On Low-Capture-Power Test Generation for Scan Testing

Xiaoqing Wen [1], Yoshiyuki Yamashita [1], Seiji Kajihara [1], Laung-Terng Wang [2],
Kewal K. Saluja [3], and Kozo Kinoshita [4]

[1] Dept. of CSE, Kyushu Institute of Technology, Iizuka 820-8502, Japan, {wen, kajihara}@cse.kyutech.ac.jp
[2] SynTest Technologies, Inc., 505 S. Pastoria Ave., Suite 101, Sunnyvale, CA 94086, USA, wang@syntest.com
[3] Dept. of ECE, 1415 Engineering Drive, University of Wisconsin - Madison, Madison, WI 53706, USA, saluja@engr.wisc.edu
[4] Faculty of Informatics, Osaka Gakuin University, Suita 564-8511, Japan, kozo@utc.osaka-gu.ac.jp

Abstract

Research on low-power scan testing has been focused on the shift mode, with little or no consideration given to the capture mode power. However, high switching activity when capturing a test response can cause excessive IR drop, resulting in significant yield loss. This paper addresses this problem with a novel low-capture-power X-filling method by assigning 0's and 1's to unspecified (X) bits in a test cube to reduce the switching activity in capture mode. This method can be easily incorporated into any test generation flow, where test cubes are obtained during ATPG or by X-bit identification. Experimental results show the effectiveness of this method in reducing capture power dissipation without any impact on area, timing, and fault coverage.

1. Introduction

Integrated circuit testing based on the full-scan methodology and automatic test pattern generation (ATPG) is the most widely adopted test strategy that is well supported by test engineers, tool vendors, and tester makers. In a full-scan sequential circuit, scan flip-flops replace all functional flip-flops and operate in two modes: *shift* and *capture*. In shift mode, scan flip-flops are connected into shift registers or scan chains directly accessible from a tester. This mode is used to load a test vector through shift-in or observe a test response through shift-out, for the combinational portion of the sequential circuit. In capture mode, scan flip-flops operate as functional flip-flops and load the test response of the combinational portion to a test vector into themselves preparatory to shift-out later in shift mode. As a result, testing a full-scan sequential circuit is reduced to testing its combinational portion, in that now it is only necessary to generate test vectors for the combinational portion with a combinational ATPG program [1].

Despite its usefulness, the applicability of scan testing is increasingly being challenged due to the following three problems: *test data volume*, *test application time*, and *test power dissipation*. The first two problems are caused by larger gate and flip-flop counts, longer scan chains, and the use of complex fault models, all inevitable in the deep submicron (DSM) era. Several approaches, such as built-in self-test (BIST), test compaction, multi-capture clocking, and decompression-compression, have been proposed to address the problems of test data volume and test application time. In this paper, we focus on the test power dissipation problem.

The power dissipation of a CMOS circuit consists of *static* dissipation due to leakage current and *dynamic* dissipation due to switching activity, with the latter being dominant. Dynamic power dissipation in full-scan testing occurs in both shift mode and capture mode. In shift mode, a test vector is shifted into all scan chains of a full-scan circuit, one bit by one bit. This results in *shift power dissipation*. In capture mode, the test response of the combinational portion of the full-scan circuit to a test vector is loaded into all flip-flops, replacing the test vector that the flip-flops currently contain. This results in *capture power dissipation*, whenever the test vector and its corresponding test response have opposite logic values at some flip-flips.

Generally, test power dissipation, consisting of both shift and capture power dissipation, is significantly higher than functional power dissipation [2]. This is especially true for high-speed and high-density DSM integrated circuits with the system-on-a-chip (SoC) scheme [3]. Excessive test power dissipation may permanently damage a circuit under test, reduce its reliability due to accelerated electromigration, or result in yield loss due to faulted test results caused by IR drop [4]. Circuit damage and reliability degradation are mostly caused by excessive heat due to shift power dissipation, while significant yield loss can also be caused by excessive capture power dissipation.

Previous techniques for reducing test power dissipation have focused mostly on reducing shift power dissipation during test application, based on four major approaches: *scheduling*, *test vector manipulation*, *circuit modification*, and *scan chain modification*. Test scheduling [2, 5] takes the power budget into consideration when selecting modules to be tested simultaneously. Test vector manipulation includes power-aware ATPG [6, 7], static compaction [8], test vector modification [9], test vector reordering [10], test vector compression [11], and coding [12]. Circuit modification includes transition blocking [13] and clock gating [14]. Scan chain modification includes scan chain reordering [11, 15], scan chain partitioning [16], and scan chain modification [17]. Techniques tailored for BIST applications, such as toggle suppression [18] and low-power test pattern generation [19], have also been proposed.

In addition to shift power dissipation, capture power dissipation is also part of test power dissipation in scan testing, as illustrated in Fig. 1. In shift mode, test vector *A* is shifted

through all scan flip-flops in the scan chain with several hundreds to several thousands of clock cycles, depending on the scan chain length. In capture mode, test response *B* is loaded into scan flip-flops to replace the current test vector *A*. This is done in one or multiple clock cycles, depending on whether the multi-capture clocking scheme is used.

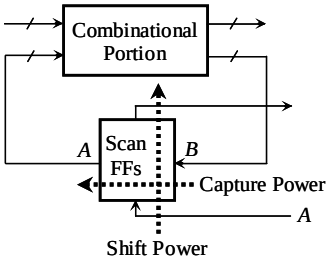

Fig. 1 Two Types of Test Power Dissipation.

Although capture power dissipation has less impact on the total heat dissipation than shift power dissipation, it may nonetheless cause significant yield loss as explained below. High switching activity in capture mode at the scan flip-flops due to the difference between *A* and *B* may result in instantaneously excessive IR drop, causing a faulted test response $B' \neq B$ to be loaded into the scan flip-flops. This results in yield loss, even though the excessive capture power dissipation does not cause too much heat dissipation.

For example, in one recently reported case, a 3M-gate industrial circuit passed all functional tests and all scan chain flush tests but showed un-repeatable behaviors only in capture mode during scan testing. Detailed analysis revealed that the circuit had multiple functional clocks, each driving a portion of the circuit; but only one test clock was used to drive all flip-flops in the circuit during scan testing. IR drop caused by high switching activity due to many flip-flops operating simultaneously was the reason for the yield loss.

The above explanation and example suggest that it is not sufficient to reduce only shift power dissipation. Capture power dissipation should also be reduced, especially in order to avoid yield loss caused by faulted test results. The ultimate solution for this is to reduce the number of flip-flops that can operate simultaneously. For a single-clock circuit, this can be achieved by selective clock gating. However, its impact on physical design is high. For a multiple-clock circuit, this can be achieved by either the one-hot or the multi-capture clocking scheme. However, the former suffers from large test data volume and the latter suffers from complicated ATPG with high memory consumption as well as the need of controlling multiple test clocks. These disadvantages motivated us to propose a new solution for reducing capture power dissipation, which should be simple, effective, and of no impact on physical and test design flows.

We notice the fact that many *test cubes*, i.e., test vectors with unspecified bits (*X*-bits), are usually generated either during ATPG [1] or obtained by *X*-bit identification [20] from a set of fully-specified test vectors. In this paper, we propose a *low-capture-power* (*LCP*) *X*-filling method for assigning 0's and 1's to the *X*-bits in a test cube so that the number of transitions at the outputs of scan flip-flops in capture mode for the resulting fully-specified test vector is reduced. Test vectors obtained by the LCP method have low capture power dissipation, resulting in reduced yield loss caused by faulted capture operations. As a totally software-based solution, the LCP method has no physical design impact and can be easily incorporated into any test generation flow.

The rest of the paper is organized as follows. Section 2 describes the research background. Section 3 presents the LCP *X*-filling method. Section 4 shows experimental results, and Section 5 concludes the paper.

2. Background

2.1 Test Cube Handling

A general ATPG procedure repeats the operations of selecting an undetected fault to produce a test vector for it. The result is usually a *test cube* with unspecified bits (*X*-bits). This test cube can be processed either immediately after its generation as in *dynamic compaction* or together with other test cubes in a post-ATPG operation as in *static compaction*, for the purpose of reducing the test set size or test power dissipation [1].

Note that test cubes processed in static compaction can also be obtained by *X*-bit identification [20] from a set of fully-specified test vectors. It has been shown that a significant percentage of bits, as high as 90% in some cases, in a fully-specified test vector set can be turned into *X*-bits without affecting its fault coverage.

The fundamental operation in test cube handling during dynamic compaction or static compaction is to determine 0's and 1's for the *X*-bits in a test cube. This operation is called *X-filling* in this paper. Obviously, different *X*-filling methods have different impact on test data volume, test application time, and test power.

2.2 Previous *X*-Filling Methods

Generally, there are three approaches to *X*-filling: *random*, *algorithmic*, and *merge-based*. Random *X*-filling assigns 0's and 1's randomly to *X*-bits in a test cube. Algorithmic *X*-filling determines logic values for the *X*-bits in a test cube in a more sophisticated way in order to better achieve a specific goal. Merge-based *X*-filling determines the logic value for an *X*-bit in a test cube depending on the logic value of the corresponding bit in another test cube to be merged with. For example, merging test cube t_1 1*X*0 with test cube t_2 11*X* will cause assigning 1 to the *X*-bit in t_1 and 0 to the *X*-bit in t_2, resulting in one test vector 110.

Algorithmic *X*-filling is often used in dynamic compaction for reducing the number of final test vectors [1]. The key issue is how to select a secondary target fault which has higher chances of being detected with the *X*-bits in a test cube. Selection methods based on fault simulation by criti-

cal path tracing, independent faults, etc. have been shown to be effective. Algorithmic *X*-filling is also used for reducing shift power dissipation by properly re-assigning 0's and 1's to the *X*-bits found by *X*-bit identification [9].

Merge-based *X*-filling is often used in static compaction for reducing the number of test vectors [1, 22] as well as for shift power reduction by carefully selecting the order of test cubes to be merged by using a cost function reflecting shift transition activity [8].

Random *X*-filling is conducted for remaining *X*-bits after algorithmic or merge-based *X*-filling is done. Its purpose is to reduce the number of test vectors since randomly assigning 0' and 1's to the *X*-bits in a test cube often increases the chances of detecting additional faults [1]. However, random *X*-filling usually adversely affects test power dissipation [8].

2.3 Motivation

Previous *X*-filling methods are largely used for reducing the number of test vectors [1, 21, 22], and there are a few *X*-filling methods available for shift power reduction [8, 9]. However, there is no *X*-filling method for capture power reduction yet. This is a serious problem as it leaves a significant yield loss factor totally uncontained.

To solve this problem, we propose a novel algorithmic *X*-filling method, called the *LCP (Low-Capture-Power) X-filling method*, for reducing the number of transitions at the outputs of scan flip-flops in capture mode. Test vectors generated with this method have low capture power dissipation, resulting in less yield loss caused by faulted capture operations. This method can be used in both dynamic compaction and static compaction, for test cubes generated during ATPG or obtained by *X*-bit identification. In addition, this method can be used with other *X*-filling methods to achieve a balanced reduction effect among the number of test vectors, shift power dissipation, and capture power dissipation.

3. LCP *X*-Filling

3.1 Problem Formalization

A general full-scan circuit is shown in Fig. 2. It consists of a combinational portion with m_1 primary inputs (PIs) and m_2 primary outputs (POs) as well as n scan flip-flops (FFs). The outputs of the scan FFs that feed the combinational portion are pseudo primary inputs (PPIs) and the functional inputs from the combinational portion to the scan FFs are pseudo primary outputs (PPOs). Note that the number of PPIs is the same as that of PPOs, while the number of PIs may or may not be the same as that of POs. Also note that, for the convenience of presentation, all scan FFs are assumed to form one scan chain with SI as the scan input and SO as the scan output. The *X*-filling method to be presented in the following, however, can be readily extended for any full-scan circuit with multiple scan chains.

In Fig. 2, v is a test cube with at least one *X*-bit. The PI and PPI bits in v are denoted by an m_1-bit vector $\langle v$: PI$\rangle$ and an n-bit vector $\langle v$: PPI$\rangle$, respectively. The combinational portion is assumed to have logic function f, and its functional response to v is $f(v)$. The PO and PPO bits in $f(v)$ are denoted by an m_2-bit vector $\langle f(v)$: PO$\rangle$ and an n-bit vector $\langle f(v)$: PPO$\rangle$, respectively.

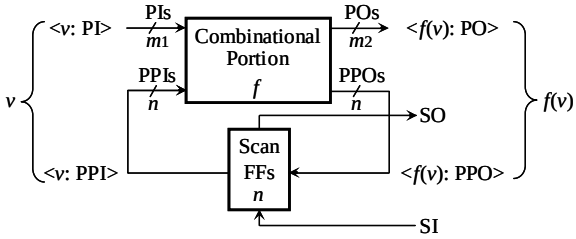

Fig. 2 A General Full-Scan Circuit

If $\langle v$: PPI$\rangle$ and $\langle f(v)$: PPO$\rangle$ are fully-specified, the result of their bit-wise exclusive-OR operation is an n-bit vector, denoted by $\langle v$: PPI$\rangle \oplus \langle f(v)$: PPO$\rangle$. Obviously, if the corresponding bits in $\langle v$: PPI$\rangle$ and $\langle f(v)$: PPO$\rangle$ are different as shown in Fig. 3, a transition, called *capture transition* in this paper, will occur at the output of the scan FF in capture mode. Obviously, the number of 1's in $\langle v$: PPI$\rangle \oplus \langle f(v)$: PPO$\rangle$, denoted by $|\langle v$: PPI$\rangle \oplus \langle f(v)$: PPO$\rangle|$, is the total number of capture transitions for v.

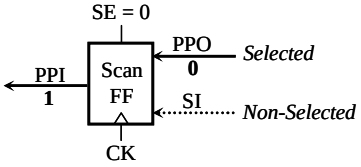

Fig. 3 Capture Transition at a Scan Flip-Flop.

Since the number of capture transitions is closely correlated with the circuit switching activity as demonstrated in [8], the LCP *X*-filling problem can be formalized as follows:

LCP X-Filling Problem: Given a test cube v for a full-scan circuit with combinational logic function f, assign 0's and 1's to the *X*-bits in v such that $|\langle v$: PPI$\rangle \oplus \langle f(v)$: PPO$\rangle|$ is minimized.

3.2 *X*-Filling Algorithm

Suppose that v is a test cube with at least one *X*-bit and $f(v)$ is the simulated response of the combinational portion of a full-scan circuit to v. Note that $f(v)$ may also have *X*-bits due to the *X*-bits in v. Depending on the appearance of *X*-bits in $\langle v$: PPI$\rangle$ and $\langle f(v)$: PPO$\rangle$, we define four *X-cases* as shown in Table 1:

Table 1 *X*-Cases

	$\langle f(v)$: PPO$\rangle$	
$\langle v$: PPI$\rangle$	without *X*	with *X*
without *X*	Case-1	Case-3
with *X*	Case-2	Case-4

The algorithm for LCP *X*-filling in each *X*-case is presented next.

3.2.1 Case-1

In Case-1, since <v: PPI> and <$f(v)$: PPO> have no X-bits, |<v: PPI> ⊕ <$f(v)$: PPO>|, the total number of capture transitions, is already determined and cannot be changed irrespective of the logic values assigned to the X bits in <v: PI>.

Since v is a test cube with at least one X-bit and <v: PPI> has no X-bits, <v: PI> must have at least one X-bit. Therefore, X-filling in Case-1 can be targeted for any other purpose, such as reducing the number of test vectors or shift power dissipation, with X-filling methods mentioned in 2.2.

3.2.2 Case-2

In Case-2, since <v: PPI> has at least one X-bit, X-filling is first conducted for <v: PPI> to reduce the number of capture transitions. This is achieved by replacing all X-bits in <v: PPI> with the same logic values at the corresponding bits in <$f(v)$: PPO>, which has no X-bit. After this assignment is done, Case-2 reduces to Case-1 since <v: PPI> no longer has any X-bit. Then Case-1 X-filling can be conducted for all the remaining X-bits in <v: PI>.

An example is shown in Fig. 4, where <v: PPI> = <$X0X1$> and <$f(v)$: PPO> = <0010>. First, 0 and 1 are assigned to the first X-bit and the second X-bit, respectively, in <v: PPI>. Then, only one X-bit remains in <v: PI>, which is handled by Case-1 X-filling.

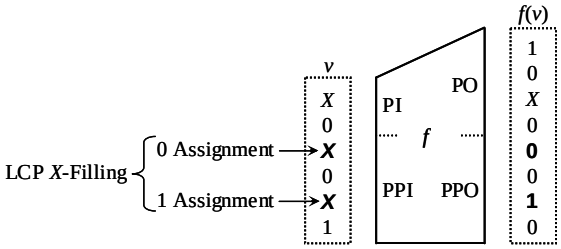

Fig. 4 Assignment-Based *X*-Filling.

3.2.3 Case-3

In Case-3, <v: PI> has at least one X-bit since <v: PPI> has no X-bit. In addition, <$f(v)$: PPO> has at least one X-bit. X-filling for the X-bits in <v: PI> is conducted in such a way that as many X-bits as possible in <$f(v)$: PPO> can have the same values as the corresponding bits in <v: PPI>, in order to reduce the number of capture transitions.

Whether an X-bit a in <$f(v)$: PPO> can have the same value as its corresponding bit b in <v: PPI> is determined by justification. For example, if b is 1, then one can try to justify 1 on a. If this is successful, 1 is placed on a; otherwise, 0 is placed on a. Note that, during justification, the logic values for some X-bits in <v: PI> will be determined.

An example is shown in Fig. 5, where <v: PPI> = <1011> and <$f(v)$: PPO> = <X010>. Obviously, placing 1 to the X-bit in <$f(v)$: PPO> reduces the number of capture transitions. Thus, justification of 1 on the X-bit in <$f(v)$: PPO> is conducted. Suppose that this is successful if 0 is assigned to the X-bit in <v: PI>. As a result, a fully-specified test vector v = <001011> is obtained and its simulated response is $f(v)$ = <10101010>.

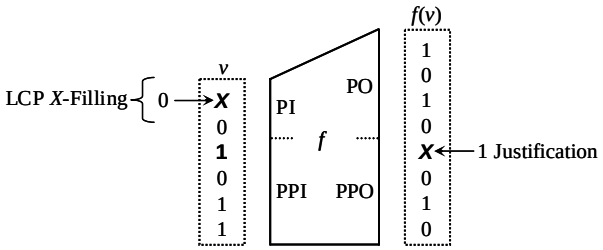

Fig. 5 Justification-Based *X*-Filling.

It is possible that <$f(v)$: PPO> has multiple X-bits. In this case, the order of the X-bits being justified affects the success ratio of justification; hence the number of reduced capture transitions. We propose the following criterion for selecting an X-bit, based on the easiness of justification:

Criterion-1: Suppose that a_1 and a_2 are two X-bits in <$f(v)$: PPO>. We obtain the sets of X-bits in <v: PI>, denoted by $X(a_1)$ and $X(a_2)$, that can be reached from a_1 and a_2, respectively. If $|X(a_1)| > |X(a_2)|$, a_1 is selected for justification since more X-bits are available for justifying a logic value on a_1. If $|X(a_1)| = |X(a_2)|$, we further obtain the average levels of all PIs with X-bits in $X(a_1)$ and $X(a_2)$, denoted by $L(a_1)$ and $L(a_2)$, respectively. Note that levels are assigned to all lines in a circuit from POs and PPOs. If $L(a_1) < L(a_2)$, a_1 is selected for justification since the PIs with X-bits in $X(a_1)$ are closer to the justification target of a_1.

Once all X-bits in <$f(v)$: PPO> are determined, Case-3 becomes Case-1. Then, Case-1 X-filling can be conducted for all the remaining X-bits in <v: PI>.

3.2.4 Case-4

In Case-4, both <v: PPI> and <$f(v)$: PPO> have X-bits. For a bit-pair <a, b> consisting of a bit a in <v: PPI> and its corresponding bit b in <$f(v)$: PPO>, there are four possible bit-pair types as summarized in Table 2:

Table 2 Bit-Pair Types

	a in <v: PPI>	b in <$f(v)$: PPO>
Type-A	0 or 1	0 or 1
Type-B	X	0 or 1
Type-C	0 or 1	X
Type-D	X	X

Obviously, there is no need to consider any Type-A bit-pair. For other bit-pairs with at least one X, we process Type-B and Type-C bit-pairs first. Only when there are no more such bit-pairs, we move on to process Type-D bit-pairs.

If both Type-B and Type-C bit-pairs exist, it is necessary to determine which type of bit-pairs to process first. Note that an X-bit in <v: PPI> for a Type-B bit-pair indicates that a capture transition can be avoided if a proper logic value is assigned to the X-bit. Also note that an X-bit in <$f(v)$: PPO> for a Type-C bit-pair indicates that a proper logic value may

be successfully justified on the *X*-bit so that a capture transition can be avoided. Therefore, we propose the following selection criterion for bit-pair selection:

Criterion-2: We compare the number of *X*-bits in <*v*: PPI> for all Type-B bit-pairs and the number of *X*-bits in <*f*(*v*): PPO> for all Type-C bit-pairs. If the former is larger than the latter, all Type-B bit-pairs are processed first with the Case-2 *X*-filling algorithm described in 3.2.2. If the latter is larger than the former, all Type-C bit-pairs are processed first with the Case-3 *X*-filling algorithm described in 3.2.3.

After *X*-filling for all Type-B and Type-C bit-pairs are conducted, it is possible that Type-D bit-pairs still remain. Suppose that <*a*, *b*> is such a bit-pair, where *a* in <*v*: PPI> and *b* in <*f*(*v*): PPO> both have *X*. In this case, we first check if 0 (1) can be assigned to both *a* and *b* in order to avoid a capture transition. This can be conducted by placing 0 (1) on *a* and trying to justify 0 (1) on *b*. If this is successful, we use 0 (1) for both *a* and *b*; otherwise, we use different values for *a* and *b*.

It is possible that there are multiple Type-D bit-pairs. In this case, we consider the multiple *X*-bits in <*f*(*v*): PPO> and use the Criterion-1 proposed for Case-3 *X*-filling to determine the order of processing Type-D bit-pairs.

An example for Type-D is shown in Fig. 6, where <*v*: PPI> = <1*X*11> and <*f*(*v*): PPO> = <1*X*10>. In this case, we try placing 0 on the *X*-bit in <*v*: PPI> and justifying 0 on the *X*-bit in <*f*(*v*): PPO>. Suppose that this is successful if 1 is assigned to the *X*-bit in <*v*: PI>. As a result, a fully-specified test vector *v* = <101011> is obtained and its simulated response is *f*(*v*) = <10101010>. Note that one capture transition is avoided in this case.

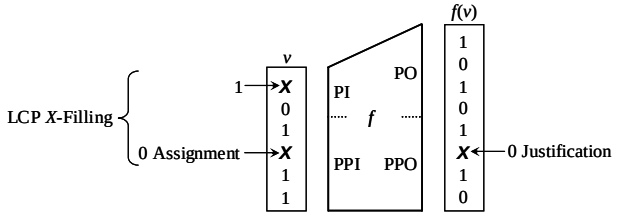

Fig. 6 Assignment-Justification-Based *X*-Filling.

After Type-B, Type-C, and Type-D bit-pairs are processed, Case-4 becomes Case-1 since both <*v*: PPI> and <*f*(*v*): PPO> no longer have any *X*-bit. Then, Case-1 *X*-filling can be conducted for all the remaining *X*-bits in <*v*: PI>.

3.3 *X*-Filling Procedure

The general procedure for LCP *X*-filling is illustrated in Fig. 7. A test cube *v* is processed based on its case type. For a Case-4 test cube, its bit-pairs for <*v*: PPI> and <*f*(*v*): PPO> will be further checked. If Type-B or Type-C bit-pairs exist, they should be processed as in Case-2 or Case-3. If there are only Type-D bit-pairs, assignment-justification will be conducted for *X*-filling. The final result of this procedure is a fully-specified test vector.

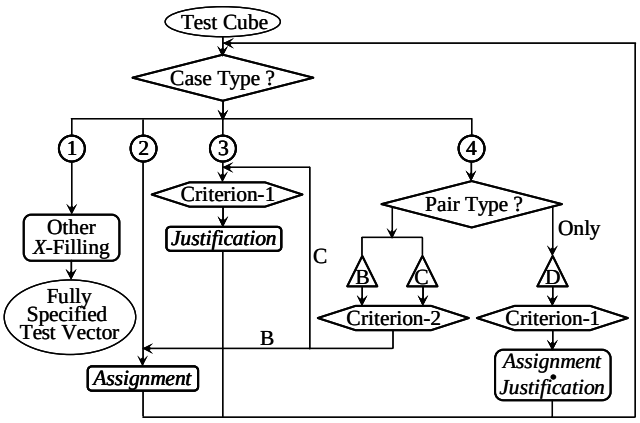

Fig. 7 LCP *X*-Filling Procedure.

4. Experimental Results

X-filling experiments were conducted on ISCAS'89 circuits. Since the major process was logic simulation, the total run time for these circuits was very short and is not reported here.

4.1 Dynamic *X*-Filling Results

Table 3 shows the results obtained by random *X*-filling and LCP *X*-filling for test cubes generated in ATPG. In ATPG, a test cube was generated for a primary fault. After that, the *X*-bits in the test cube were used to detect a secondary fault. This process was repeated until the number of detected secondary faults reached a threshold, denoted by *Limit*. Then, the remaining *X*-bits in the test cube were filled randomly or with the LCP method. In Table 3, the number of test vectors, the average number of node transitions per test vector, and the maximum number of node transitions for each case in capture mode are shown under "*# of Vec.*", "*Ave. Trans.*", and "*Max. Trans.*", respectively.

Table 3 Results for Dynamic *X*-Filling

Circuit	Fault Cov.	Random			LCP					
		No Limit			No Limit			Limit = 100		
		# of Vec.	Ave. Trans.	Max. Trans.	# of Vec.	Ave. Trans.	Max. Trans.	# of Vec.	Ave. Trans.	Max. Trans.
s1196	100	130	27.0	57	126	6.8	46	129	6.2	46
s1238	94.91	141	27.0	62	139	6.0	43	146	6.0	43
s1423	99.08	36	164.5	275	37	135.1	255	45	139.6	268
s5378	99.13	113	938.0	1133	112	412.6	1099	115	407.4	977
s9234	93.48	138	1647	2245	141	972.8	2206	145	1000	2109
s13207	98.46	262	2097	2757	263	763.7	2126	262	793	2220
s15850	96.68	132	2077	3212	124	977.0	2340	119	986	2457
s35932	89.91	18	6638	8255	18	4242	8255	38	2993	5797
s38417	99.47	102	6543	8257	102	4397	7679	118	4388	6517
s38584	95.85	124	3752	7289	124	2243	6287	135	2240	4525

Table 3 shows that on average, LCP *X*-filling (*Limit* = ∞) achieved 49.3% reduction for the average number of node transitions and 13.3% reduction for the maximum number of node transitions, compared with random *X*-filling.

Note that the smaller the value of *Limit*, the more remaining *X*-bits in a test cube, thus the higher node transition reduction effect achieved by LCP *X*-filling. However, the smaller

the value of *Limit*, the larger the number of test vectors. These contradicting trends were verified by experimenting with three largest ISCAS'89 benchmark circuits, as shown in Fig. 8. It is clear that a "good" value exists for *Limit*, which can balance the node transition reduction effect and the number of test vectors. In the case of Fig. 8, for example, 100 is obviously such a value for *Limit*.

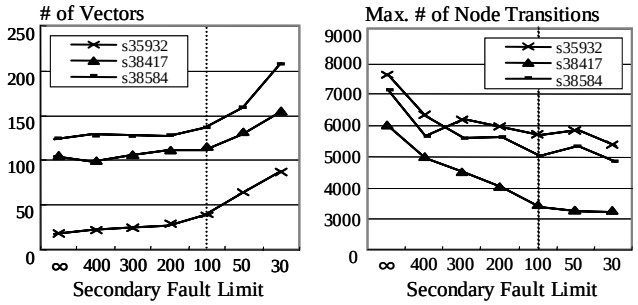

Fig. 8 Impact of Secondary Fault Limit.

The experimental results for LCP *X*-filling (*Limit* = 100) are also shown in Table 3. It can be seen that on average, LCP *X*-filling (*Limit* = 100) can achieve 53.7% more reduction for the maximum number of node transitions, compared with LCP *X*-filling (*Limit* = ∞), at the cost of 16.6% more test vectors.

4.2 Static *X*-Filling Results

Table 4 shows the results obtained by random *X*-filling and LCP *X*-filling for test cubes obtained by an *X*-bit identification procedure [20] from a set of fully-specified test vectors. As shown in Table 4, even with compacted test vectors an average of 64.5% of all bits in a set of fully-specified test vectors could be turned into *X*-bits without affecting its fault coverage. These *X*-bits were then filled randomly or with the LCP method. In Table 4, *X* (%) shows the percentage of *X*-bits identified from a set of fully-specified test vectors, while all other items have the same meaning as in Table 3.

Table 4 Results for Static *X*-Filling

Circuit	*# of Vec.*	*Fault Cov.* (%)	*X* (%)	Random		LCP	
				Ave. Trans.	*Max. Trans.*	*Ave. Trans.*	*Max. Trans.*
s1196	113	100	55.06	8.70	14	1.62	10
s1238	125	94.91	54.98	9.24	14	1.78	9
s1423	24	99.08	41.11	26.5	43	20.63	34
s5378	100	99.13	71.03	90.6	108	40.82	91
s9234	111	93.48	67.17	80.24	110	34.36	61
s13207	235	98.46	91.61	245.6	333	74.27	244
s15850	97	96.68	76.14	181.0	252	66.78	173
s35932	12	89.91	34.35	817.3	1533	569.0	1517
s38417	87	99.47	73.40	491.2	592	177.1	323
s38584	114	95.85	79.65	424.3	785	193.9	437

Table 4 shows that on average, LCP *X*-filling achieved 45.2% reduction for the average number of node transitions and 21.6% reduction for the maximum number of node transitions, compared with random *X*-filling.

5. Conclusions

This paper addressed a new test power reduction problem, i.e. reducing capture power dissipation to avoid yield loss caused by faulted test responses in capture mode. A novel low-capture-power (LCP) *X*-filling method has been proposed for assigning 0's and 1's to unspecified (*X*) bits in a test cube in order to reduce the switching activity at FFs and in the circuit for the resulting fully-specified test vector. Experimental results have shown its effectiveness.

More evaluations are under way to assess the effect of the LCP *X*-filling method directly through power consumption instead of switching activity at flip-flops. Extension of the LCP method to a double-capture scheme is also under way.

References

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

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

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

[4] T. Yoshida and M. Watari, "A New Approach for Low Power Scan Testing," *Proc. Intl. Test Conf.*, pp. 480-487, 2003.

[5] R. Chou, K. Saluja, and V. Agrawal, "Scheduling Tests for VLSI Systems under Power Constraints," *IEEE Trans. on VLSI Systems*, vol. 5, no. 6, pp. 175-185, 1997.

[6] S. Wang and S. Gupta, "ATPG for Heat Dissipation Minimization during Test Application," *IEEE Trans. on Computers*, vol. 47, no. 2, pp. 256-262, 1998.

[7] F. Corno, P. Prinetto, M. Redaudengo, and M. Reorda, "A Test Pattern Generation Methodology for Low Power Consumption," *Proc. VLSI Test Symp.*, pp. 35-40, 2000.

[8] R. Sankaralingam, R. Oruganti, and N. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," *Proc. VLSI Test Symp.*, pp. 35-40, 2000.

[9] S. Kajihara, K. Ishida, and K. Miyase, "Test Vector Modification for Power Reduction during Scan Testing," *Proc. VLSI Test Symp.*, pp. 160-165, 2002.

[10] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits during Test Application," *IEEE Trans. on Computer-Aided Design*, vol. 17, no. 12, pp. 1325-1333, 1998.

[11] A. Chandra and K. Chakrabarty, "Combining Low Power Scan testing and Test Data Compression for System-on-a-Chip," *Proc. Design Automation Conf.*, pp. 166-169, 2001.

[12] A. Chandra and K. Chakrabarty, "Reduction of SoC Test Data Volume, Scan Power and Testing Time Using Alternating Run-Length Codes," *Proc. Intl. Conf. on Computer Aided Design*, pp. 673-678, 2002.

[13] A. Hertwig and H. Wunderlich, "Low Power Serial Built-In Self-Test," *Proc. European Test Workshop*, pp. 49-53, 1998.

[14] R. Sankaralingam, R. Oruganti, and N. Touba, "Reducing Power Dissipation during Test Using Scan Chain Disable," *Proc. VLSI Test Symp.*, pp. 319-324, 2001.

[15] Y. Bonhomme, P. Girard, C. Landrault, and S. Pravossoudovitch "Power Driven Chaining of Flip-Flops in Scan Architectures," *Proc. Intl. Test Conf.*, pp. 796-803, 2002.

[16] J. Saxena, K. Butler, and L. Whetsel, "A Scheme to Reduce Power Consumption during Scan Testing," *Proc. Intl. Test Conf.*, pp. 670-677, 2001.

[17] O. Sinanoglu and A. Orailoglu, "Scan Power Minimization through Stimulus and Response Transformations," *Proc. Design, Automation and Test in Europe*, pp. 404-409, 2004.

[18] S. Gerstendoerfer and H. Wunderlich, "Minimized Power Consumption for Scan-Based BIST," *Proc. Intl. Test Conf.*, pp. 77-84, 1999.

[19] S. Wang, "Generation of Low-Power-Dissipation and High-Fault Coverage Patterns for Scan-Based BIST," *Proc. Intl. Test Conf.*, pp. 834-843, 2002.

[20] 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, Feb. 2004.

[21] S. Kajihara, K. Taniguchi, K. Miyase, I. Pomeranz, S. Reddy, "Test Data Compression Using Don't-Care Identification and Statistical Encoding," *Proc. Asian Test Symp.*, pp. 67-72, 2002.

[22] X. Lin, J. Rajski, I. Pomeranz, S. M. Reddy, "On Static Test Compaction and Test Pattern Ordering for Scan Designs," *Proc. Intl. Test Conf.*, pp. 1088-1097, 2001.