A New ATPG Method for Efficient Capture Power Reduction During Scan Testing

Xiaoqing Wen [1], Seiji Kajihara [1], Kohei Miyase [2], Tatsuya Suzuki [1], Kewal K. Saluja [3], Laung-Terng Wang [4], Khader S. Abdel-Hafez [4], and Kozo Kinoshita [5]

[1] Dept. of CSE, Kyushu Institute of Technology, Iizuka 820-8502, Japan
[2] Innovation Plaza Fukuoka, Japan Science and Technology Agency, Fukuoka 814-0001, Japan
[3] Dept. of ECE, University of Wisconsin - Madison, Madison, WI 53706, USA
[4] SynTest Technologies, Sunnyvale, CA 94086, USA
[5] Faculty of Informatics, Osaka Gakuin University, Suita 564-8511, Japan

Abstract

High power dissipation can occur when the response to a test vector is captured by flip-flops in scan testing, resulting in excessive IR drop, which may cause significant capture-induced yield loss in the DSM era. This paper addresses this serious problem with a novel test generation method, featuring a unique algorithm that deterministically generates test cubes not only for fault detection but also for capture power reduction. Compared with previous methods that passively conduct X-filling for unspecified bits in test cubes generated only for fault detection, the new method achieves more capture power reduction with less test set inflation. Experimental results show its effectiveness.

1. Introduction

Scan testing, which is based on full-scan design and combinational automatic test pattern generation (ATPG), is the most widely adopted test scheme for digital integrated circuits. Due to its simplicity and efficiency, scan testing will remain dispensable in the deep submicron (DSM) era.

In a full-scan sequential circuit, all functional flip-flops (F/Fs) are replaced with scan F/Fs that operate in two modes: *shift* and *capture*. In shift mode, scan F/Fs form one or more scan chains directly accessible from a tester. This mode is used to load a test vector through shift-in or obtain a test response through shift-out, for the combinational portion of the sequential circuit. In capture mode, scan F/Fs operate as functional F/Fs and the response of the combinational portion for a test vector is loaded into them. Therefore, the task of testing a full-scan sequential circuit is reduced to that of testing its combinational portion, in that now it is sufficient to generate test vectors only for the combinational portion with combinational ATPG [1].

However, the applicability of scan testing is being severely challenged recently by the following four problems: (1) *test data volume*, (2) *test application time*, (3) *test heat*, and (4) *test-related yield loss*.

The problems of test data volume and test application time are caused by larger gate and F/F counts, longer scan chains, and the use of complex fault models, such as transition and path delay fault models. Many methods have been proposed to address these problems, using such approaches as test compaction, test compression-decompression, multi-capture test clocking, etc. [2].

The problem of test heat is caused by the accumulative impact of power dissipation in shift mode during scan testing [3]. This is because shifting a test vector or the response to a test vector through all scan chains needs a large number of consecutive clock pulses, depending on the maximum scan chain length. Accumulatively, shift power dissipation can cause excessive test heat, which may permanently damage the circuit under test or lower its reliability due to accelerated electromigration.

Many methods have been proposed to tackle the test heat problem through shift power reduction [4], and they are based on four major approaches: *scheduling*, *test vector manipulation*, *circuit modification*, and *scan chain modification*. Test scheduling considers the power budget in selecting modules to be tested simultaneously. Test vector manipulation includes low-power ATPG, static compaction, test vector modification, test vector reordering, test vector compression, and coding. Circuit modification includes transition blocking, clock gating, and multi-duty scan. Scan chain modification includes scan chain reordering, scan chain partitioning, and scan chain modification.

The problem of test-related yield loss is caused by such factors as wrong test set-up, excessive test power, over-test, etc. In the past, errors in the test set-up (test programs, automatic test equipment (ATE), and peripheral circuitry) were the major reason for yield loss during testing. Recently, excessive test power dissipation in scan testing has emerged as a significant yield-killer [5]. The reason is that severe IR drop may occur due to excessive test power dissipation when a clock pulse is applied in shift or capture mode, causing direct F/F malfunction and/or increasing circuit delay. This leads to faulted values in F/Fs, resulting in test-related yield loss on top of process-related yield loss.

In this paper, we focus on how to reduce test-related yield loss caused by excessive test power dissipation. The reason is that this problem is worsening rapidly, especially for large-scale, high-speed, and low-power DSM circuits.

Obviously, since IR drop may occur whenever an excessively large number of F/Fs change their output values simultaneously during scan testing, it is necessary to reduce test power dissipation in both shift and capture modes in order to reduce test-related yield loss. The current status of research on shift power reduction and capture power reduction is briefly summarized bellow:

Shift Power Reduction

Many methods [4] have been proposed for reducing test power dissipation in shift mode. Most of them are initially targeted for reducing average shift power but they usually can reduce peak shift power as well, making these methods also effective for reducing shift-induced yield loss. For example, the MD-Scan method [5] uses different phases in the same shift cycle for different clock domains to dramatically reduce the number of simultaneously switching F/Fs in shift mode. Note that this method is independent of test vectors, allowing test vectors to be manipulated for achieving other goals, such as capture power reduction.

Capture Power Reduction

Compared with shift power reduction, capture power reduction is a less researched yet more challenging area. Different from shift power reduction, capture power reduction usually relies on test vector manipulation. Most of previous methods are based on *X*-filling [6-8], i.e. properly assigning 0's and 1's to all unspecified bits (*X*-bits) in a *test cube* so as to reduce the capture power dissipation of the resulting fully-specified *test vector*. However, these methods all suffer from the problem that the specified bits in a test cube are usually determined only for fault detection, with capture power reduction being totally neglected.

There are two approaches to test cube generation: In the *in-ATPG* approach, logic values for some inputs of a circuit are determined only for the purpose of detecting a target fault, and the result is usually a test cube since not all inputs need to be assigned with logic values [1]. There is also a test generation method [9] that takes test power reduction into consideration, but it lacks generality in that one fault needs to be detected by a pair of input vectors based on a special clock-disabling low power design. In the *post-ATPG* approach, a fully-specified test vector or test set is given, and some bits are changed to *X*-bits if doing so does not affect fault coverage [7, 10]. Here again, capture power reduction is not considered in such *X*-bit identification.

Obviously, the *specified* bits in a test cube generated by the above methods are not necessarily good for capture power reduction although they can detect some faults. As a result, the overall capture power reduction effect may be unsatisfactory since *X*-filling can only reduce capture power related to the *unspecified* bits in a test cube.

In this paper, we propose a novel concept, called *capture-aware (CA) test cube generation*, for deterministically generating test cubes not only for fault detection but also for capture power reduction. The key idea is that, a target fault can usually be detected by many test cubes, and some of them can reduce capture power at the same time. In order to generate such a test cube, a unique PODEM-based algorithm is proposed by using *capture conflict* based backtrack to take capture power reduction into consideration and by using *implication stack restoration* to guarantee fault detection. Obviously, further conducing *X*-filling on test cubes generated in this way results in better performance for overall capture power reduction.

The rest of the paper is organized as follows: Section 2 describes the research background. Section 3 presents a new ATPG method for capture power reduction based on CA test cube generation and *X*-filling. Section 4 shows experimental results, and Section 5 concludes the paper.

2. Background

2.1 Capture Power Problem

Fig. 1 shows a general full-scan circuit with one scan chain for the sake of clarity. v is a test vector or a test cube. The combinational portion implements logic function F, and its functional response to v is $F(v)$. The PI and PPI bits in v as well as PO and PPO bits in $F(v)$ are denoted by $\langle v$: PI$\rangle$, $\langle v$: PPI$\rangle$, $\langle F(v)$: PO$\rangle$, and $\langle F(v)$: PPO$\rangle$, respectively.

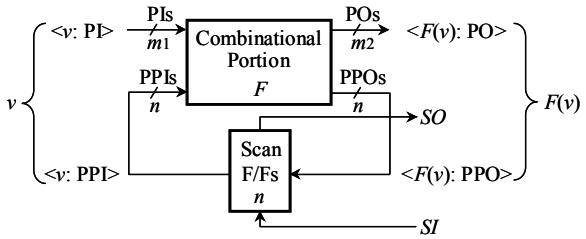


Fig. 1 A General Full-Scan Circuit.

Definition 1: If a bit a in $\langle v$: PPI$\rangle$ and its corresponding bit b in $\langle F(v)$: PPO$\rangle$ have opposite logic values at a scan F/F in Fig. 1, a *capture transition* is said to occur at the output of the scan F/F in capture mode during scan testing. The number of capture transitions for v is denoted by $CT(v)$.

Obviously, if v is a fully-specified test vector, $CT(v) = |\langle v$: PPI$\rangle \oplus \langle F(v)$: PPO$\rangle|$. It has been demonstrated that $CT(v)$ is closely correlated with the level of switching activity caused by the test vector v [7]. If $CT(v)$ is too large, excessive switching activity will occur when $\langle F(v)$: PPO$\rangle$ is captured into the scan F/Fs, resulting in IR drop and consequently, capture-induced yield loss [5, 8, 11].

In order to reduce capture-induced yield loss, a test vector v should have low switching activity in capture mode. This can be achieved by making sure that $CT(v)$ is under a pre-set limit. Therefore, the *LCP* (*Low Capture Power*) test generation problem can be formalized as follows:

LCP Test Generation Problem: Generate a test vector v for a full-scan circuit, such that $CT(v) < c\_limit$, where *c_limit* is a pre-set capture transition limit.

2.2 Previous Solutions

Most of previous LCP test generation methods are based on *X*-filling techniques [7, 8], i.e. properly assigning 0's and 1's to all unspecified bits in a test cube *v'* to reduce *CT*(*v*) of the resulting fully-specified test vector *v*. However, the specified bits in such a test cube are usually determined only for fault detection, not for capture power reduction.

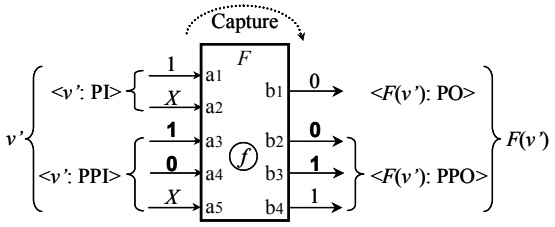


Fig. 2 Impact of specified bits in a test cube.

As shown in Fig. 2, the test cube is $v' = \langle a1, a2, a3, a4, a5\rangle = \langle 1, X, 1, 0, X\rangle$ for detecting the fault *f*. The specified bits in <*v'*: PPI> are $a3 = 1$ and $a4 = 0$. Obviously, this causes two capture transitions for $a3/b2$ and $a4/b3$. *X*-filling the *X*-bit $a5$ with 1 can avoid a capture transition for $a5/b4$ [7, 8]. However, such *X*-filling cannot reduce capture transitions related to the specified bits, $a3$ and $a4$.

2.3 Motivation

The example of Fig. 2 indicates that, *X*-filling only for unspecified bits in a test cube may not be enough to achieve a satisfactory effect in capture power reduction. In other words, capture power reduction must also be considered when determining specified bits in test cube generation.

Note that a target fault can often be detected by multiple test cubes, and that some of them have fewer capture transitions related to specified bits. In Fig. 2, for example, suppose that the test cube $v'' = \langle a1, a2, a3, a4, a5\rangle = \langle 0, X, 0, 1, X\rangle$ can also detect the fault *f*, and that the functional response to *v''* is $\langle b1, b2, b3, b4\rangle = \langle 1, 0, 1, 1\rangle$. In this case, specified bits in *v''* do not cause any capture transition.

Based on this observation, we propose a novel capture-aware or CA test cube generation algorithm to deterministically find such test cubes good for capture power reduction. Then, a new method for LCP test generation can be built by combining this algorithm with any *X*-filling technique for overall capture power reduction.

3. New Method for LCP Test Generation

3.1 Overview

The new method for LCP test generation is based on a two-pass flow as follows:

Pass-1: Conventional detection-oriented ATPG is used to generate a compact test set *T* with satisfactory fault coverage.
Pass-2: Any high-capture-power test vector *v* in *T* is identified and replaced with a new test vector *v''* such that $CT(v'') < c\_limit$, where *c_limit* is a pre-set limit.

Fig. 3 shows the general flow of the new method for LCP test generation. The initial test set *T* is generated by a conventional detection-oriented ATPG procedure ① in Pass-1. In Pass-2, first, *Ttar* is obtained in ② by selecting any vector *v* from *T* if $CT(v) > c\_limit$. Then, for each vector *v* in *Ttar*, a target fault list *Ftar*(*v*) is obtained in ③ such that no fault coverage loss occurs if all faults in *Ftar*(*v*) are detected. Following that, a capture-aware or CA test cube generation procedure ④ is repeated to generate a new test cube *v'* progressively to detect all faults in *Ftar*(*v*), with capture power reduction being taken into consideration. After that, *X*-bits in *v'* are filled with a low capture power or LCP *X*-filling procedure ⑤, and a new fully-specified test vector *v''* is obtained. Finally, a new test set *T'* is obtained in ⑥ by replacing the original test vector *v* in *T* with *v''*. Obviously, *T'* has the same fault coverage as *T* but test vectors in *T'* have low capture power.

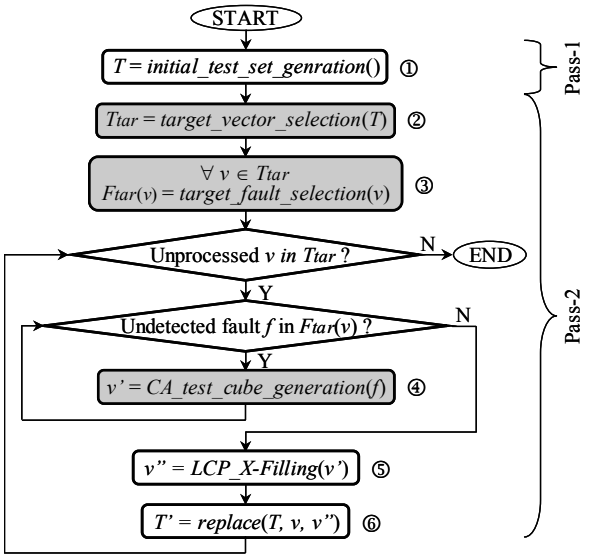


Fig. 3 Flow of the New Method for LCP Test Generation.

The procedures ② ~ ④ in Fig. 3, which are marked in gray, are unique to the new method for LCP test generation. The details of the procedures ② ~ ④ are presented in 3.2, 3.3, and 3.4, respectively.

3.2 Target Vector Selection

The *target_vector_selection*(*T*) procedure in Fig. 3 is used to identify the set of high-capture-power test vectors from *T*, and these vectors are stored in *Ttar*. Its purpose is to avoid wasting efforts in processing a test vector that already has low capture power.

Target vector selection is best based on power analysis, but this approach is time-consuming and layout information may not be available at that stage. Therefore, in this paper, we select a test vector *v* as a target vector if $CT(v) \geq c\_limit$. Generally, *c_limit* can be set by taking the power budget of a design into consideration or by using a heuristic approach as mentioned in Section 4.

3.3 Target Fault Selection

After the set of high-capture-power test vectors *Ttar* is identified, low-capture-power test vectors need to be generated to replace the test vectors in *Ttar*, one at a time. In order to generate a new vector to replace a vector *v* in *Ttar*, it is necessary to select a list of faults *Ftar*(*v*) to target so that no fault coverage loss occurs. Generally, this target fault selection should satisfy the following conditions:

(**a**) $\sum_{v \in Ttar} Ftar(v)$ should contain all faults that are only detected by test vectors in *Ttar*. This is to guarantee that no fault coverage loss occurs.

(**b**) *Ftar*(*v*) should contain faults that are easier to be detected with a test cube of low capture power.

(**c**) *Ftar*(*v*) should be made as small as possible.

The *target_fault_selection*(*v*) procedure in Fig. 3 is used to obtain the target fault list *Ftar*(*v*) that satisfies the three conditions. An example is shown in Fig. 4.

	HCP Vector?	Fault Classification	Target Fault List *Ftar*(*vi*)
*v*1	Y	(*f*1) (*f*6) [*f*9] [*f*10]	*f*1 *f*6 *f*9
*v*2	N	*f*2 *f*11	
*v*3	N	*f*3 *f*12	
*v*4	Y	(*f*4) (*f*7) (*f*8) [*f*9] *f*12	*f*4 *f*7 *f*8
*v*5	Y	(*f*5) [*f*10] *f*11	*f*5 *f*10

○: vector-essential fault □: set-essential fault

Fig. 4 Example of Target Fault Selection.

In Fig. 4, the original test set is T = {*v*1, *v*2, *v*3, *v*4, *v*5}. Suppose that *v*1, *v*4, and *v*5 are high-capture-power (HCP) test vectors. That is, *Ttar* = {*v*1, *v*4, *v*5}. There are 12 faults and the detection information obtained by fault simulation is also shown in Fig. 4. Note that *f*11 and *f*12 are also detected by *v*2 and *v*3, which are not in *Ttar*. That is, the set of faults that are only detected by test vectors in *Ttar* is *TA* = {*f*1, *f*4 ~ *f*10}. Obviously, no fault coverage loss will occur if all faults in *TA* are detected by a set of new test vectors.

All faults in *TA* can be classified into two groups: A *vector-essential fault* is detected by exactly one test vector in *Ttar* and by no other test vector in *T*. All faults marked in circles in Fig. 4 are vector-essential faults. On the other hand, a *set-essential fault* is detected by multiple test vectors in *Ttar* and by no other test vector in (*T* – *Ttar*). All faults marked in squares in Fig. 4 are set-essential faults.

All vector-essential faults of *v* should be included in *Ftar*(*v*). For example, *f*1 and *f*6 should be included in *Ftar*(*v*1). On the other hand, a set-essential fault of *v* can be included in *Ftar*(*v*) or in the target fault list of another test vector that also detects the fault, without any fault coverage loss. For example, *f*9 is a set-essential fault detected by *v*1 and *v*4. That is, *f*9 can be included in either *Ftar*(*v*1) or *Ftar*(*v*4).

The decision on whether to place a set-essential fault of *v* into the current *Ftar*(*v*) is made by checking how the easiness of detecting the faults in *Ftar*(*v*) with a test cube of low capture power is affected by the decision. In order to measure the easiness, we introduce a new heuristic concept as follows:

Definition 2: Let *fa* and *fb* be two faults in a full-scan circuit. Denote the sets of PPIs that are structurally reachable from *fa* and *fb* by *RI*(*a*) and *RI*(*b*), respectively. Denote the sets of PPOs that are structurally reachable from *fa* and *fb* by *RO*(*a*) and *RO*(*b*), respectively. The *overlapping degree* between *fa* and *fb*, denoted by *od*(*fa*, *fb*), is defined as follows:

$$od(fa, fb) = \sum_{i=a,b} \frac{|RI(a) \cap RI(b)|}{RI(i)} + \sum_{i=a,b} \frac{|RO(a) \cap RO(b)|}{RO(i)}$$

As illustrated in Fig. 5, the larger the value of *od*(*fa*, *fb*), the more *fa* and *fb* overlap at PPIs and PPOs. This indicates that it may be difficult to reduce capture transitions when generating a test cube to detect both *fa* and *fb*.

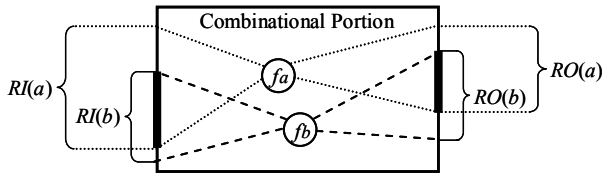


Fig. 5 Concept of Overlapping Degree.

Suppose that a set-essential fault *f* is detected by a test vector *v* whose current target fault list is *Ftar*(*v*) = {*fn*1, *fn*2, …, *fnp*}, where *f* ∉ *Ftar*(*v*). We first calculate *od*(*f*, *fn*1), *od*(*f*, *fn*2), …, and *od*(*f*, *fnp*), and then obtain the average overlapping degree as follows:

$$aod(f, Ftar(v)) = \sum_{i=1,2,\ldots,p} od(f, fn_i) / |Ftar(v)|$$

Now, in order to decide where to place a set-essential fault *f* that is detected by test vectors *vm*1, *vm*2, …, and *vms*, we calculate *aod*(*f*, *Ftar*(*vm*1)), *aod*(*f*, *Ftar*(*vm*2)), …, and *aod*(*f*, *Ftar*(*vms*)), and place *f* into the target fault list of the test vector that has the lowest average overlapping degree.

In Fig. 4, for example, at the time when we need to determine where to place the set-essential fault *f*9 (detected by *v*1 and *v*4), *Ftar*(*v*1) and *Ftar*(*v*4) are {*f*1, *f*6} and {*f*4, *f*7, *f*8}, respectively. Suppose that *aod*(*f*9, *Ftar*(*v*1)) < *aod*(*f*9, *Ftar*(*v*4)). In this case, *f*9 is placed into *Ftar*(*v*1). The final result of target fault selection is also shown in Fig. 4.

3.4 Capture-Aware Test Cube Generation

After the target fault list *Ftar*(*v*) is properly obtained for a high-capture-power test vector *v*, the next step is to generate a low-capture-power test cube to detect all faults in *Ftar*(*v*). The capture-aware or CA test cube generation procedure, *CA_test_cube_generation*(*f*) in Fig. 3, is used for this purpose. Its general flow is shown in Fig. 6.

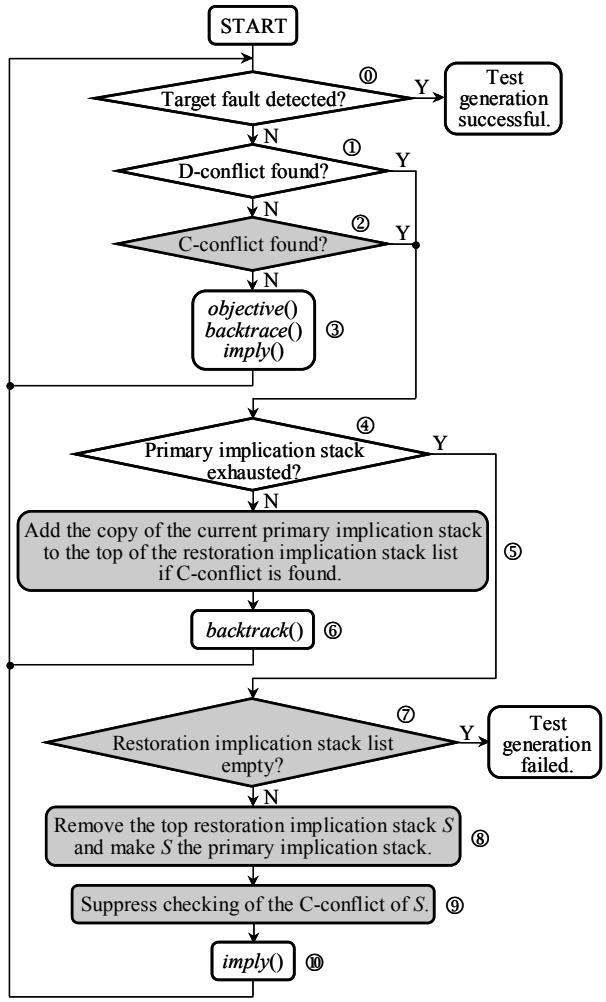


Fig. 6 General Flow of *CA_test_cube_generation*(*f*).

Generally, *CA_test_cube_generation*(*f*) is based on PODEM [12] and has a few enhancements marked in gray. These enhancements are based on two new concepts, *capture conflict* and *restoration implication stack*, that allow *CA_test_cube_generation*(*f*) to generate a test cube to detect fault *f*, and at the same time to reduce the number of capture transitions w.r.t. the specified bits in the test cube as much as possible. The details are as follows:

In a conventional PODEM-based test generation procedure, backtrack occurs only when *X*-path-checking finds a *detection conflict*, or *D-conflict* in short, that there is no path containing undetermined values between the gates of D-frontiers and any PO or PPO in order to be able to complete a sensitized path for fault detection [12].

In *CA_test_cube_generation*(*f*), we introduce a new backtrack condition, called *capture conflict*, or *C-conflict* in short, that a PPI and its corresponding PPO have opposite logic values, indicating a capture transition at a scan F/F. If there are *n* scan F/Fs, there are *n* C-conflicts, donated by C_1, C_2, …, C_n, as shown in Fig. 7, where C_i is the C-conflict at the PPI and PPO lines for the *i*-th scan F/F. In comparison, there is only one D-conflict for any failed *X*-path-check.

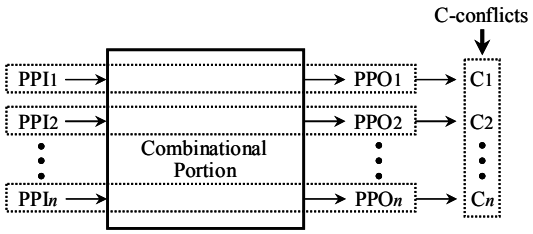


Fig. 7 C-Conflicts.

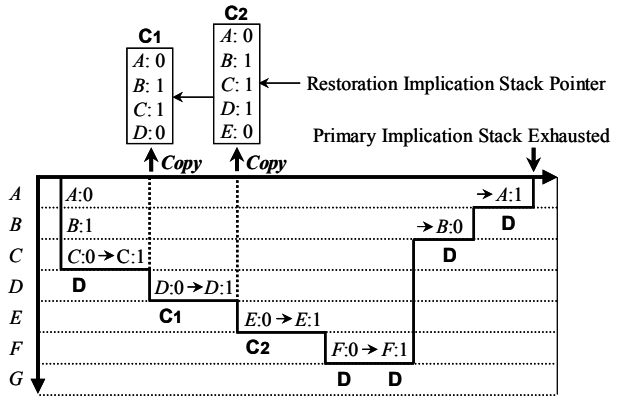
C-conflicts are checked in *CA_test_cube_generation*(*f*) in the order of their impacts on capture power dissipation. A simple heuristic to assess the impact of the C-conflict C_i is to count the number of gates in the combinational portion that are reachable from the output of the *i*-th scan F/F.

CA_test_cube_generation(*f*) backtracks in ⑥ when either a D-conflict in ① or a C-conflict in ② is found. However, a D-conflict and a C-conflict are fundamentally different for the following reasons: If the search space is exhausted only because of D-conflicts, test generation really fails. However, if at least one C-conflict occurs before the search space is exhausted, test generation may be made successful if the C-conflict is ignored. A test cube generated by ignoring a C-conflict can still detect the target fault, but cannot avoid a capture transition at the corresponding scan F/F.

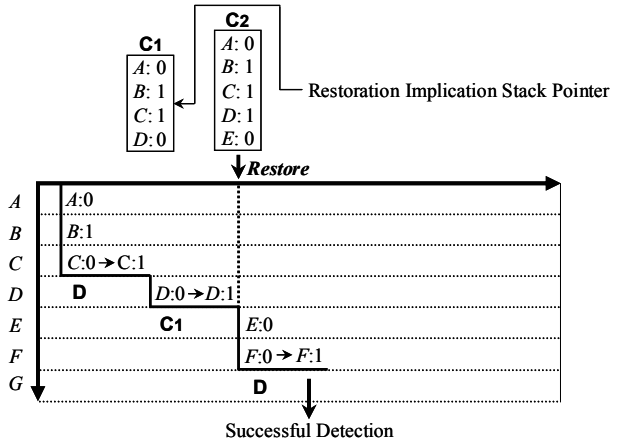
Obviously, it is beneficial to check for C-conflicts to reduce capture transitions but it is also necessary to prevent a C-conflict from blocking the generation of a test cube for detecting the target fault. Therefore, we introduce two types of implication stack: A *primary implication stack* is similar to what is used in a conventional PODEM-based ATPG procedure and it is used for managing the search space. A *restoration implication stack* is a copy of the primary implication stack obtained when a C-conflict is found. Since multiple C-conflicts may occur, there may exist multiple restoration implication stacks. These stacks are placed in a list, called a *restoration implication stack list*.

When the primary implication stack is exhausted in ④, one checks if the restoration implication stack list is empty in ⑦. A non-empty list means that at least one C-conflict occurred, contributing to the failing of the current test generation pass. In this case, the top or latest stack *S* in the restoration implication stack list is removed from the list and restored as the primary implication stack in ⑧. In addition, the C-conflict corresponding to the stack *S* is suppressed from further C-conflict checking in ⑨. Then, test generation is resumed. This way, a test cube, which detects the target fault and at the same time reduces the number of capture transitions as much as possible, can be generated.

An example of conducting test cube generation by *CA_test_cube_generation*(*f*) is shown in Fig. 8, where *A* through *G* are PPI lines. Suppose that *backtrace*() determines logic values for these lines during test cube generation in the search order of *A* to *G*. In addition, denote the primary implication stack by *PS*.



(a) Failed Test Generation due to the D-Conflict and C-Conflicts



(b) Successful Test Generation by Implication Stack Restoration

Fig. 8 Example of Capture-Aware Test Cube Generation.

As shown in Fig. 8 (a), when PS = <A: 0, B: 1, C: 0>, the D-conflict occurs, which is indicated by D. Backtracking brings logic 1 to C, and *backtrace*() further determines logic 0 for D. When PS = <A: 0, B: 1, C: 1, D: 0>, C-conflict C1 occurs. In this case, a copy of PS, denoted by C1, is placed into the restoration implication stack list. Backtracking brings logic 1 to D. Similarly, when PS = <A: 0, B: 1, C: 1, D: 1, E: 0>, C-conflict C2 occurs, and a copy of PS, denoted by C2, is placed into the restoration implication stack list. At the end, PS is exhausted due to a few more occurrences of the D-conflict.

In Fig. 8 (b), the top stack, C2, in the restoration implication stack list, is restored as the primary implication stack, and test generation is resumed with C-conflict C2 being suppressed. The resulting test cube is <A, B, C, D, E, F, G> = <0, 1, 1, 1, 0, 1, X>, which detects the target fault and avoids a capture transition corresponding to C-conflict C1.

4. Experimental Results

The new method for LCP test generation, whose flow is shown in Fig. 3, was implemented and experiments were conducted on ISCAS'89 circuits. The results are summarized in Table 1.

Table 1 Results of LCP Test Generation

Circuit	Fault Cov. (%)	Ori. Test Generation			LCP Test Generation		
		# of Vec.	*Max. Trans.*	*CPU* (Sec.)	*# of Vec.*	*Max. Trans.*	*CPU* (Sec.)
s1238	94.9	125	18	0.2	128	11	18.2
s1423	99.1	24	49	0.1	27	29	7.6
s5378	99.1	100	102	0.5	106	85	20.3
s9234	93.5	111	124	2.9	133	99	62.7
s13207	98.5	235	380	4.3	235	286	66.9
s15850	96.7	97	282	6.3	101	169	114.2
s35932	89.9	12	1548	36.1	13	922	114.4
s38417	99.5	87	590	78.0	91	491	224.9
s38584	95.9	114	925	54.9	121	459	479.5

In the experiments, *c_limit* was set as 50% of the maximum number of capture transitions of original test vectors. On average, the maximum number of capture transitions was reduced by **32.1**%, much higher than the **21.6**% obtained by *X*-filling only [8], at the cost of **7.1%** increase in the number of test vectors, due to the fact that multiple new test vectors might be generated to detect all target faults in $F_{tar}(v)$ for v. Using the original detection order will solve this issue, and its implementation is under way.

5. Conclusions

This paper proposed a novel algorithm for test cube generation not only for fault detection but also for capture power reduction, by introducing the concepts of capture conflict and implication stack restoration into ATPG. This algorithm, together with low-capture-power *X*-filling, leads to more effective reduction of capture-induced yield loss.

More evaluations are being planned to assess the effect of the proposed method directly through power analysis.

References

[1] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1994.

[2] L.-T. Wang, X. Wen, H. Furukawa, F. Hsu, S. Lin, S. Tsai, K. S. Abdel-Hafez, and S. Wu, "VirtualScan: A New Compressed Scan Technology for Test Cost Reduction," *Proc. Int'l Test Conf.*, pp. 916-925, 2004.

[3] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," *Proc. VLSI Test Symp.*, pp. 4-9, 1993.

[4] P. Girad, "Survey of Low-Power Testing of VLSI Circuits," *IEEE Design & Test of Computers*, Vol. 19, No. 3, pp. 82-92, 2002.

[5] T. Yoshida and M. Watari, "MD-Scan Method for Low Power Scan Testing," *Proc. Intl. Test Conf.*, pp. 480-487, 2003.

[6] W. Li, S. M. Reddy, and I. Pomeranz, "On Reducing Peak Current and Power During Test," *Proc. CSASV*, pp. 156-161, 2005.

[7] R. Sankaralingam and N. Touba, "Controlling Peak Power During Scan Testing," *Proc. VLSI Test Symp.*, pp. 153-159, 2002.

[8] X. Wen, H. Yamashita, S. Kajihara, L.-T. Wang, K. Saluja, and K. Kinoshita, "On Low-Capture-Power Test Generation for Scan Testing," *Proc. VLSI Test Symp.*, pp. 265-270, 2005.

[9] J. Silva, J. Monteiro, and K.A. Sakallah, "Test Pattern Generation for Circuit Using Power Management Techniques", *Proc. IEEE European Test Workshop*, 1997.

[10] K. Miyase and S. Kajihara, "XID: Don't Care Identification of Test Patterns for Combinational Circuits," *IEEE Trans. Computer-Aided Design*, Vol. 23, No. 2, pp. 321-326, 2004.

[11] J. Saxena, K. M. Butler, V. B. Jayaram, and S. Kundu, "A Case Study of IR-Drop in Structured At-Speed Testing," *Proc. Intl. Test Conf.*, pp. 1098-1104, 2003.

[12] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Computers*, Vol. 30, No. 3, pp. 215-222, 1981.