IEEE the 8th Asian Test Symposium (ATS'99), pp. 192-199, Nov. 1999.

Static and Dynamic Test Sequence Compaction Methods for Acyclic Sequential Circuits Using a Time Expansion Model

Toshinori Hosokawa*1, Tomoo Inoue*2, Toshihiro Hiraoka*1, Hideo Fujiwara*3

*1 Corporate Semiconductor Development Division, Matsushita Electric Industrial Co., Ltd.
3-1-1 Yagumo-nakamachi, Moriguchi, Osaka 570-8501, Japan
*2 Faculty of Information Sciences, Hiroshima City University,
3-4-1, Ozukahigashi, Asaminami-Ku, Hiroshima, 731-3194, Japan
*3 Graduate School of Information Science, Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara 630-0101, Japan

Abstract

Test sequences for acyclic sequential circuits can be generated using a time expansion model. The test sequences have features that: (1) the length of each test sequence for each target fault is uniform, and (2) positions of 'don't cares' (X) of each test sequence for each target fault are independent of any target fault. In this paper, focusing on the features, we present two test sequence compaction methods: static compaction and dynamic compaction. The static test sequence compaction method uses a template . The dynamic test sequence compaction method uses a reverse transformation fault simulation: a fault simulation for a time expansion model with test patterns into which test sequences are reversely transformed after the static compaction. Experimental results for some acyclic sequential circuits show that the compaction methods reduce the number of test patterns by 66% to 81%.

1 Introduction

Automating test design for LSIs is required when it comes to large and complex LSIs. Test cost for LSIs can be divided into points of view of:

(1) Test pattern generation time to achieve high fault efficiency,

and

(2) Test application time at automatic test equipments.

Test generation for sequential circuits is generally considered to be a hard problem. For such sequential circuits, design for testability (DFT) such as full scan design [1, 2] and partial scan design is an important approach to reduce the test cost of (1). It is an important issue to select flip-flops(FFs) to be replaced with scan FFs in partial scan design [3, 4, 5, 6] . It is well known that acyclic sequential circuits can be generated test sequences for using a combinational ATPG (automatic test pattern generator) [6, 7, 8, 9, 10] . Thus, in partial scan circuits, high fault efficiency can be achieved with smaller hardware overhead than in full scan design circuits when FFs to be replaced with scan FFs are selected so that the kernel circuit, which is the portion of the circuit excluding the scan FFs, is the acyclic sequential circuit. Especially, [9, 10] presented the test sequence generation method using a time expansion model.

On the other hand, a test pattern compaction technique was proposed to reduce the test cost of (2). Recently many papers [11, 12, 13, 14, 15, 16, 17] are presented on a test pattern compaction technique. Test pattern compaction methods [11, 12, 13]for combinational circuits and test sequence compaction methods [14, 15, 16, 17] for sequential circuits were presented. In this paper, we present the test sequence compaction methods to reduce test sequences generated using a time expansion model. We turn our attention to the rule that the position of don't cares(Xs) in any test sequence generated using a time expansion model is independent of the values of the test sequence and present two compaction methods: static compaction method and dynamic compaction method.

This paper is organized as follows. Section 2 describes a test sequence generation method for acyclic sequential circuits using a time expansion model. Section 3 presents the static test sequence compaction method. Section 4 presents the dynamic test sequence compaction method. Section 5 provides experimental results. Finally, Section 6 describes conclusions and future works.

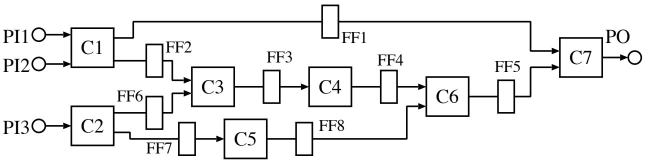


Figure 1. Acyclic sequential circuit: S.

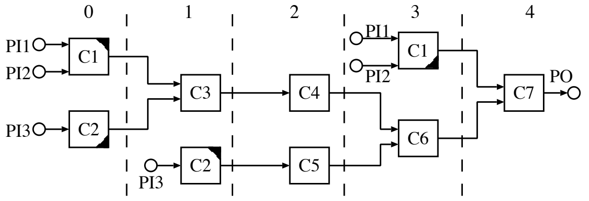


Figure 2. Time expansion model of S : $C(S)$.

2 Test Sequence Generation Method for Acyclic Sequential Circuits Using a Time Expansion Model

2.1 Test Sequence Generation

The test sequence generation method [9] for acyclic sequential circuits using a time expansion model is explained using an example. Fig.1 shows the example of acyclic sequential circuit S. In this figure, $FF1, FF2, ..., FF8$ represent FFs, $C1, C2, ..., C7$ represent combinational logic blocks, $PI1, PI2$, and $PI3$ represent primary inputs, and PO represents a primary output.

Fig.2 shows $C(S)$ which is the time expansion model of S. $C(S)$ is a combinational circuit into which combinational logic blocks of S are expanded in the input direction from each primary output to primary inputs. The combinational logic blocks, primary inputs, and primary output of $C(S)$ have label t (t is a set of integers). Generally in a time expansion model, assuming that A and B are combinational logic blocks and mutually connected, and $t(A)$ and $t(B)$ are labels of A and B, respectively, $|t(A) - t(B)|$ shows the number of FFs between A and B in the corresponding acyclic sequential circuit. In Fig.2, the number located at the top of each column denotes the value of the labels of the combinational logic blocks, primary inputs, and primary output in the column. Also a highlighted part in a combinational logic block represents a portion of the lines and gates which are not combinationally reachable to any primary output and any input of other combinational logic blocks, and they are removed in $C(S)$.

A fault f_e in $C(S)$ corresponding to a fault f_a in S is a multiple fault which exists in the same line in each combinational logic block in $C(S)$ corresponding to a combinational logic block in S where f_a exists. For example, a fault in $C1$ in S is dealt with as a two fold fault in $C(S)$ because two combinational logic blocks ($C1s$) exist in $C(S)$ as shown in Fig.2. Here, the following theorem can be obtained [9] when let F_a be the set of faults in S, and let F_e be the set of faults in $C(S)$ corresponding to faults in S.

(1) There exists only one fault $f_e \in F_e$ in $C(S)$ corresponding to each fault $f_a \in F_a$ in S.

(2) There exists a test pattern for a fault $f_e \in F_e$ corresponding to the $f_a \in F_a$ if and only if there exists a test sequence for fault f_a.

(3) A test pattern for a fault $f_e \in F_e$ can be transformed into a test sequence for the fault $f_a \in F_a$ corresponding to fault f_e.

Thus a test pattern for a fault in $C(S)$ can be transformed into a test sequence for S and detect the fault in S corresponding to the fault in $C(S)$ with the transformed test sequence. Hence a combinational ATPG capable of dealing with multiple stuck-at-faults (denoted by MC-ATPG) can be applied to $C(S)$ and generate test patterns. After that, test patterns generated for $C(S)$ are transformed into test sequences for S referring to values of labels where each primary input exists.

Assuming that ($PI1(0)$, $PI2(0)$, $PI3(0)$, $PI3(1)$, $PI1(3)$, $PI2(3)$) = (0, 1, 0, 0, 1, 0) is the test pattern for a fault in $C(S)$, in S, the value of $PI1$ at time 0 is 0, the value of $PI2$ at time 0 is 1, the value of $PI3$ at time 0 is 0, the value of $PI3$ at time 1 is 0, the value of $PI1$ at time 3 is 1, and the value of $PI2$ at time 3 is 0. Hence the test pattern is transformed into the test sequence as shown in Table 1(a). Here, $PIi(t)$ is the value of PIi which exists in label t.

2.2 Test Sequence Compaction

T_1 and T_2 are test sequences for S transformed from two test patterns for $C(S)$: $t_1 = (0, 1, 0, 0, 1, 0)$ and $t_2 = (1, 1, 1, 0, 0, 0)$. Table 1 (a) shows T_1 and Table 1 (b) shows T_2. T_2 can be placed at time 2 in T_1 because some Xs exist in T_1. Thus T_1 and T_2 is compacted into the test sequence T shown in Table 2. Also in test sequence transformed, the positions where 0s or 1s are specified and the positions where Xs are specified are uniform for every transformed test sequence. Whether several test sequences are compatible or not can be determined independently of values of test sequences from this information. Thus a test compaction method can be statically determined before a test sequence generation and test sequences can be compacted fast. The static compaction method is described in section 3.

Some Xs remain in T as shown in Table 2 and T' is generated as eshown in Table 3 after 0s or 1s are set to Xs in T at random. For example, in T', the test sequence from time 1 to time 5 is different from T_1 and

Table 1. Test Sequence for S.

(a) Test sequence T_1

Time	$PI1$	$PI2$	$PI3$
0	0	1	0
1	X	X	0
2	X	X	X
3	1	0	X
4	X	X	X

(b) Test sequence T_2

Time	$PI1$	$PI2$	$PI3$
0	1	1	1
1	X	X	0
2	X	X	X
3	0	0	X
4	X	X	X

Table 2. Compacted test sequence:T.

Time	$PI1$	$PI2$	$PI3$
0	0	1	0
1	X	X	0
2	1	1	1
3	1	0	0
4	X	X	X
5	0	0	X
6	X	X	X

Table 3. Compacted test sequence: T'.

Time	$PI1$	$PI2$	$PI3$
0	0	1	0
1	1	0	0
2	1	1	1
3	1	0	0
4	1	0	1
5	0	0	0
6	0	1	1

T_2. Undetected faults in S may be detected if fault simulation is performed for S with this test sequence. Also the test sequence is reversely transformed into the test pattern, so that ($PI1(0)$, $PI2(0)$, $PI3(0)$, $PI3(1)$, $PI1(3)$, $PI2(3)$) = (1, 0, 0, 1, 1, 0) is extracted. Fault simulation can be also performed for $C(S)$ with this test pattern. This dynamic compaction method is described in section 4.

3 Static Test Sequence Compaction Method

3.1 Definitions

Definition 1 (Template): Let T be a test sequence for a sequential circuit with primary inputs $(P_0, P_1, ..., P_{w-1})$. Let w be the number of primary inputs. Let l be the length of T. The value of P_i at time t in T is denoted by $T(t, i)$. A test sequence T' obtained by replacing every value such that $T(t, i) = 0$ or $1 (0 \leq t < l, 0 \leq i < w)$ with 'b' is called a template of T. Also a test sequence which consists of Xs and/or 'bs' is simply called a template. The length of T' is called the length of a template or template length. Let $Maxt$ be maximum time in T' where 'b' exists. Let $Mint$ be minimum time in T' where 'b' exists. The value such that $Maxt - Mint + 1$ is called the effective length of T'. □

Definition 2 (Compatible): Let $T_1(T_2)$ be a test sequence for a sequential circuit with primary inputs $(P_0, P_1, ..., P_{w-1})$. Let w be the number of primary inputs. Let l_1 be the length of T_1. Let l_2 be the length of T_2. The values of P_i at time t in T_1 and T_2 are denoted by $T_1(t, i)$ and $T_2(t, i)$, respectively. T_2 is called to be compatible with T_1 with skew k if there exists k (k is non-negative integer) to satisfy one or both of conditions following:

(1) $k \geq l_1$.

Table 4. Operation $\cap_c$

$\cap_c$	b	X
b	ϕ	b
X	b	X

(2) $T_1(t, i) = X$ or $T_2(t - k, i) = X$ for any t, i such that $k \leq t < \min\{l_1, k + l_2\}, and, 0 \leq i < w$.

Also, $T_1(T_2)$ is called to be compatible with skew k if $T_1 = T_2$. □

T into which T_1 and T_2 are compacted can be represented by the following equations using operation $\cap_c$ as shown in table 4 for any i such that $0 \leq i < w$.

$$\begin{array}{ll} T(t, i) = T_1(t, i) & \text{for } 0 \leqq t < \min\{l_1, k\},\ k + l_2 \leqq t < l_1 \\ T(t, i) = T_1(t, i) \bigcap_c T_2(t\text{-}k, i) & \text{for } k \leqq t < \min\{l_1, k + l_2\} \\ T(t, i) = X & \text{for } l_1 \leqq t < k \\ T(t, i) = T_2(t\text{-}k, i) & \text{for } \max\{l_1, k\} \leqq t < k + l_2 \end{array}$$

Example 1: Fig.3 shows T' into which T is compacted with skew 2 when T is compatible with skew 2.

Definition 3: (Primitive Template) Let S be an acyclic sequential circuit. Let $C(S)$ be the time expansion model corresponding to S. Let t be a test pattern for $C(S)$. Suppose that any X is not included in t. Let T be a test sequence into which t is transformed. Let T' be the template of T. T' is called a primitive template of $C(S)$.

Example 2: Table 5 shows the primitive template of the time expansion model in Fig.2.

3.2 Problem Formulation

Assuming that the total number of test patterns which are generated in a time expansion model $C(S)$ is n and n test patterns are transformed into n test sequences, to minimize a total length of test sequences by compacting n test sequences is an ideal target. However, it is difficult to minimize a total length of test sequences by compacting n test sequences because a large amount of its calculation is required. In this subsection, the optimum problem to maximize the number of primitive templates which are compacted into a template with a fixed length and minimize the effective length of the template is formulated. Before this formulation, a compatibility graph which represents that a primitive template is compatible with skew k is defined as follows.

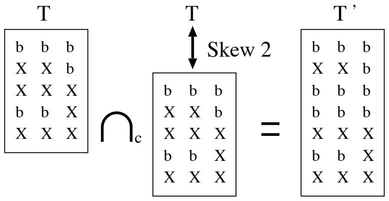


Figure 3. Example that T is compatible with skew 2.

Table 5. Primitive template for Fig.2.

Time	$PI1$	$PI2$	$PI3$
0	b	b	b
1	X	X	b
2	X	X	X
3	b	b	b
4	X	X	X

Definition 4 (Compatibility graph): A compatibility graph is an undirected graph $G(V, E, t)$, where a vertex $v \in V$ denotes a primitive template. Each vertex has a label $t : V \rightarrow Z+$ ($Z+$ denotes a set of non-negative integers). For any vertices $u, v (\in V, u \neq v), t(u) \neq t(v)$. An edge$(u, v) \in E$ denotes that a primitive template is compatible with skew $|t(u) - t(v)|$. □

A set of vertices which form clique (subcomplete graph) is extracted from a compatibility graph and a template for test sequence compaction is generated by compacting primitive templates of the number of vertices with the value of skew which is label t of each vertex. Let m be the number of vertices in a clique and let k be the effective length of a template. The total test length of test sequences compacted by using the template is $\frac{k \times n}{m}$ when n is multiples of m, and the template generated by maximizing m and minimizing k is effective for test sequence compaction. Thus this problem is dealt with as maximum clique extraction problem [18].

Maximum clique extraction problem

Given: A compatibility graph $G(V, E, t)$

Solution: An optimal clique C which minimizes $|\max t(v) - \min t(u)|$ in maximum cliques which maximize the size of a clique, where $u, v \in C, v \neq u, \max t(v)$ is the maximum value of a label, and $\min t(u)$ is the minimum value of a label.

Example 3: The primitive template is shown in Table 6. Fig.4 shows the compatibility graph which is considered whether the primitive template of Table 6 is compatible or not with the values of skew from 1 to 7. In Fig.4, three cliques, $C1, C2$, and $C3$ are considered. In this figure, a vertex with label t is denoted by V_t. The vertices included in $C1$ are V_0, V_1,and V_7. The vertices included in $C2$ are V_0, V_1, and V_2. The vertices included in $C3$ are V_0 and V_1. $C1$ and $C2$ are maximum cliques. The template for test sequence compaction is generated from $C1$. Three primitive templates are compacted into the template with skew 1 and 7, and the effective length of that is 13. Here, assuming that the total number of test patterns which are generated for a time expansion model is n, the length of total test sequences is $13n/3$ by using the template. The effective length of the template which is generated from $C2$ is 8, and the length of total test sequences is $8n/3$, the effective length of the template which is generated from $C3$ is 7, and the length of total test sequences is $7n/2$. Compared with the length of total test sequences of $C1$ and that of $C2$, the length of total test sequences of $C2$ is shorter than that of $C1$. Also compared with the length of total test sequences of $C2$ and that of $C3$, the length of total test sequences of $C2$ is shorter than that of $C3$. The template which is effective for test sequence compaction can be generated by extracting the maximum clique such that | the maximum value of a label of a vertex in a clique − the minimum value of a label of a vertex in a clique | is minimized.

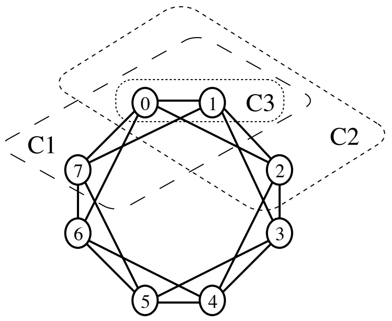


Figure 4. Compatibility graph for Table 6

Table 6. Primitive template

Time	$PI1$	$PI2$	$PI3$
0	b	b	b
1	X	X	X
2	X	X	X
3	X	b	X
4	b	X	X
5	X	X	b

3.3 Heuristic Algorithm

In this subsection, we present a heuristic algorithm for finding an optimal template for test sequence compaction. Any edge (u, v) of a compatibility graph is weighted as heuristics. The weight is denoted by $w(u, v)$. Let $nbr(u)$ and $nbr(v)$ be sets of adjacent vertices of u and v, respectively. Let A and B be $nbr(u) - \{v\}$ and $nbr(v) - \{u\}$, respectively. $w(u, v)$ is defined as $|A \cup B| - |A \cap B|$. The number of this weight is the number of adjacent vertices of v and u which can not be added to the clique when v and u are added to a clique.

Fig.5 shows a heuristic algorithm to extract a maximum clique from a given compatibility graph. Function Extract_max_clique in Fig.5 is described. First of all, vertex v which has label 0 is added to clique C. Next, the product set of adjacent sets for each u in C is found and let S be the product set (function Candidates). Processing that S is found, v is selected from S, and v is added to C is iterated until S becomes an empty set. The following three heuristic measures are calculated when only one vertex v in S is selected. Heuristic measure (1) is taken of heuristic measure (2) and heuristic measure (2) is taken of heuristic measure (3) when v in

```
Extract_max_clique(G)             /* G = (V,  E,  t) */
{
     C = { v s.t. t(v) == 0};
     while((S = Candidates(C)) ! = φ){
          v=Best_vertex(S,C) ;
          C = C ⋃ {v};
     }
     return C ;
}
Candidates (C)
{
     S = { v |  v∈ V - C  ∧ ∀ u ∈  C, ∃ (u, v) ∈  E};
     /* S is a set of vertices s.t. any vertex  in C is compatible with them */
     return S;
}
Best_vertex(S, C)                       /* heuristics */
{
     V1 = { v | Σ  w(u, v) is minimum in S, u  ∈ C};
     V2 = { v | |nbr(v)| is maximum in V1};
     V3 = {v | t(v) is minimum in V2};
     return (one vertex selected from V3);
}
```

Figure 5. Heuristic algorithm of a template generation.

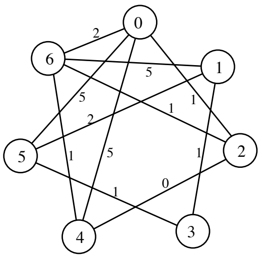


Figure 6. Compatibility graph for Table 5

S is selected (function Best_vertex).

Heuristic measure (1): v in S such that the sum total of $w(u, v)(u \in C)$ is minimized is selected and added to a set vertices $V1$, so that the number of vertices in C may increase. Thus, the number of primitive templates which is compacted into the template for test sequence compaction may increase.

Heuristic measure (2): v in $V1$ such that $|nbr(v)|$ is maximized is selected and added to a set vertices $V2$, so that the number of vertices in C may increase. Thus, the number of primitive templates which is compacted into the template for test sequence compaction may increase.

Heuristic measure (3): v in $V2$ such that $t(v)$ is minimum is selected and added to a set vertex $V3$, so that the effective length of generated template may be short.

After extraction of C, a template for test sequence compaction is generated by compacting primitive templates which are as many as vertices in C with the skews equal to labels of vertices in C. The generated template is iteratively used in ATPG. Also a template can be placed at the time corresponding to effective length of the previous template.

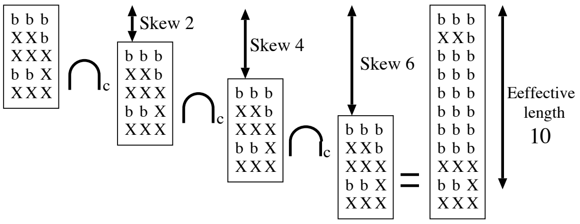


Figure 7. Template generated from Fig.6.

Example 4: Fig.6 shows the compatibility graph which is considered whether the primitive template of Table 5 is compatible or not with the values of skew from 1 to 6. In this figure, a vertex with label t is denoted by V_t. Also the set of vertices in the clique which is found by applying the heuristic algorithm shown in Fig.5 is $\{V_0, V_2, V_4, V_6\}$ and the template for test sequence compaction is generated as shown in Fig.7.

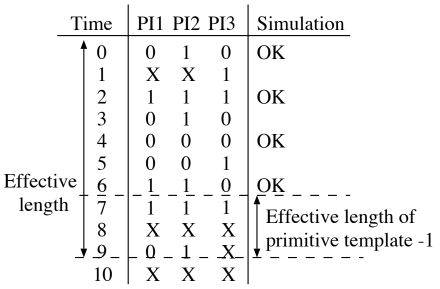
4 Dynamic Test Sequence Compaction Method

Fig.8 shows the test sequence generation algorithm for acyclic sequential circuits including the static test sequence compaction method using a template and the dynamic test sequence compaction method by a reverse transformation fault simulation.

First of all, the time expansion model is generated for a given acyclic sequential circuit. Next, the template for the static test sequence compaction is generated from the compatibility graph. Here, suppose that the template is generated by compacting N primitive templates. Thus, N test patterns in the time expansion model can be unconditionally transformed into N test sequences referring to the template (the static test sequence compaction). However, in the template, fault simulation is not performed with test sequences except N transformed test sequences. The processing that the test pattern is generated for a undetected fault f in the time expansion model, fault simulation is performed for the time expansion model with the test pattern, detected faults are removed from a set of undetected faults F, and the test pattern is transformed into the test sequence, is iterated. After the N iterations (n is equal to N), the test patterns for the time expansion model are extracted by reversely transforming test sequences with the length of a primitive template which fault simulation has not been performed with yet. More specifically, the head of the primitive template is placed on the head of a test sequence in the template with which fault simulation has not been performed and a test pattern is generated by extracting the values at ‘bs’ in the placed primitive

```
Test_generation_acyclic(S)
{
    Generate a time expansion model corresponding to S.
    Generate a template for a test sequence compaction.
        (compacted N primitive templates)
    n = 0.
    for (undetedcted fault f $\in$ F)
    {
        Select a fault f from undetected fault set F.
        Generate a test pattern for f in the time expansion model.
        Perform fault simulation for the time expansion model
        with the test pattern.
        Remove detected faults from F.
        Transform the test pattern into a test sequence with a test
        sequence compaction using the template.
        n++;
        if (n= = N)
        {
            Extract test patterns by a reverse transformation.
            Perform fault simulation for the time expansion
            model with the test patterns.
            Remove detected faults from F.
            n = 0;
        }
    }
}
```

Figure 8. Test sequence generation algorithm including static and dynamic compaction

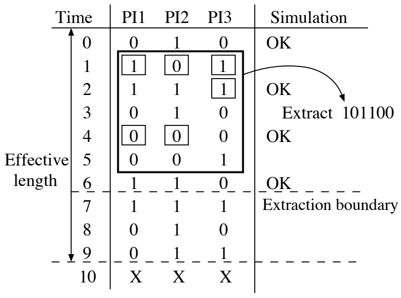


Time	PI1	PI2	PI3	Simulation
0	0	1	0	OK
1	X	X	1	
2	1	1	1	OK
3	0	1	0	
4	0	0	0	OK
5	0	0	1	
6	1	1	0	OK
7	1	1	1	
8	X	X	X	
9	0	1	X	
10	X	X	X	

Figure 9. Example of compaction using the template of Fig.7.

template from the test sequence. Next, fault simulation is performed for the time expansion model with the extracted test patterns and detected faults with the test patterns are removed from the set of undetected faults F (reverse transformation fault simulation). Here, suppose that before the reverse transformation fault simulation, 0 or 1 is set to each X in test sequences at random after the static test sequence compaction. Final test sequences are generated by iterating the processing that a template is placed at the time corresponding to the value of effective length of the previous template. Thus, the processing that N test patterns are generated and the reverse transformation fault simulation is performed, is iterated until the set of undetected faults F becomes empty.

Example 5: Fig.7 shows a template for the static test sequence compaction. Four primitive templates shown in Table 5 are compacted into the template shown in Fig.7. There are four test patterns generated in a time expansion model, ($PI1(0)$, $PI2(0)$, $PI3(0)$, $PI3(1)$, $PI1(3)$, $PI2(3)$) = (0, 1, 0, 1, 0, 1), (1, 1, 1, 0, 0, 0), (0, 0, 0, 1, 1, 1), (1, 1, 0, 1, 0, 1). The test sequence shown in Fib.9 is obtained so that four test sequences into which four test patterns are transformed are compacted using the template. The next template will be placed at time 10 because the effective length of the template is 10. Thus, 0 or 1 is set to each X from time 0 to time 9 in the test sequence at random. The test pattern $(1, 0, 1, 1, 0, 0)$ is extracted by transforming the test sequence from time 1 to time 5 reversely as shown in Fig.10. In Fig.10, the extracted values are boxed in small squares. The test pattern $(0, 1, 0, 0, 1, 1)$ is extracted by transforming the test sequence from time 3 to time 7 reversely and test pattern $(0, 0, 1, 0, 0, 1)$ is extracted by transforming the test sequence from time 5 to time 9 reversely in like manner.



Time	PI1	PI2	PI3	Simulation
0	0	1	0	OK
1	$\boxed{1}$	$\boxed{0}$	$\boxed{1}$	
2	1	1	$\boxed{1}$	OK
3	0	1	0	Extract 101100
4	$\boxed{0}$	$\boxed{0}$	0	OK
5	0	0	1	
6	1	1	0	OK
7	1	1	1	Extraction boundary
8	0	1	0	
9	0	1	1	
10	X	X	X	

Figure 10. Example of extraction of a test pattern for a time expansion model.

5 Experimental Results

Table 7 shows the properties of practical circuits (#1-#4) used on these experiments. In Table 7, #PI shows the numbers of primary inputs, #PO shows the numbers of primary outputs, #FF shows the numbers of FFs, SR shows the ratios of scan FFs to all FFs which are necessary to make circuits acyclic sequential circuits, and #GATE shows the numbers of gates (the conversion to NAND gates with 2 inputs) in the circuits. Table 8 shows the experimental results of templates generation using presented heuristic algorithm. In order to see the effectiveness of the number of vertices in a compatibility graph in a template generation, we made experiments on finding the maximum cliques for each circuit with different numbers of vertices in a compatibility graph. In Table 8, NV shows the numbers of vertices in the compatibility graphs, CPU shows times (unit: second) consumed to make compatibility graphs and to find cliques on SUN Ultra2 platform (frequency:

Table 7. Properties of experimental circuits.

CIR	#PI	#PO	#FF	SR(%)	#GATE
#1	97	16	176	0	4687
#2	113	16	272	0	4706
#3	342	11	1150	61.9	11682
#4	673	487	3744	72.5	51135

Table 8. Experimental results: Template generation.

CIR	NV	CPU	VL	NP	TL	CR
#1	8	0.01	12	3	984	4.00
	15	0.02	18	5	864	3.60
	114	1.16	119	37	738	3.22
	171	6.54	176	56	720	3.14
	200	8.35	205	66	708	3.11
#2	11	0.01	16	3	1616	5.33
	21	0.02	29	5	1762	5.80
	153	1.50	162	30	1635	5.40
	200	4.66	209	39	1622	5.36
	304	25.70	314	58	1638	5.41

Table 9. Experimental results: Heurisitic algorithm.

CIR	HVL	HNP	HCR	OVL	ONP	OCR
#1	206	66	3.12	205	67	3.06
#2	209	39	5.35	208	39	5.33

Table 10. Experimental results: Compaction method using a template.

CIR	NC	TC			3TC		
	TL	CPU	TL	TR	CPU	TL	TR
#1	1589	8.85	708	44.56	6.08	910	57.27
#2	3030	4.58	1622	53.53	5.86	1821	60.10
#3	1140	13.17	584	51.23	9.98	572	50.18
#4	13740	8.86	7573	55.12	5137.75	6791	49.43

296MHz, SPECint95:12.1, SPECfp: 18.3), VL shows the effective lengths of the generated templates, NP shows the numbers of primitive templates which are compacted into the templates, TL shows the lengths of total test sequences, CR shows the ratio of VL to NP (compaction efficiency). As CR decreases, a compaction efficiency becomes high. This table shows only the results for circuits #1 and #2. For circuits #3 and #4, optimal templates can be obtained independently of the numbers of vertices NV, i.e., the global optimal templates are the same as the local ones. In the concrete, the optimal templates for these circuits can be generated by using a compatibility graph which consists of two vertices connected with the shortest edge (i.e., $|t(v) - t(u)|$ is the minimum). As the number of vertices in the compatibility graph increases, CPU also increases. We found that the templates are generated in small computation time, especially for NV $\leq$ 200. The number NV of vertices in a compatibility graph used in all the experiments reported in the following is 200. The consideration between compaction efficiency and the number of vertices in a compatibility graph remains future works.

Table 9 shows the experimental results of the heuristic algorithm compared with maximum cliques obtained by calculating all cliques exhaustively in order to evaluate the efficiency of the heuristic algorithm. In Table 9, HVL shows the effective lengths of the templates generated using the heuristic algorithm, HNP shows the numbers of compacted primitive templates, HCR shows the compaction efficiencies (HLV/HNP), OVL shows the effective lengths of the templates generated from maximum cliques, ONP shows the numbers of compacted primitive templates, OCR shows the compaction efficiencies (OVL/ONP). The results in Table 9 show that the compaction efficiencies of templates generated using the heuristic algorithm are almost the same as those generated from the maximum cliques.

Table 10 shows the experimental results of the static test sequence compaction method using a template. In order to see the effectiveness of our static compaction method, we also made the static test sequence compaction based on three values (0, 1, and X) [8, 15]. The results based on three values, shown as 3TC in Table 10, are derived as follows:

(1) Let T be a set of all the generated test patterns. Select a test pattern t from T, and transform t into a test sequence (denoted by s).
(2) Select a new test pattern t' and transform t' into a test sequence (denoted by s').
(3) Compact s' into s with the minimum skew, i.e., search a time in s where s' can be placed from the head of s. Let s be a resultant test sequence obtained from s and s'.
(4) Repeat (2) and (3) until all the generated test patterns are compacted.

Also, in Table 10, NC shows the experimental results without a test sequence compaction and TC shows the experimental results with the static test sequence compaction using templates. CPU shows times (unit: second) consumed to compact test sequences on SUN Ultra2 platform (frequency: 296MHz, SPECint95:12.1, SPECfp: 18.3), TL shows the lengths of the total test sequences, and TR shows the ratios of TL of TC and 3TC to TL of NC. TC shortened the lengths of test sequences compared with NC by 45 to 55%. The lengths of test sequences of TC is shorter than those of 3TC for #1 and #2. The lengths of test sequences of 3TC is shorter than those of TC for #3 and #4. We found that the static test sequence compaction method is effective for the length of test sequences when the global optimal template is not the same as the local one. As for test sequence compaction time, the computation times of TC are almost same as those of 3TC for #1, #2, and #3. However, the computation time of TC is almost 57 times

Table 11. Experimental results: Compaction method using reverse transformation fault simulation.

CIR	TC		TCRF			
	CPU	TL	CPU	TL	TR1	TR2
#1	215.88	708	136.92	287	40.53	18.06
#2	138.24	1622	74.57	803	49.51	26.50
#3	44.32	584	51.54	351	60.10	30.79
#4	1150.23	7573	1525.21	4792	63.28	34.88

as fast as that of 3TC for #4 which is a large circuit. From this table, we can see that the static test sequence compaction method using a template can faster compact than three-value-based compaction method.

Table 11 shows the experimental results of the dynamic test sequence compaction method by reverse transformation fault simulation. In Table 11, TC shows the experimental results of the static test sequence compaction method using a template, TCRF shows the experimental results of the combination of the static test sequence compaction method and the dynamic test sequence compaction method, CPU shows times (unit: second) consumed to generate test sequences including test sequence compaction on SUN Ultra2 platform (frequency: 296MHz, SPECint95:12.1, SPECfp: 18.3), TL shows the lengths of the total test sequences, TR1 shows the ratios of TL of TCRF to TL of TC, and TR2 shows the ratios of TL of TCRF to TL of NC. TCRF shortened the lengths of test sequences compared with TC by 37 to 59%, and shortened the lengths of test sequences compared with NC by 66 to 81%. The combination of the static test sequence compaction method and the dynamic test sequence compaction method could furthermore enhance the compaction efficiency.

6 Conclusion

In this paper, we presented the static test sequence compaction method using a template and the dynamic test sequence compaction method by reverse transformation fault simulation for acyclic sequential circuits. We could obtain the following conclusions from the experimental results for practical circuits.

(1) Presented heuristic algorithm is effective.

(2) The static test sequence compaction method using a template could shorten the lengths of test sequences by 45 to 55%.

(3) The combination of the static test sequence compaction method and the dynamic test sequence compaction method could furthermore shorten the length of test sequences by 37 to 59%.

Our future work is that we will consider the relation between the compaction efficiency and the number of vertices in compatibility graph.

References

[1] H.Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[2] M.Abramovici, M.A.Breuer, and A.D.Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[3] K.-T.Cheng and V.D.Agrawal, *A partial scan method for sequential circuits with feedback*, IEEE Trans. Comput., Vol.39, No4, pp.544-548, Apr.1990.

[4] D.H.Lee and S.M.Reddy, *On determining scan flip-flops in partial scan design approach*, Proc. IEEE Proc. Int. Conf. Computer-Aided Design, pp.322-325, Nov. 1990.

[5] S.T.Chakradhar, A.Balakrishnan, and V.D.Agrawal, *An exact algorithm for selecting partial scan flip-flops*, Proc. ACM/IEEE Design Automation Conf., pp.81-86, June. 1994.

[6] R.Gupta, R.Gupta, and M.A.Breuer, *The BALLAST methodology for structured partial scan design*, IEEE Trans. Comput., Vol.39, No.4, pp.538-544, Apr. 1990.

[7] T.Takasaki, T.Inoue, and H.Fujiwara, *Partial scan design methods based on internally balanced structure*, IEEE Proc. Asia and South Pacific Design Automation Conf., pp.211-216, Feb. 1998.

[8] T.Hosokawa, T.Hiraoka, M.Ohta, M.Muraoka, and S.Kuninobu, *A partial scan design method based on n-fold line-up structures*, IEEE Proc. Asian Test Symp., pp.306-311, Nov. 1997.

[9] T.Inoue, T.Hosokawa, T.Mihara, and H.Fujiwara, *An optimal time expansion model based on combinational ATPG for RT level circuits*, IEEE Proc. Asian Test Symp., pp.190-197, Dec. 1998.

[10] A.Kunzmann and J.J.Wunderlich, An analytical approach to the partical scan problem, Journal of Electronic testing Theory and Applications, pp163–174, Vol.1, Num.2, May 1990.

[11] P.Goel and B.C.Rosales, *Test generation and dynamic compaction of tests*, IEEE Proc. Int. Test Conf., pp.189-192, Oct. 1979.

[12] G.Tromp, *Minimal test sets for combinational circuits*, IEEE Proc. Int. Test Conf., pp.204-209, Oct. 1991.

[13] S.Kajihara, I.Pomeranz, K.Kinoshita, and S.M.Reddy, *Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits*, IEEE Trans. on Computer-Aided Design, pp.1496-1504, Dec. 1995.

[14] I.Pomeranz and S.M.Reddy, *Vector restoration based static compaction of test sequences for synchronous sequential circuits*, IEEE Proc. Int. Conf. on Computer Design, pp.360-365, Oct. 1997.

[15] T.M.Niermann, R.K.Roy, J.H.Patel, and J.A.Abraham, *Test Compaction for Sequential Circuits*, IEEE Trans. on Computer-Aided Design, Vol.11, NO.2, pp.260-267, Feb. 1992.

[16] M.S.Hsiao, E.M.Rudnick, and J.H.Patel, *Fast algorithm for static compaction of sequential circuit test vectors*, IEEE Proc. VLSI Test Symp., pp188-195, Apr. 1997.

[17] S.T.Chakradhar and A.Raghunathan, *Bottleneck removal algorithm for dynamic compaction in sequential circuits*, IEEE Trans. on Computer-Aided Design, Vol.16, NO.10, pp.1157-1172, Oct. 1997.

[18] Giovanni De Micheli, *Synthesis and optimization of digital circuits*, McGraw-Hill, Inc., 1994.