIs are designed at the Register Transfer Level (RTL), and RTL circuits consist of a data path part and a controller part. The data path contains hardware element (e.g., registers, multiplexers, and operational modules) and signal lines. The controller, on the other hand, is represented by a finite state machine (FSM). The controller and the data path are interconnected by internal signals: control signals and status signals. A non-scan-based Design For Testability (DFT) method of the data path part is proposed in [6], whereas a non-scan-based DFT method for the controller part is proposed in [5]. At-speed testing is possible and test patterns for a stuck-at fault model are completely generated using non-scan-based DFT methods. In [5], [6], both control signals from the controller and status signals from the data path were assumed to be directly controllable from primary inputs and observable at primary outputs. As mentioned above, if at-speed functional testing and/or delay testing are applied to VLSIs with a non-scan-based DFT, the test quality can be further improved. As for the FSM, which is the controller part of an RTL circuit, the circuit specification is described explicitly. Thus, high test quality is expected by performing a logical fault testing and a timing fault testing under the constraints of the circuit specifications.

In consideration of these tests, a fault-independent one-pattern test generation method and a fault-independent two-pattern test generation method that enable complete logical fault testing and timing fault testing have been proposed [7], [8]. However, when the number of state transitions increases, the test length drastically increases. It is necessary to detect a specified fault model (e.g. stuck-at fault) completely and to detect main fault models such as bridging fault, transition fault, and path delay fault as much as possible for state-observable FSMs. An n-detection test generation method (FSOD) used to increase the fault sensitization coverage [9] comparatively detected many bridging faults and transition faults.

Manuscript received February 6, 2009.
Manuscript revised June 1, 2009.

[†]The author is with the Graduate School of Industrial Technology, Nihon University, Narashino-shi, 275–8575 Japan.
[††]The author is with the College of Industrial Technology, Nihon University, Narashino-shi, 275–8575 Japan.
[†††]The author is with the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma-shi, 630–0192 Japan.

*A previous version of this paper has been published at Asian Test Symposium 2008.

DOI: 10.1587/transinf.E93.D.24

Copyright © 2010 The Institute of Electronics, Information and Communication Engineers

This paper proposes a fault-dependent test generation method to detect specified fault models completely and to increase defect coverage as much as possible under test length constraint. This paper also proposes weighted state transition coverage as a measure of test quality.

This paper is organized as follows. In Sect. 2, the definition of state-observable FSMs is given. In Sect. 3, the detection conditions of main fault models and an n-detection test generation method to increase defect coverage are described. In Sect. 4, a fault-dependent test generation method for state-observable FSMs is proposed, and experimental results for MCNC'91 FSM benchmarks [10] with many state transitions are discussed in Sect. 5. Finally, Sect. 6 concludes the paper and discusses future research possibilities.

2. State-Observable FSMs

Definition 1 (State-observable FSMs):
When an initial state can be identified by observing an output sequence without being dependent on the input sequence, the FSM is said to be ***state-observable***. More specifically, when an initial state can be identified by observing an output sequence of length *k*, the FSM is said to be *k* state-observable.

Figure 1 shows an example of an FSM. In this figure, ST0 through ST5 and T0 through T11 show the states and the input values, respectively, of the state transitions (the value of each primary input {0, 1, X}, where X denotes don't care). DFT transforms an FSM to a one-state observable FSM by making the outputs of the status registers in the FSM observable. In this paper, a one-state observable FSM is hereinafter referred to simply as a state observable FSM. A synchronous sequential circuit is synthesized from the FSM by logic synthesis. Figure 2 shows the logic circuit model that corresponds to the FSM after logic synthesis. Since the pseudo primary inputs (PPI), which are the outputs of the status registers, are observable in this figure, the PPIs connect with the primary output. Thus, multiplexers are added on the PPI and are connected to the primary outputs of the data path in order to reduce the overhead of primary output pins [11]. Here, PI, PO, SR, PPI, PPO, and R denote the primary inputs, primary outputs, status registers, pseudo primary inputs (outputs of the status registers), pseudo primary outputs (inputs of the status registers), and a reset input, respectively.

In testing state-observable FSMs, the PI value is applied to a state-observable FSM, the resulting PO values are observed, the state is then transferred from the current state to the next state, and the resulting PPI values are observed. A series of these procedures is referred to as a test for state-observable FSMs.

Example 1: In Fig. 1, T0 is applied to state ST0 on the state-observable FSM and the state is transferred from ST0 to ST1. T1 is then applied, and the state is transferred from ST1 to ST2. Next, the test for the state-observable FSM is explained in detail. R is activated and the values of the status registers are initialized to ST0 in the first cycle. In

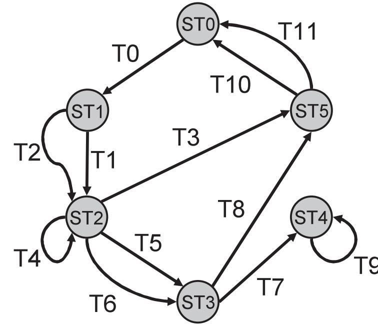

Fig. 1 Example of an FSM. (six states)

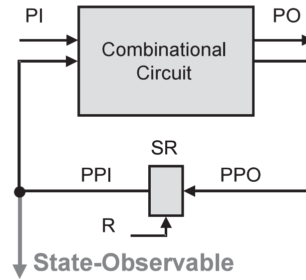

Fig. 2 Logic model for a state-observable FSM.

the second cycle, T0 is applied and the values of the POs for (PI, PPI) = (T0, ST0) are observed just before the rising edge of the clock. Here, (PI, PPI) indicates that the value of PI is applied to the PPI value (state) for the state-observable FSM. Moreover, the PPI value is observed after the rising edge of the clock. Thus, it is verified that the state is successfully transferred from ST0 to ST1. In the third cycle, T1 is applied and the PO values for (PI, PPI) = (T1, ST1), which are observed just before the rising edge of the clock. The resulting PPI value is observed after the rising edge of the clock. Thus, it is verified that the state is successfully transferred from ST1 to ST2.

The FSM has both a completely specified FSM [14], in which the next state and the output are specified for all of the inputs of each state, and an incompletely specified FSM [11], in which the next state and the output are not specified for all of the inputs of each state. In this paper, state transitions in the incompletely specified FSMs that are not specified are assumed to be the same as either of the state transitions that are specified.

3. Detection Conditions for Each Fault Model

First, an n-detection test generation method to increase fault sensitization coverage [9] is explained. Next, detection conditions for the main fault models such as bridging faults [2], and transition faults [4] are described.

3.1 An n-Detection Test Generation Method to Increase Fault Sensitization Coverage

Definition 2 (Fault Sensitization Coverage):
Fault sensitization coverage for fault f is defined as the ratio of the number of signal lines sensitized by test set T to the number of all signal lines that are reachable from f. Here, sensitized signal lines for f are lines on the fault propagation path at the time that f is detected. Fault sensitization coverage for the whole circuit is expressed by the average value of fault sensitization coverage for all faults. The equations to solve fault sensitization coverage for f and the whole circuit are expressed as follows.

- *senf*: Fault sensitization coverage for fault f

$$senf = \frac{\text{Number of sensitized signal lines}}{\left\{\begin{array}{l}\text{Number of the signal lines}\\ \text{which are reachable from } f\end{array}\right\}} \times 100 \qquad (1)$$

- *SEN*: Fault sensitization coverage for the whole circuit

$$SEN = \frac{\sum senf}{\text{Number of faults}} \qquad (2)$$

An n-detection test generation method to increase fault sensitization coverage, FSOD, can be used for stuck-at faults to increase fault sensitization coverage based on the following strategies.

(1) For each fault, FSOD generates n test patterns that sensitize different fault propagation paths and detect faults.
(2) FSOD selects a D-frontier [1], [2] to sensitize long fault propagation path segments.

3.2 Detection of Bridging Faults

A bridging fault is a fault model that expresses a short between signal lines. Bridging faults are classified into AND type and OR type based on failure behavior. It is necessary to generate a test pattern that detects a stuck-at 0 (1) fault for one signal line and sets 0 (1) to the other signal line in order to detect an AND (OR) type bridging fault. In this paper, U model [12] is used. Both an AND type and an OR type must be detected for the detection of a U model of the bridging fault. A bridging fault may be detectable only when it sensitizes a specific path. Therefore, if test patterns are generated so that many paths are sensitized as much as possible, bridging fault coverage is increased. Since FSOD sensitizes many fault propagation paths by increasing fault sensitization coverage, it is considered that the generated test patterns achieve high bridging fault coverage.

3.3 Detection of Transition Faults

A transition fault model assumes that a delay fault affects only one signal line in the circuit. There are two transition faults associated with each signal line: a slow-to-rise fault and a slow-to-fall fault. It is assumed that in the fault-free circuit each signal line has some nominal delay. Delay faults result in an increase of this delay. Under the transition fault model, the extra delay caused by the delay fault is assumed to be large enough to prevent the transition from reaching any primary output at the time of observation. In other words, the transition fault can be observed independent of whether the transition propagates through a long or short path to any primary output. To detect a transition fault, it is necessary to apply a test pattern pair, V = (v1, v2). For testing a slow-to-rise fault (a slow-to-fall fault), the first pattern, v1, initializes the fault site to 0 (1), and the second pattern, v2, is a test pattern for stuck-at-0 (1) fault at the fault site. It is considered that FSOD can detect a small size of delay fault because it sensitizes a long fault propagation path to increase fault sensitization coverage. Because FSOD also generates n-detection test patterns, transition probability of fault sites between the first pattern and the second pattern is high. Then, the probability of transition fault detection is considered to increase.

4. Fault Dependent Test Generation Method for State-Observable FSMs

This method generates a test sequence by generating an FSM test generation graph from state-observable FSMs and searching for a path. We propose weighted one-state transition coverage and weighted two-state transition coverage as measures of test quality for logical fault testing and timing fault testing, respectively, for the generated test sequence.

4.1 FSM Test Generation Graph

Definition 3 (FSM test generation graph):
Given an FSM M and a set T of test patterns generated by FSOD, an *FSM test generation graph* is defined as a directed graph $G = (V, E, s, d, t, wt_v, wt_e)$ that has the following properties.

1. Each vertex v in V corresponds to a state transition of M.
2. s: $V \rightarrow$ A defines a source state of each state transition for M corresponding to a vertex v, where A denotes a set of m-bit state assignment code words and m is the number of state assignment variables or the size of the state register;
3. d: $V \rightarrow$ A defines a destination state of each state transition for M corresponding to a vertex v;
4. t: $V \rightarrow$ B defines an input value of each state transition for M corresponding to a vertex v, where B denotes a set of n-bit primary input vectors and n is the number of primary inputs;
5. There is an edge (u, v) in E if $d(u) = s(v)$;
6. wt_v: $V \rightarrow \{0, 1\}$ where $wt_v(v) = 1$ if $(s(v), t(v))$ is equivalent to a test pattern generated by FSOD and $wt_v(v) = 0$ otherwise;
7. wt_e: $E \rightarrow Z$ where Z is the set of all integers, $wt_e((\text{v1}, \text{v2}))$ is the Hamming distance between

$(s(v1), t(v1))$ and $(s(v2), t(v2))$ if $wt_v(v2)$ is 1, and $wt_e((v1, v2))$ is 0 if $wt_v(v2)$ is 0.

The quality of logical fault testing is considered to increase by executing test patterns generated by FSOD. Therefore, the weight wt_v is assigned to the vertex. The transition between the first pattern and the second pattern must occur at a fault site in order to increase the quality of transition fault testing. It has been reported that when the number of transitions at primary inputs is large, the number of transition at the internal signal lines is also large [13]. Thus, when the Hamming distance between the first pattern and the second pattern is large, the probability that the transition will occur is high. Therefore, the probability for the detection of transition faults becomes high.

Example 2: Figure 3 shows the state-observable FSM. Figure 4 shows the FSM test generation graph of Fig. 3. The two test patterns, $(\mathrm{PPI}_1, \mathrm{PPI}_2, \mathrm{PI}) = (1, 0, 0)$ and $(\mathrm{PPI}_1, \mathrm{PPI}_2, \mathrm{PI}) = (1, 0, 1)$, are generated for the combinational circuit after logic synthesis using the FSOD. In Fig. 4, a state assignment code of the source state (label s), a state assignment code of the destination state (label d), and the input value in each vertex corresponding to a state transition are assigned. Moreover, the weight wt_v is assigned to each vertex and the weight wt_e is assigned to each edge. In Fig. 4, triangles indicate the values of wt_v and the squares indicate the values of wt_e. Each vertex is expressed as (s, d, t). The edge $((00, 01, 0), (01, 10, 1))$ means that the state 00 transfers to the state 01 with input 0 and the state 01 transfers to the state 10 with input 1. Since the test pattern generated by FSOD, $(\mathrm{PPI}_1, \mathrm{PPI}_2, \mathrm{PI}) = (1, 0, 0)$, corresponds to the vertex $(10, 10, 0)$, the wt_v is 1. Similarly, since the test pattern generated by FSOD, $(\mathrm{PPI}_1, \mathrm{PPI}_2, \mathrm{PI}) = (1, 0, 1)$, corresponds to the vertex $(10, 00, 1)$, the wt_v is 1. In other vertices, wt_vs are 0. The weight of the edge, $wt_e((01, 10, 1), (10, 10, 0))$ is assigned 3 which is the Hamming distance between $\{s, t\} = \{10, 0\}$ of $(10, 10, 0)$ and $\{s, t\} = \{01, 1\}$ of $(01, 10, 1)$. The weight of the edge, $wt_e((00, 10, 1), (10, 10, 0))$ is assigned 2 which is the Hamming distance between $\{s, t\} = \{10, 0\}$ of $(10, 10, 0)$ and $\{s, t\} = \{00, 1\}$ of $(00, 10, 1)$. The weight of the edge, $wt_e((10, 10, 0), (10, 10, 0))$ is assigned 0 which is the Hamming distance between $\{s, t\} = \{10, 0\}$ of $(10, 10, 0)$ and $\{s, t\} = \{10, 0\}$ of $(10, 10, 0)$. Likewise, The weight of the edge, $wt_e((00, 10, 1), (10, 00, 1))$ is assigned 1 which is the Hamming distance between $\{s, t\} = \{10, 1\}$ of $(10, 00, 1)$ and $\{s, t\} = \{00, 1\}$ of $(00, 10, 1)$. The weight of the edge, $wt_e((01, 10, 1), (10, 00, 1))$ is assigned 2 which is the Hamming distance between $\{s, t\} = \{10, 1\}$ of $(10, 00, 1)$ and $\{s, t\} = \{01, 1\}$ of $(01, 10, 1)$. The weight of the edge, $wt_e((10, 10, 0), (10, 00, 1))$ is assigned 1 which is the Hamming distance between $\{s, t\} = \{10, 1\}$ of $(10, 00, 1)$ and $\{s, t\} = \{10, 0\}$ of $(10, 10, 0)$. In the other edges, wt_es are 0.

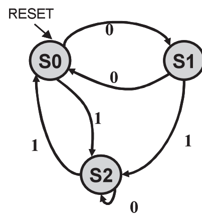

Fig. 3 Example of an FSM. (Three States)

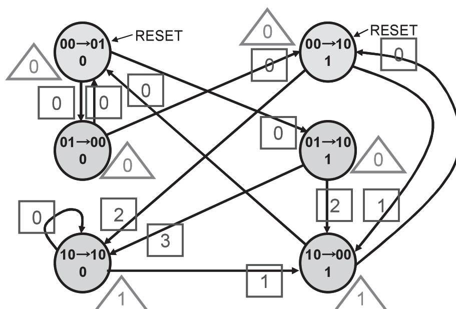

Fig. 4 FSM test generation graph.

4.2 Weighted State Transition Coverage

The two types of weighted state transition coverage are defined as follows.

Definition 4 (Weighted one-state transition coverage): The *Weighted one-state transition coverage* is expressed in Eq. (3) and is used as the measure of the test quality for logical fault testing.

$$\text{Weighted one-state transition coverage} = \frac{\text{Sum of weights of vertices covered by test sequence}}{\text{Sum of weights for all vertices}} \times 100(\%) \tag{3}$$

Definition 5 (Weighted two-state transition coverage): The *Weighted two-state transition coverage* is expressed in Eq. (4) and is used as the measure of the test quality for timing fault testing.

$$\text{Weighted two-state transition coverage} = \frac{\sum_v \max\left\{\begin{array}{l}\text{The weight of input edges for each vertex } v\\ \text{which covered by test sequence}\end{array}\right\}}{\sum_v \max\left\{\text{The weights of input edges for each vertex } v\right\}} \times 100(\%) \tag{4}$$

Weighted one-state transition coverage is calculated using the weights assigned to vertices while weighted two-state transition coverage is obtained using the weights assigned to edges in an FSM test generation graph. Since an n-detection test generation method (FSOD) has been reported to detect many bridging faults and transition faults [9], it can be said that the two types of weighted state transition coverage increase as the ratio of test patterns of FSOD covered by the test sequence generated for FSMs increases.

The following problem is formulated for the test generation for state-observable FSMs under test length constraint.

Problem Formulation:
Input:

- a state-observable FSM.
- a test set that can detect all detectable stuck-at faults on valid states.
- a test set generated by the FSOD.

Constraint: test length
Output: a test sequence for the state-observable FSM such that all detectable stuck-at faults on valid states are detected.
Optimization:

(1) maximization of weighted one-state transition coverage
(2) maximization of weighted two-state transition coverage

The valid states are assigned to PPI values as constrains. FSOD is performed for the combinational circuit part to generate the test patterns. Then, an FSM test generation graph is generated, and the given stuck-at fault test pattern set are assigned to the corresponding vertices on the FSM test generation graph. Next, the test patterns generated by the FSOD are assigned to the corresponding vertices on the FSM test generation graph. Finally, paths are searched from the FSM test generation graph such that all of the edges on which stuck-at fault tests are assigned are traversed at least once. The traversal passes along vertices such that as many test patterns generated by the FSOD are assigned as possible, so as to increase the weighted one-state transition coverage. The traversal also passes along the edges with the largest possible weight, in order to increase the weighted two-state transition coverage. If all stuck-at fault test patterns do not cover vertices under test constraint, the problem is not given any solution since it is not the focus of this work.

4.3 Strategy of Test Generation

The procedure of test generation is as follows. Reset states are first set to current states.

STEP1
From the current state of an FSM test generation graph, a k-state transition search is done and all of the paths are extracted. k is a parameter with a positive integer value.

STEP2
One path is selected by using the heuristic algorithm applied to all of the paths, and it is added to the test sequence.

STEP3
To reduce test length, a transition to selected path is limited as it is not needed.

STEP4
If the test sequence does not abide with the given test length constraint, start again from STEP1. The final state of the selected path is set to its current state.

The heuristic algorithm is explained as follows. In the early stage of test generation, the probability of having uncovered vertices at which stuck-at test patterns are assigned is high in k state transition paths because the number of uncovered vertices is large. In this case, the state transition path is selected by the following priority of heuristics: 3, 4, 5, 1, and 2. If multiple paths are selected by the highest priority heuristic, the next priority heuristic is applied to these selected paths. The same process continues until only one path is selected. If there are multiple paths that satisfy the last order heuristic, a path is arbitrary selected from the multiple paths. When the number of uncovered vertices, at which stuck-at test patterns are assigned, is small, the probability that the patterns appear in the k state transition path is low. If the number of uncovered vertices, at which stuck-at test patterns are assigned, is 0 repeatedly m times, the state transitions path is selected by the priority of heuristics: 1, 2, 3, 4, and 5. The same process described above is done until only one path is selected. Here, k and m are parameters with positive integer values.

Heuristic 1
For complete stuck-at fault detection, the algorithm preferentially selects a path that includes many uncovered vertices where stuck-at fault test patterns are assigned.

Heuristic 2
To reduce test length, the algorithm preferentially selects a path such that the distance from the current state to uncovered vertices, where stuck-at test patterns are assigned, is short. The algorithm can transfer at next k-state transition search efficiently to uncovered vertices where stuck-at fault test patterns are assigned.

Heuristic 3
To increase the quality of logical fault testing, the algorithm preferentially selects a path such that the total sum of wt_v is large. As a result, the weighted one-state transition coverage becomes high.

Heuristic 4
To reduce test length, the algorithm preferentially selects a path such that the distance from the current state to uncovered vertices, where test patterns generated by FSOD are assigned, is short. The algorithm can transfer at next k-state transition search efficiently to uncovered vertices where test patterns generated by FSOD are assigned.

Heuristic 5
In order to increase the quality of timing fault testing, the

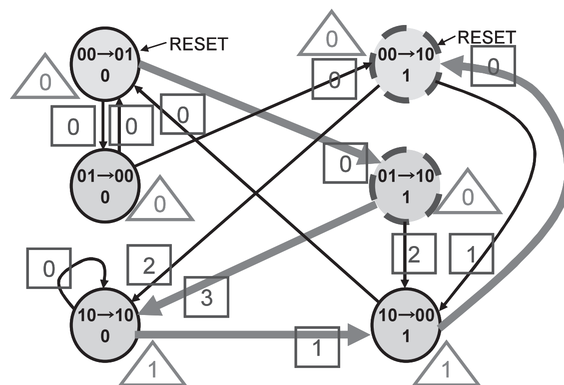

Fig. 5 Example of test sequence.

algorithm preferentially selects a path such that the total sum of wt_e is large. As a result, the weighted two-state transition coverage becomes high.

Example 3: Given the stuck-at test patterns, $(PPI_1, PPI_2, PI) = (0, 0, 1)$, and $(PPI_1, PPI_2, PI) = (0, 1, 1)$, Fig. 5 shows the FSM test generation graph of Fig. 3. FSOD generates the test patterns, $(PPI_1, PPI_2, PI) = (1, 0, 0)$, and $(PPI_1, PPI_2, PI) = (1, 0, 1)$. In Fig. 5, the vertices indicated by dashed lines are vertices where stuck-at test patterns are assigned. When the test sequence (0, 1, 0, 1) is generated from the reset state, the weighted one-state transition coverage is 100% (2/2) whereas the weighted two-state transition coverage is 80% ($(3 + 1)/(3 + 2) = 4/5$).

5. Experimental Results

The test generation method was implemented and applied to MCNC'91 benchmark circuits [10]. The characteristics of MCNC'91 benchmark circuits are shown in Table 1. In this table, Circuit, #Node, #PI, #PO, #Reg, and #Edge denote the circuit name of the FSM, the number of states, the number of primary inputs, the number of primary outputs, the number of status registers and the number of state transitions, respectively. In these experiments, the FSMs were made state observable through DFT, and three test generations were performed for state-observable FSMs.

Table 2 shows the experimental results of fault-independent one-pattern test generation method (1a) [7], [8] and the fault-dependent one-pattern test generation method (1b) [7], [8].

Table 3 presents the experimental results of the proposed method when the value of m was set to three, and the test length constraint was set to the same test length as 1b. This algorithm detects stuck-at faults completely. The value m is a parameter for switching timing in the algorithm shown in the heuristic priority rules.

Table 4 shows the experimental results of the proposed method when test length constraint was set to 300, 500, 800, and 1300. The circuits indicated by the "*" symbol in the table are the ones in which stuck-at fault could not be detected completely by the test lengths of 300, 500, 800 and 1300. The value k was set to 3 in all experiments. Moreover, the value n of n-detection for FSOD was set to 5. We also changed the value of n to see the effect and the trend was the same as the case of $n = 5$ in the experiments.

Table 5 shows the experimental results of the proposed method when specified fault models are set to stuck-at fault and transition fault, and test length constraint was set to 3000. The circuits indicated by the "*" symbol in the table are ones for which stuck-at fault and transition fault could

Table 1 FSM benchmark characteristics.

Circuit	#Node	#PI	#PO	#Reg	#Edge
cse	16	7	7	4	2048
ex1	20	9	19	5	10240
keyb	19	7	2	5	2432
kirkman	16	12	6	4	65536
planet	48	7	19	6	6144
pma	24	8	8	5	6144
s1	20	8	6	5	5120
s1488	48	8	19	6	12288
s1494	48	8	19	6	1288
s208	18	8	2	5	4608
s298	218	3	6	8	1744
s386	13	7	7	4	1664
s420	18	8	2	5	4608
s820	25	18	19	5	6553600
s832	25	18	19	5	6553600
sand	32	11	9	5	65536
styr	30	9	10	5	15360
tma	20	7	6	5	2560

Table 3 Experimental results. (With 1b test length constraint)

Circuit	Proposed methond (stuck-at-fault)							
	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQM (ppm)	CPUtime (sec)
cse	100.00	99.87	91.98	169	26.02	28.52	147	1.13
ex1	100.00	99.46	92.78	145	25.42	22.23	206	0.62
keyb	100.00	98.58	89.68	220	14.63	26.96	189	26.24
kirkman	100.00	99.47	93.84	159	32.70	28.59	101	3016.46
planet	100.00	99.83	93.83	286	38.13	36.05	294	0.14
pma	100.00	98.78	88.55	201	22.99	22.21	302	0.10
s1	100.00	99.55	91.26	154	12.98	14.34	272	0.34
s1488	100.00	99.73	93.99	632	31.94	30.23	366	6.60
s1494	100.00	99.80	93.20	577	30.59	27.34	410	10.97
s208	100.00	99.72	94.82	117	39.18	33.45	24	17.86
s298	100.00	99.80	94.47	1330	38.64	45.33	821	503.54
s386	100.00	99.74	93.12	87	27.01	29.19	86	0.19
s420	100.00	99.60	95.79	120	40.00	32.28	18	5.17
s820	100.00	99.64	89.60	468	22.08	20.51	297	14.18
s832	100.00	99.65	89.36	444	22.97	21.75	331	28.90
sand	100.00	99.52	90.58	221	17.57	16.79	376	10.16
styr	100.00	99.63	90.02	210	22.45	21.11	356	1.88
tma	100.00	99.32	87.78	141	28.32	27.39	211	0.03
average	100.00	99.54	91.93	315.61	27.42	26.90	267.15	202.47

Table 2 Experimental results for logical fault testing.

Circuit	1a								1b							
	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQM (ppm)	CPUtime (sec)	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQM (ppm)	CPUtime (sec)
cse	100.00	100.00	93.95	7705	100.00	75.13	141	9.60	100.00	99.76	89.63	169	26.77	18.63	194	0.17
ex1	100.00	100.00	97.22	30096	100.00	72.82	107	30.62	100.00	99.36	89.57	145	2.71	2.24	270	0.11
keyb	100.00	100.00	97.59	9131	100.00	73.52	73	129.70	100.00	98.43	81.10	220	24.39	15.28	343	2.22
kirkman	ND	ND	ND	ND	ND	ND	ND	ND	100.00	98.91	86.56	159	2.66	2.69	223	45.01
planet	100.00	100.00	97.58	16315	100.00	73.31	146	1.94	100.00	99.71	87.66	286	5.28	3.79	577	0.03
pma	100.00	100.00	88.47	13338	100.00	91.76	339	1.82	100.00	99.26	89.24	201	8.77	7.12	274	0.03
s1	100.00	100.00	90.81	9012	100.00	73.94	297	3.17	100.00	99.42	75.46	154	4.42	3.72	730	0.08
s1488	100.00	100.00	96.04	89797	100.00	66.79	270	162.46	100.00	99.74	81.34	632	9.36	4.19	1055	1.10
s1494	100.00	100.00	97.54	91753	100.00	63.99	179	284.51	100.00	99.77	83.53	577	7.06	3.94	901	1.54
s208	100.00	100.00	100.00	32414	100.00	74.51	2	64.64	100.00	99.34	87.56	117	9.28	5.07	69	0.24
s298	100.00	100.00	85.81	11143	100.00	37.43	1861	66.24	100.00	99.78	85.83	1330	51.80	27.04	1914	6.81
s386	100.00	100.00	92.73	6066	100.00	80.06	114	4.11	100.00	99.64	83.75	87	15.52	11.68	197	0.04
s420	100.00	100.00	97.89	32351	100.00	31.70	13	43.37	100.00	99.20	86.84	120	3.16	2.92	61	0.19
s820	ND	ND	ND	ND	ND	ND	ND	ND	100.00	99.68	76.88	468	0.19	0.18	642	2.97
s832	ND	ND	ND	ND	ND	ND	ND	ND	100.00	99.47	77.57	444	0.00	0.00	691	3.97
sand	100.00	100.00	97.91	89955	100.00	79.65	146	384.44	100.00	99.42	86.40	221	1.05	0.94	518	1.28
styr	100.00	100.00	92.94	46890	100.00	71.51	328	80.13	100.00	99.58	58.12	210	6.58	4.35	1457	0.26
tma	100.00	100.00	91.94	4242	100.00	84.53	153	0.28	100.00	99.10	62.29	141	14.16	12.56	625	0.01
average	100.00	100.00	94.56	32680.53	100.00	70.04	277.75	84.47	100.00	99.42	81.63	315.61	10.73	7.02	596.72	3.67

Table 4 Experimental results. (Test length constraint)

Circuit	Proposed methond (stuck-at-fault, TL=300)								Proposed methond (stuck-at-fault, TL=500)							
	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQL (ppm)	CPUtime (sec)	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQL (ppm)	CPUtime (sec)
cse	100.00	99.96	95.43	300	57.25	47.54	86	1.40	100.00	99.98	96.30	500	85.87	67.97	75	2.47
ex1	100.00	99.79	96.43	300	67.12	55.48	92	1.03	100.00	99.90	97.91	500	91.19	73.97	55	1.94
keyb	100.00	99.01	92.09	300	35.77	40.66	154	28.04	100.00	99.61	94.50	500	67.07	60.44	109	36.09
kirkman	100.00	99.63	96.23	300	66.92	56.46	62	3228.22	100.00	99.75	97.11	500	84.79	72.77	47	3787.88
planet	100.00	99.84	94.50	300	40.29	38.11	267	0.13	100.00	99.89	97.28	500	67.87	63.14	149	0.17
pma	100.00	99.31	91.65	300	41.00	38.60	219	0.12	100.00	99.83	93.98	500	74.88	66.41	154	0.18
s1	100.00	99.81	95.21	300	45.58	43.86	160	0.48	100.00	99.84	96.12	500	70.99	65.73	134	0.70
*s1488	100.00	99.74	94.39	633	31.94	30.23	357	6.55	100.00	99.74	94.39	633	31.94	30.23	357	6.55
*s1494	100.00	99.80	93.04	566	29.08	26.32	432	10.91	100.00	99.80	93.04	566	29.08	26.32	432	10.91
s208	100.00	99.81	95.85	300	67.01	59.28	19	34.69	100.00	99.91	97.41	500	86.60	78.43	11	70.91
*s298	100.00	99.79	94.61	1217	35.15	43.55	780	491.19	100.00	99.79	94.61	1217	35.15	43.55	780	491.19
s386	100.00	99.92	97.13	300	89.08	79.00	37	0.54	100.00	99.97	97.90	500	100.00	91.05	29	1.29
s420	100.00	99.70	97.89	300	69.47	58.36	16	9.95	100.00	99.80	98.95	500	87.37	77.43	11	20.86
*s820	100.00	99.64	89.38	439	16.88	15.45	306	13.95	100.00	99.65	89.88	500	25.60	23.90	294	14.23
*s832	100.00	99.62	89.00	415	17.62	16.60	342	28.64	100.00	99.71	89.79	500	29.31	27.82	316	31.27
sand	100.00	99.63	92.33	300	30.05	27.84	327	16.04	100.00	99.79	94.83	500	59.23	52.83	225	24.42
styr	100.00	99.74	91.26	300	38.32	31.98	308	2.24	100.00	99.79	93.56	500	63.04	52.15	234	3.10
tma	100.00	99.69	91.81	300	69.03	60.30	141	0.05	100.00	99.89	95.19	500	94.69	81.77	88	0.09
average	100.00	99.68	94.45	300.00	55.15	49.04	145.23	255.61	100.00	99.82	95.38	500.00	72.57	63.72	128.84	266.37

Circuit	Proposed methond (stuck-at-fault, TL=800)								Proposed methond (stuck-at-fault, TL=1300)							
	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQL (ppm)	CPUtime (sec)	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQL (ppm)	CPUtime (sec)
cse	100.00	99.98	96.79	800	100.00	80.01	58	6.97	100.00	99.99	97.16	1300	100.00	82.45	52	7.97
ex1	100.00	99.94	98.43	800	100.00	83.27	43	4.96	100.00	99.94	98.78	1300	100.00	85.99	34	5.90
keyb	100.00	99.82	94.77	800	87.80	77.36	107	81.30	100.00	99.87	95.04	1300	100.00	92.76	105	168.26
kirkman	100.00	99.89	98.24	800	99.24	84.89	28	6360.48	100.00	99.89	98.49	1300	100.00	88.36	23	6939.13
planet	100.00	99.91	98.56	800	83.21	76.98	87	0.32	100.00	99.95	98.97	1300	99.52	91.83	66	0.67
pma	100.00	99.87	96.73	800	92.65	81.21	92	0.31	100.00	99.93	98.45	1300	100.00	91.98	39	0.58
s1	100.00	99.90	97.87	800	97.24	87.18	73	1.48	100.00	99.90	98.33	1300	100.00	93.42	54	2.13
s1488	100.00	99.77	94.87	800	44.82	39.81	333	6.99	100.00	99.83	95.47	1300	56.69	50.36	301	10.17
s1494	100.00	99.84	94.44	800	45.88	40.07	343	12.84	100.00	99.89	94.97	1300	57.65	51.01	302	20.13
s208	100.00	100.00	96.37	800	100.00	91.81	11	116.64	100.00	100.00	97.41	1300	100.00	91.81	11	127.22
*s298	100.00	99.79	94.61	1217	35.15	43.55	780	491.19	100.00	99.80	94.47	1300	37.92	44.90	813	506.98
s386	100.00	99.97	90.29	800	100.00	94.01	22	1.59	100.00	99.97	98.28	1300	100.00	94.77	22	2.01
s420	100.00	100.00	98.28	800	100.00	91.11	11	33.96	100.00	100.00	98.95	1300	100.00	91.11	11	37.10
s820	100.00	99.82	98.95	800	48.79	45.31	216	18.17	100.00	99.90	95.95	1300	73.65	67.85	131	30.16
s832	100.00	99.85	92.70	800	50.50	46.73	279	40.74	100.00	99.92	94.82	1300	75.25	68.53	175	70.13
sand	100.00	99.91	98.02	800	87.52	76.87	104	37.87	100.00	99.92	98.66	1300	100.00	89.34	70	61.11
styr	100.00	99.89	94.73	800	90.93	73.52	191	5.29	100.00	99.90	95.97	1300	100.00	85.15	149	13.12
tma	100.00	99.93	96.88	800	100.00	90.49	54	0.15	100.00	99.93	97.01	1300	100.00	93.20	50	0.21
average	100.00	99.90	96.29	800.00	84.03	74.15	120.78	395.89	100.00	99.92	97.07	1300.00	88.93	80.82	133.88	444.61

Table 5 Experimental results. (Two specified fault model)

Circuit	Proposed methond (transition fault & stuck-at-fault, TL=3000)							
	SFC(%)	BFC(%)	TFC(%)	TL	W1STC (%)	W2STC (%)	SDQL (ppm)	CPUtime (sec)
cse	100.00	100.00	100.00	3000	100.00	84.45	4	17.70
ex1	100.00	99.98	100.00	3000	100.00	87.85	0	14.05
keyb	100.00	99.95	100.00	3000	100.00	94.95	15	379.10
kirkman	100.00	99.91	100.00	3000	100.00	90.11	0	30030.55
planet	100.00	99.96	100.00	3000	100.00	93.08	22	1.25
pma	100.00	99.96	100.00	3000	100.00	93.64	0	1.49
s1	100.00	99.98	100.00	3000	100.00	95.20	6	4.85
*s1488	100.00	99.95	100.00	3600	45.65	50.11	29	51.24
*s1494	100.00	99.97	100.00	3403	43.70	47.49	50	90.18
s208	100.00	100.00	100.00	3000	100.00	91.81	2	218.67
*s298	100.02	99.96	100.00	9775	61.15	71.57	60	3656.98
s386	100.00	99.97	100.00	3000	100.00	95.30	4	4.47
s420	100.00	100.00	100.00	3000	100.00	91.46	9	64.56
s820	100.00	99.92	100.00	3000	72.54	70.40	19	110.38
s832	100.00	99.99	100.00	3000	76.24	73.53	14	339.44
sand	100.00	99.97	100.00	3000	100.00	92.05	0	394.86
styr	100.00	99.97	100.00	3000	100.00	86.36	12	42.71
tma	100.00	99.97	100.00	3000	100.00	96.21	0	0.49
average	100.00	99.97	100.00	3000.00	100.00	92.09	3.77	2830.07

not be detected completely by the test lengths of 3000.

In Tables 2, 3, 4, and 5, Circuit, TL, and CPU time denote the circuit name of the FSM, the test length, and the time for the test generation, respectively. SFC, BFC, TFC, W1STC, W2STC, and SDQL denote the stuck-at fault coverage, the bridging fault coverage, the transition fault coverage, the weighted one-state transition coverage, the weighted two-state transition coverage, and the statistical delay quality level [14] that evaluated with statistical delay quality model, respectively. Each logical fault testing targets only faults that can be detected on valid states [7]. Each timing fault testing targets only faults that can be detected on the transition between valid states [7], [8]. We assumed that in non-feedback bridging faults between two signal lines in circuit, U model [12] is used as the condition for detection. Using this, the bridging fault is recognized only when both the AND-type bridging fault and the OR-type bridging fault between two signal lines are detected.

First, the experimental results of the proposed method are considered when the test length constraint was set to same test length as 1b. Stuck-at faults can be completely tested. The weighted one-state transition coverage increased by an average of 16.69%, and the weighted two-state transition coverage increased by an average of 19.89% with the same test length compared to the fault-dependent one-pattern test generation method for the stuck-at fault model. Bridging fault coverage increased by an average of 0.12%, transition fault coverage increased by an average of 10.30%, and SDQL decreased by an average of 329 ppm. In particular, for kirkman, the weighted one-state transition coverage increased by 30.04%, bridging fault coverage increased by 0.56%. For styr, the weighted two-state transition coverage increased by 16.76%, transition fault coverage increased by 31.90%, SDQL decreased by 1100 ppm, and the quality of the timing fault testing was improved. As observed, the proposed test generation method resulted to

an increase in the weighted state transition coverage and fault coverage for each fault models as compared with the fault-dependent one-pattern test generation method, using the same test length. The proposed method is also effective in detecting delay faults of small size because the values of SDQL are smaller compared to those of the fault-dependent one-pattern test generation method. Therefore, we conclude that the proposed method is better than the fault-dependent one-pattern test generation method.

Next, the experimental results are considered for the proposed test generation with test length constraints. Stuck-at fault can be completely tested and the test length is greatly reduced compared with the fault-independent one-pattern test generation method. Moreover, high fault coverage for bridging fault and transition fault can be obtained. In particular, for s386, when test length constraint was set to 500, the weighted two-state transition coverage increased by 10.99%, the transition coverage increased by 5.17%, SDQL decreased by 85 ppm, and the quality of the timing fault testing was improved. With these, we can see the comparison between the proposed test generation method and the fault-independent one-pattern test generation. The proposed method drastically reduced the test length and it is effective in detecting delay faults of small sizes because the values of SDQL are smaller than those of the fault-independent one-pattern test generation method. As for a transition fault, the proposed method increased fault coverage using a realistic test length.

Next, the experimental results are considered for the proposed method when specified fault models are set to stuck-at-fault and transition fault. Stuck-at fault and transition fault can be completely tested and the test length was greatly reduced as compared with the fault-independent one-pattern test generation method. Moreover, high fault coverage for a bridging fault can be obtained. In particular, for pma, when test length constraint was set to 3000, the weighted two-state transition coverage increased by 1.88%, the transition coverage increased by 11.53%, SDQL decreased by 339 ppm, and the quality of the timing fault testing improved.

6. Conclusion

This paper proposed a test generation method to detect specified fault models completely and to increase defect coverage as much as possible under the test length constraint. Weighted state transition coverage as a measure of test quality is also presented. The proposed test generation method was evaluated for MCNC '91 benchmark circuit and the following conclusions were obtained.

(1) The proposed test generation method increased the test quality of logical fault testing and the timing fault testing compared with the fault-dependent one-pattern test generation method.
(2) The proposed test generation method greatly reduced the test length compared with the fault-independent one-pattern test generation method and the quality of both the logical fault testing and the timing fault testing were comparatively high.

Acknowledgment

This work was supported in part by the scholarship of Futaba Electronics Memorial Foundations in 2008 and in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B (No. 20300018).

References

[1] H. Fujiwara, Logic Testing and Design for Testability, MIT Press, 1985.
[2] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems Testing and Testable Design, IEEE Press, 1995.
[3] P.C. Maxwell, R.C. Aitken, R. Kollitz, and A.C. Brown, "IDDQ and AC scan: The war against unmodelled defects," Proc. IEEE Int. Test Conf., pp.250–258, Oct. 1996.
[4] A. Krstic and K.-T. Cheng, Delay Fault Testing for VLSI Circuits, Kluwer Academic Publishers, 1998.
[5] S. Ohtake, T. Masuzawa, and H. Fujiwara, "A non-scan approach to DFT for controllers achieving 100% fault efficiency," J. Electronic Testing: Theory and Applications (JETTA), vol.16, no.5, pp.553–566, Oct. 2000.
[6] H. Wada, T. Masuzawa, K.K. Saluja, and H. Fujiwara, "Design for strong testability of RTL data paths to provide complete fault efficiency," Proc. 13th Int. Conf. on VLSI Design, pp.300–305, 2000.
[7] T. Hosokawa and H. Fujiwara, "A functional test method for state observable FSMs," IEEE 6th Workshop on RTL and High Level Testing (WRTLT'05), pp.123–130, July 2005.
[8] T. Hosokawa, R. Inoue, and H. Fujiwara, "Fault-dependent/independent test generation methods for state observable FSMs," IEEE 16th Asian Test Symposium (ATS'07), pp.275–278, Oct. 2007.
[9] T. Hosokawa and K. Yamazaki, "An n-detection test generation method to increase fault sensitization coverage," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J90-D, no.6, pp.1474–1482, June 2007.
[10] S. Yang, "Logic synthesis and optimization benchmarks user guide," Technical Report 1991-IWLS-UG-Saeyang, Microelectronics Center of North Carolina, 1999.
[11] S. Ohtake, H. Wada, T. Masuzawa, and H. Fujiwara, "A non-scan DFT method at register-transfer level to achieve complete fault efficiency," IEEE Proc. Asian South Pacific Design Automation Conference, pp.599–604, 2000.
[12] Y. Takamatsu, T. Shiosaka, T. Yamada, and K. Yamazaki, "A fault model and test generation for bridging faults in CMOS circuit," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J81-D, no.6, pp.872–879, June 1998.
[13] S. Kajihara, K. Ishida, and K. Miyase, "Average power reduction in scan testing by test vector modification," IEICE Trans. Inf. & Syst., vol.E85-D, no.10, pp.1483–1489, Oct. 2002.
[14] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.
[15] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara, "Feasibility evaluation of the statistical delay quality model (SDQM)," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J89-D, no.8, pp.1717–1728, Aug. 2006.

Ryoichi Inoue received the M.E. degree in Mathematical information engineering, College of industrial technology, Nihon University, in 2008. He is graduate student of D.E. in Nihon University. His research interest includes design for testability and high level testing.

Toshinori Hosokawa received the B.E. degree in Electronics and Communication Engineering from Meiji University, Kawasaki, Japan, in 1987. He also received the Ph.D. degree from Meiji University in 2001. He was with Matsushita Electric Industrial Co., Ltd from 1987 to 2003. He was temporarily with Semiconductor Technology Academic Research Center (STARC) from 2000 to 2003. He was also a lecturer at Meiji University in 2001 and 2002. Presently he is a Professor at Department of Mathematical Information Engineering, College of Industrial Technology, Nihon University, Chiba, Japan. His research interests are automatic test pattern generation, fault simulation, design for testability, Synthesis for Testability, high level testing, logic simulation engine, and hardware/software co-verification. He is a member of IEEE (Institute of Electrical & Electronics Engineers) and IPSJ (Information Processing Society of Japan).

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received many awards including Okawa Prize for Publication, IEEE CS (Computer Society) Meritorious Service Awards, IEEE CS Continuing Service Award, and IEEE CS Outstanding Contribution Awards. He served as an Editor and Associate Editors of several journals, including the IEEE Trans. on Computers, and Journal of Electronic Testing: Theory and Application, and several guest editors of special issues of IEICE Transactions of Information and Systems. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a fellow of the IPSJ.