A Design Scheme for Delay Fault Testability of Controllers Using State Transition Information

Tsuyoshi Iwagaki Satoshi Ohtake Hideo Fujiwara

Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City 630-0192, Japan
E-mail: {*tsuyo-i, ohtake, fujiwara*}*@is.aist-nara.ac.jp*

Abstract

This paper proposes a non-scan testing scheme to enhance delay fault testability of controllers. In this scheme, the original behavior of a given controller is used in test application, and the faults which cannot be detected by the original behavior are tested by an extra logic called an invalid test state/transition generator (ISTG). Our scheme allows the following: (1) use of a combinational test generation tool, (2) achieving short test application time and (3) at-speed test. We experimentally show the effectiveness of our method. In our method, unlike scan-based methods, ISTGs can be designed flexibly in response to test qualities demanded by circuit designers.

1. Introduction

Modern high speed VLSI circuits need delay fault testing because conventional stuck-at fault testing cannot guarantee the timing correctness of the circuits. Usually, such circuits are designed at register-transfer level (RTL). A VLSI circuit designed at RTL generally consists of a controller, represented by a state transition graph (STG), and a data path, represented by hardware elements (e.g, registers, multiplexers (MUXs) and operational modules). Recently, *design for testability (DFT)* techniques for RTL circuits have been proposed [5].

In general, delay test generation for VLSI circuits is a hard problem. In [1], all the redundant path delay faults in a sequential circuit are classified into three categories: (1) combinationally non-activated paths, (2) sequentially non-activated paths and (3) unobservable fault effect. A sequential test generation tool (ATPG) spends huge time to identify these redundant faults, and it is virtually impossible to identify all of them. To facilitate delay test generation, standard scan designs [8, 7, 2, 9] and enhanced scan designs [3, 1] have been proposed [4]. These design methods make most or all faults in categories (2) and (3) irredundant by making all the flip-flops (FFs) completely controllable and observable. Thus, since we need only identifying faults in category (1) by using a combinational ATPG, the test generation time is significantly reduced and the fault efficiency becomes higher. However, in scan-based delay testing, due to the scan-shift operation, the test application time becomes longer. In addition, the scan-shift operation is performed at a low clock speed while second vectors of two-pattern tests are launched at the rated clock speed. This situation may cause inductive voltage drops because the operating speed, i.e., circuit current, rapidly changes. As a consequence of the voltage drops, the test will fail. Therefore, it is desirable that the operating speed is constant in test application. For stuck-at fault model, a DFT method for controllers, which overcomes the drawbacks of scan-based testing, has been proposed [6]. This method is a non-scan based one to achieve 100% fault efficiency by using a combinational ATPG, short test application time and at-speed test. In this method, the above merits are realized by appending an extra logic, called an *invalid test state generator (ISG)*, and some extra pins to the original controller.

This paper proposes a non-scan testing scheme, which is an extension of one in [6], to enhance delay fault testability of controllers. In this scheme, the original behavior of a given controller is used in test application. For only the faults which cannot be detected by the original behavior, we append an extra logic, called an *invalid test state/transition generator (ISTG)*, to the original controller. In order to design ISTGs flexibly in response to test quality demanded by circuit designers, we classify the redundant faults in a controller into five categories. Based on these categories, we can choose test qualities for delay faults by designing ISTGs appropriately. Our scheme allows the following: (1) use of a combinational ATPG, (2) achieving short test application time and (3) at-speed test. Experimental results show the effectiveness of our method.

2. Preliminaries

2.1. Target circuit and fault model

In this paper, we target controllers represented by STGs. Figure 1 is an example of a controller represented by an STG, and Figure 2 is a gate-level implementation of a controller. It is assumed that a controller has a reset signal, i.e., we can make a transition from any state to the reset state by activating the reset signal. We also assume that, for a given

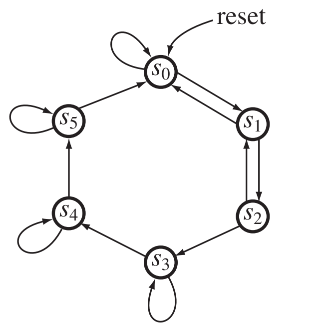


Figure 1. STG representing a controller.

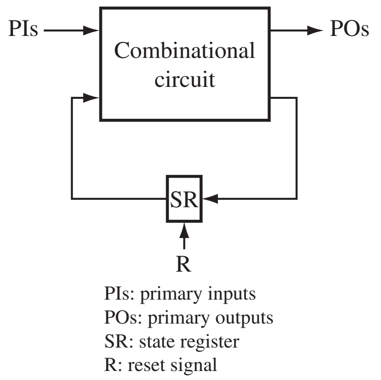


Figure 2. Synthesized controller.

controller, the mapping information in a state assignment is known. Our target fault model is the delay fault model.

2.2. Terminologies

Here, we define several terminologies. For any value of the state register (SR) in a sequential circuit synthesized from a given STG, if the corresponding state of the value is reachable from the reset state in the STG, then the state is called a *valid state*. Otherwise, it is called an *invalid state*. For a synthesized controller (Figure 2), a combinational circuit extracted from the controller by replacing the SR with pseudo primary inputs (PPIs) and pseudo primary outputs (PPOs) is called a *combinational test generation model* (Figure 3). Given a controller, each two-pattern test, (V_1, V_2), for the combinational test generation model can be denoted as $(I_1\&S_1, I_2\&S_2)$, where I_1 and I_2 are the values of primary inputs (PIs), S_1 and S_2 are the values of PPIs, and & is concatenation. Suppose the value of a present state is S_1 in an STG. Then, if the value of the next state is S_2 when I_1 is applied, the two-pattern test is called a *valid two-pattern test*. Otherwise, it is called a *invalid two-pattern test*. The transition corresponding to a valid (resp. invalid) two-pattern test is called a *valid* (resp. *invalid*) *test transition.* A valid state that appears in some valid/invalid two-pattern tests is called a *valid test state*, and an invalid state that appears in some invalid two-pattern tests is called an *invalid test state*. We show an example of test states/transition in Figure 4.

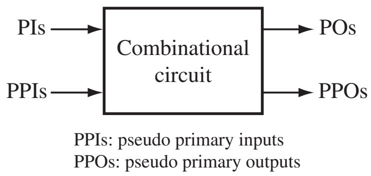


Figure 3. Combinational test generation model.

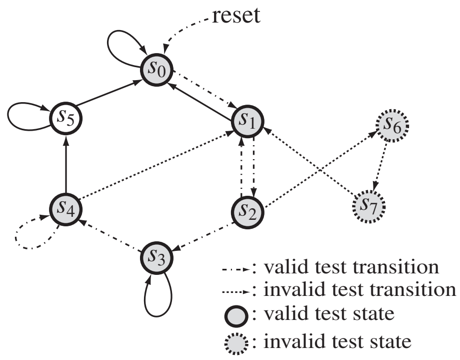


Figure 4. Test states/transitions.

3. Proposed method

3.1. Test architecture

In our testing scheme, the original behavior of a given controller is used in test application, i.e., valid two-pattern tests are applied by the original behavior. The faults that cannot be detected by the original behavior are tested by an extra logic called an *invalid test state/transition generator (ISTG)*. Our test architecture is shown in Figure 5. In Figure 5, respective DFT elements play the following roles:

- The ISTG generates invalid two-pattern tests.
- The extra pins of t_{sel} distinguish among invalid two-pattern tests in which first vectors are the same.
- The extra pins of t_{out} observe the value of the SR.
- The extra pin of t_{mode} and MUXs switch between the signal from the combinational part of the controller and that from the ISTG.

This architecture can achieve short test application time and at-speed test because the scan-shift operation is never used.

3.2. Flow of our method

The procedure of our method is as follows:

Step 1: Generate valid two-pattern tests under constraints.
Step 2: Generate invalid two-pattern tests including *don't care* (X) values under no constraints.
Step 3: Make an ISTG.
Step 4: Generate a test sequence for the original circuit.

In the next subsection, the details of these steps are explained.

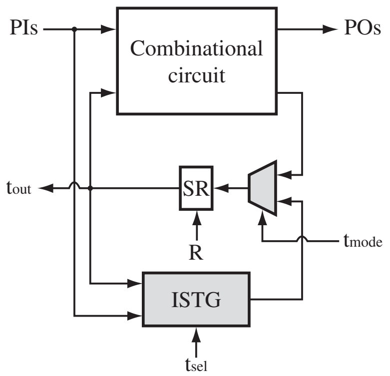
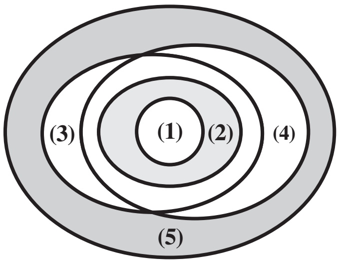


Figure 5. Proposed test architecture.



(1) Combinationally redundant faults
(2) Sequentially redundant faults at logic level with t_{out}
(3) Sequentially redundant faults at logic level without t_{out}
(4) Sequentially redundant faults at functional level with t_{out}
(5) Sequentially redundant faults at functional level without t_{out}

Figure 6. Classes of redundant faults.

3.3. Details of our procedure

First, we describe several classes of the redundant faults in a controller. These redundant faults are classified into five categories shown in Figure 6 based on t_{out}, which is our DFT element. The redundant faults in class (1) are redundant in the combinational part of the controller. Some irredundant faults in the combinational part become redundant due to the limitation of state transitions in the synthesized controller. Such faults belong class (3). In synthesizing a given STG, some new states and transitions are generally added to the synthesized controller. This implies that some irredundant faults in the complement of class (3) become redundant if we only consider the original behavior of the given STG. We classify these faults into class (5). If we append t_{out} to the synthesized controller, some redundant faults in classes (3) and (5) are detectable. Thus, classes (3) and (5) change into classes (2) and (4) respectively if t_{out} is added. In the following discussion, we use the above classification.

In Step 1 of the previous subsection, for the combinational test generation model of a given controller, we use a combinational ATPG. In order to generate valid two-pattern tests, we give some information (constraints) to the ATPG. A constraint is a tuple, (C_1, C_2), including the values of PIs and PPIs. We extract constraints from a given STG. The values of PIs and PPIs corresponding to a transition in the STG are used as the value of a constraint. Some parts of a constraint have unspecified values. In generating a two-pattern test, the values of C_1 and C_2 are set as the values of the first vector and second vector, respectively. Then, the ATPG only generates vectors for the unspecified parts of the first vector and second vector. If the value corresponding to a transition in a given STG is used as a constraint, generated two-pattern tests under the constraint can be always applied by the original behavior of the controller. In Step 1, we use all the constraints corresponding to the transitions in the STG. Therefore, we can identify all the redundant faults in class (4). If the designer of a given controller judges that it is sufficient to test only the faults in the complement of class (4), Steps 2 and 3 are skipped, i.e., only t_{out} is appended to the controller as a DFT element. However, the remaining faults that are not detected by Step 1 may affect the operation in a manufactured chip if at least one of them exists in the chip. This is because that the remaining faults may be activated due to new states and transitions appended to the original controller in synthesizing. Steps 2 and 3 should be performed if the designer is concerned with these faults.

In Step 2, we generate two-pattern tests, which are invalid, including X values for the remaining faults in Step 1 under no constraint. In Step 3, for each X in invalid two-pattern tests, we contrive to assign 0 or 1 in order to reduce the hardware (area and pin) overhead and test application time. This step identifies all the redundant faults in class (1).

In Step 3, we construct an ISTG. An ISTG realizes the functions of invalid two-pattern tests for the redundant faults in class (4) except (1). For example, given n invalid two-pattern tests $t_1 = (I_1^1\&S_1^1, I_2^1\&S_2^1)$, $t_2 = (I_1^2\&S_1^2, I_2^2\&S_2^2)$, $\ldots, t_n = (I_1^n\&S_1^n, I_2^n\&S_2^n)$, an ISTG must realize the functions shown in the truth table (Table 1). Note that if there exist m two-pattern tests that satisfy $I_1^1\&S_1^1 = I_1^2\&S_1^2 = \cdots = I_1^m\&S_1^m$ and $S_2^i \neq S_2^j$ ($\forall i, j,\ 1 \leq i, j \leq m,\ i \neq j$), we need t_{sel} whose bit width is $\lceil \log m \rceil$ to distinguish among them.

Table 1. Truth table of an ISTG.

Inputs	Outputs
$I_1^1\&S_1^1$	S_2^1
$I_1^2\&S_1^2$	S_2^2
$\vdots$	$\vdots$
$I_1^n\&S_1^n$	S_2^n

The area overhead of an ISTG depends on the number of varieties of transitions in invalid two-pattern tests. By using a technique, e.g., [10], which can identify some sequentially redundant faults in advance, and removing those faults from

Table 2. Intersection operator.

$\cap$	0	1	X
0	0	$\emptyset$	0
1	$\emptyset$	1	1
X	0	1	X

the fault list in Step 1 or 2, we can reduce the area overhead because the number of invalid two-pattern tests, i.e., the number of varieties of transitions, is reduced. Moreover, for each X in invalid two-pattern tests, if we contrive ways to assign 0 or 1, the hardware overhead and test application time can be reduced. We have investigated several techniques to assign the suitable value to X. However, in the following discussion, we only pick up two techniques (Strategy 1 and 2) to reduce the area overhead due to the limitation of space.

The problem to obtain an ISTG whose area is minimum is formalized as follows:

Given: The set of invalid two-pattern tests including X values.

Solution: An ISTG whose area is minimum.

As mentioned previously, the area of an ISTG depends on the number of varieties of transitions in invalid two-pattern tests. Therefore, if we reduce the number of rows in the truth table of the ISTG, the area of the ISTG can be reduced. We introduce a terminology here. For two vectors $V_i = (v_1^i, v_2^i, \ldots, v_n^i)$ and $V_j = (v_1^j, v_2^j, \ldots, v_n^j)$ ($v \in \{0, 1, X\}$), they are said to be *compatible* if $v_k^i \cap v_k^j \neq \emptyset$ $(\forall v_k^i, v_k^j)$, where $\cap$ is the intersection operator defined by Table 2.

Two techniques to reduce the area overhead are as follows:

Strategy 1: We first consider to minimize the number of rows in the truth table of the ISTG. Given two invalid two-pattern tests $t_i = (I_1^i \& S_1^i, I_2^i \& S_2^i)$ and $t_j = (I_1^j \& S_1^j, I_2^j \& S_2^j)$, if $I_1^i \& S_1^i \& S_2^i$ and $I_1^j \& S_1^j \& S_2^j$ are compatible, we can merge them into one function by assigning the suitable value. For example, given invalid two-pattern tests $t_1 = (0X\&01, 1X\&0X)$ and $t_2 = (X1\&X1, 00\&01)$, "$0X010X$" and "$X1X101$" are compatible. Therefore, if we assign the value as follows: $t_1 = (01\&01, 1X\&01)$, $t_2 = (01\&01, 00\&01)$, we can merge them into the function of "0101|01" (inputs|outputs). From this observation, we solve this problem as a clique partitioning problem (CPP) [5] on a compatibility graph, where a vertex t represent a vector and an edge (t_i, t_j) indicate that two vertices t_i and t_j are compatible.

Strategy 2: If the values of $S_2^1, S_2^2, \ldots, S_2^n$ in Table 1 are the same, the area of the ISTG will be very small. This is because that the ISTG always output the same value. In the second way to reduce the area of an ISTG, we consider to minimize the number of varieties of the output values. This problem is also solved by using a similar way of Strategy 1.

Note that, in the above discussion, although we only consider the area overhead, we have developed several techniques to reduce the bit width of t_{sel} and the test application time. However, we omit to describe them due to the limitation of space.

Table 3. Distance matrix.

	R	t_1	t_2	$\cdots$	t_n
R	∞	$d(R,t_1)$	$d(R,t_2)$	$\cdots$	$d(R,t_n)$
t_1	$d(t_1,R)$	∞	$d(t_1,t_2)$	$\cdots$	$d(t_1,t_n)$
t_2	$d(t_2,R)$	$d(t_2,t_1)$	∞	$\cdots$	$d(t_2,t_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_n	$d(t_n,R)$	$d(t_n,t_1)$	$d(t_n,t_2)$	$\cdots$	∞

In Step 4, in order to generate a test sequence, we determine an order of applying all the two-pattern tests to the original controller. Here, we consider the problem to obtain the test sequence which has the minimum length as an asymmetric traveling salesperson problem (ATSP) on a graph with a distance matrix, where a vertex t corresponds to a two-pattern test, and an arc (t_i, t_j) of corresponds to the path between t_i and t_j. The distance $d(t_i, t_j)$ means the minimum clock cycles that are needed to apply the first vector of t_j after applying t_i. Note that if the values of the second vector of t_i and the first vector of t_j are the same, the value of $d(t_i, t_j)$ is -1. Thus, we can generate a test sequence by solving the corresponding ATSP. Table 3 is an example of a distance matrix. In this table, $d(R, t)$ (resp. $d(t, R)$) denotes the minimum distance from the reset state R (resp. S_3) to S_1 (resp. R), where S_1 is the state in applying the first vector of t, and S_3 is the reaching state after applying the second vector of t. Note that, in a solution of the ATSP, vertex R is always visited first because the initial state is R in the test application.

4. Advantages of our method

4.1. Conventional methods and our method

In this subsection, we compare the proposed method to conventional methods (standard scan and enhanced scan methods).

Standard scan method: Test generation for a controller designed by this method requires a combinational ATPG which supports the skewed-load [7, 8] or broad-side [9] mode. In generating tests under the skewed-load mode, redundant faults in class (1) are dealt with as redundant faults in this scheme. When the mode is the broad-side one, redundant faults in class (1) and a subset of redundant faults in class (3) except (1) are dealt with as redundant faults. Generated two-pattern tests are applied to the controller through a scan chain in the skewed-load or broad-side fashion. The test application time is estimated as $n(n_{\mathrm{SSFF}} + 2) + n_{\mathrm{SSFF}}$, where n and n_{SSFF} are the number of two-pattern tests and standard scan FFs (SSFFs), respectively. In this method, each SSFF in the controller has an additional MUX. Therefore, the area overhead is $A_{\mathrm{MUX}} \times n_{\mathrm{SSFF}}$, where A_{MUX} is the area of the additional MUX. Due to the additional MUXs, the circuit delay increases. The increasing delay is equal to the delay of an MUX. This method needs three additional pins. Note that we assume that this method has a single scan chain.

Enhanced scan method: We can generate tests for a controller designed by this method by using a combinational ATPG. In the test generation, redundant faults in class (1) are dealt with as redundant faults in this scheme. The test application time is estimated as $2n(n_{\mathrm{ESFF}}+1)+n_{\mathrm{ESFF}}$, where n_{ESFF} is the number of enhanced scan FFs (ESFFs). Each ESFF in the controller has an additional MUX and a hold latch (HL) [3]. The area overhead is, therefore, $(A_{\mathrm{MUX}}+A_{\mathrm{HL}})\times n_{\mathrm{ESFF}}$. The delay penalty is higher than that of the standard scan method because of the HL. The increasing circuit delay is equal to the sum of the delays of an MUX and an HL. Furthermore, the pin overhead of this method is high compared with that of the standard scan method because the HL have to be controlled by an additional pin. The total number of additional pins is four. Note that it is also assumed that this method has a single scan chain.

Our method: In our method, we first generate tests for the combinational test generation model of a controller by using a combinational ATPG under the constraints extracted from the STG. The test generation is repeated n_c times, where n_c is the number of constraints. In generating tests, redundant faults in class (4) are dealt with as redundant faults in this process. Note that if a circuit designer decided that generating tests for faults in class (4) except (1) is not necessary, the next process can be omitted. Otherwise, we generate tests for the remaining faults under no constraint. In this process, redundant faults in class (1) are dealt with as redundant faults. For the generated invalid two-pattern tests for the remaining faults, we construct an ISTG. The test application time is determined by an order of applying all the two-pattern tests to the controller. Therefore, if we can obtain a good solution of the corresponding ATSP, the test application time can be reduced. The area overhead is $A_{\mathrm{MUX}}\times n_{\mathrm{FF}}+A_{\mathrm{ISTG}}$, where n_{FF} is the number of FFs and A_{ISTG} is the area of the ISTG. The proposed method has the same delay penalty compared to that of the standard scan method. The extra pins ($\mathrm{t_{sel}}$, $\mathrm{t_{out}}$ and $\mathrm{t_{mode}}$) are needed in our method. The sum of the bit width of these pins is $|\mathrm{t_{sel}}|+|\mathrm{t_{out}}|+1$. In a controller-data path circuit, which is composed of a controller and a data path, we can use the primary inputs and outputs of the data path as $\mathrm{t_{sel}}$ and $\mathrm{t_{out}}$, respectively. It allows the pin overhead to be reduced to two.

Here, we mention several essential differences among these methods. Since the scan-shift operation is needed in scan-based methods, we cannot perform at-speed test. However, our method allows it. Furthermore, in our method, we can reduce the area overhead by using a technique, e.g., [10], which can identify some sequentially redundant faults in advance, and removing those faults from a fault list. In contrast with our method, the hardware overheads of scan-based methods cannot be reduced even if the technique is used. In other words, ISTGs can be designed flexibly in response to test qualities demanded by circuit designers.

4.2. Experimental results

In this subsection, we evaluate test generation time, fault efficiency, test application time and hardware overhead of the proposed method.

We used four MCNC '91 benchmark circuits shown in Table 4. Columns "#PIs", "#POs", "#States" and "#FFs" denote the numbers of primary inputs, primary outputs, states and FFs, respectively. Column "Area" is the area estimated by Design Compiler (Synopsys). In synthesizing benchmark circuits, binary encodings and one-hot encodings were used.

Tables 5 and 6 show test generation results and hardware overheads in binary encodings and one-hot encodings, respectively. TestGen (Synopsys) was used as a delay test generation tool, and the transition fault model was targeted. In Tables 5 and 6, we compared our method (NS) to an standard scan technique (SS) and an enhanced scan technique (ES). Note that, in SS and ES, we assumed a single scan chain. For SS, we compared only the hardware overhead. Columns "TGT [s]", "FE [%]" and "TAT [CC (clock cycles)]" denote test generation time, fault efficiency under the non-robust criterion and test application time, respectively. Columns "Area OH [%]" and "Pin OH" denote area overhead and pin overhead, respectively. Column "Ratio of Area OH" denotes the ratio among the area overheads in respective methods.

In the test generation results, the test generation time of our method was longer than that of ES because we performed test generation for all the constraints of a given circuit. However, especially in the one-hot encodings, the test application time of our method was significantly short compared with that of ES. Furthermore, unlike ES, we can perform at-speed test in our method. This implies that the actual test application time of our method becomes much shorter than that of ES.

In the result of hardware overhead, the area overhead of SS was the smallest of all. The area overhead of our method was smaller than that of ES, except two cases. Note that, in this experiment, we randomly assigned 0 or 1 to X in invalid two-pattern tests for simplicity. Therefore, the area overhead of our method can be improved if we use the techniques described in Section 3.2. Moreover, by using a technique, e.g., [10], which can identify sequentially redundant faults in advance, and removing those faults from a fault list, the area overhead can be reduced. Also note that if we do not take into account the redundant faults in class (4) (Figure 6), ISTGs will not be necessary. In other words, only $\mathrm{t_{out}}$ is appended to the original controller as a DFT element. The pin overhead of our method was the largest of all. However, if we consider a controller-data path circuit, we can use the primary inputs and outputs of the data path as $\mathrm{t_{sel}}$ and $\mathrm{t_{out}}$, respectively. As a result of the sharing, the

Table 4. Circuit characteristics.

Circuit name	#PIs	#POs	#States	#FFs		Area	
				Binary	One-hot	Binary	One-hot
dk15	3	5	4	2	4	127	168
dk17	2	3	8	3	8	134	173
kirkman	12	6	16	4	16	360	500
sand	11	9	32	5	32	866	1,020

Table 5. Test generation results and hardware overheads in binary encodings.

Circuit name	TGT [s]		FE [%]		TAT [CC]		Area OH [%]			Pin OH			Ratio of area OH
	ES	NS	ES	NS	ES	NS	SS	ES	NS	SS	ES	NS	SS : ES : NS
dk15	0.16	0.77	100.00	100.00	170	138	11.0	22.0	11.8	3	4	4	1 : 1.10 : 1.01
dk17	0.13	1.27	100.00	100.00	211	144	15.7	31.3	18.7	3	4	4	1 : 1.14 : 1.03
kirkman	0.50	14.20	100.00	100.00	934	429	7.8	15.6	19.2	3	4	8	1 : 1.07 : 1.11
sand	1.52	30.09	100.00	100.00	1,973	979	4.0	8.1	17.8	3	4	10	1 : 1.04 : 1.13

Table 6. Test generation results and hardware overheads in one-hot encodings.

Circuit name	TGT [s]		FE [%]		TAT [CC]		Area OH [%]			Pin OH			Ratio of area OH
	ES	NS	ES	NS	ES	NS	SS	ES	NS	SS	ES	NS	SS : ES : NS
dk15	0.11	0.92	100.00	100.00	294	153	16.7	33.3	17.9	3	4	6	1 : 1.14 : 1.01
dk17	0.11	1.03	100.00	100.00	494	153	32.4	64.7	41.0	3	4	11	1 : 1.24 : 1.07
kirkman	0.57	15.88	100.00	100.00	2,974	645	23.0	45.9	31.6	3	4	18	1 : 1.19 : 1.07
sand	1.32	21.88	100.00	100.00	9,470	977	22.0	43.9	29.5	3	4	36	1 : 1.18 : 1.06

pin overhead can be reduced.

5. Conclusions and future works

This paper proposed a non-scan testing scheme to enhance delay fault testability of controllers. Our scheme allows the following: (1) use of a combinational ATPG, (2) achieving short test application time and (3) at-speed test. In the proposed method, for only the faults needed to be tested, we append DFT elements to a given controller. That is, in response to test qualities demanded by circuit designers, we can design ISTGs flexibly. We showed the effectiveness of our method by the experiment. Our future works are to develop ways to reduce pin overhead and test generation time.

Acknowledgments

We would like to thank Prof. Michiko Inoue of Nara Institute of Science and Technology for her valuable comments. This work was supported in part by 21st Century COE Program and in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(2) (No. 15300018).

References

[1] T. J. Chakraborty, V. D. Agrawal and M. L. Bushnell, "Design for testability for path delay faults in sequential circuits," *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 453–457, 1993.

[2] K.-T. Cheng, S. Devadas and K. Keutzer, "Delay-fault test generation and synthesis for testability under a standard scan design methodology," *IEEE Trans. on CAD*, Vol. 12, No. 8, pp. 1217–1231, Aug. 1993.

[3] B. I. Dervisoglu and G. E. Stong, "Design for testability: using scanpath techniques for path-delay test and measurement," *Proc. Int. Test Conf.*, pp. 365–374, 1991.

[4] A. Krstić and K.-T. Cheng, *Delay fault testing for VLSI circuits*, Boston: Kluwer Academic Publishers, 1998.

[5] M. T.-C. Lee, *High-level test synthesis of digital VLSI circuits*, Boston: Artech House, 1997.

[6] S. Ohtake, T. Masuzawa and H. Fujiwara, "A non-scan approach to DFT for controllers achieving 100% fault efficiency," *Journal of Electronic Testing: Theory and Applications (JETTA)*, Vol. 16, No. 5, pp. 553–566, Oct. 2000.

[7] S. Patil and J. Savir, "Skewed-load transition test: part II, coverage," *Proc. Int. Test Conf.*, pp. 714–722, 1992.

[8] J. Savir, "Skewed-load transition test: part I, calculus," *Proc. Int. Test Conf.*, pp. 705–713, 1992.

[9] J. Savir and S. Patil, "On broad-side delay test," *Proc. VLSI Test Symp.*, pp. 284–290, 1994.

[10] R. C. Tekumalla and P. R. Menon, "On redundant path delay faults in synchronous sequential circuits," *IEEE Trans. on Computers*, Vol. 49, No. 3, Mar. 2000.