A Novel Scheme to Reduce Power Supply Noise for High-Quality At-Speed Scan Testing
Xiaoqing Wen, Kohei Miyase, Seiji Kajihara, Tatsuya Suzuki, Yuta Yamato, Patrick Girard, Yuji Ohsumi, Laung-Terng Wang

To cite this version:
A Novel Scheme to Reduce Power Supply Noise for High-Quality At-Speed Scan Testing

Xiaoqing Wen 1, Kohei Miyase 1, Seiji Kajihara 1, Tatsuya Suzuki 1, Yuta Yamato 1, Patrick Girard 2, Yuji Ohsumi 3, and Laung-Terng Wang 4

1 Kyushu Institute of Technology, Iizuka 820-8502, Japan
2 LIRMM, 161 rue Ada, 34392 Montpellier cedex 05, France
3 Hibikino R&D Center, DNP Co. Ltd., Kitakyushu 808-0135, Japan
4 SynTest Technologies, Inc., 505 S. Pastoria Ave., Sunnyvale, CA 94086, USA

Abstract

High-quality at-speed scan testing, characterized by high small-delay-defect detecting capability, is indispensable to achieve high delay test quality for DSM circuits. However, such testing is susceptible to yield loss due to excessive power supply noise caused by high launch-induced switching activity. This paper addresses this serious problem with a novel and practical post-ATPG X-filling scheme, featuring (1) a test relaxation method, called path keeping X-identification, that finds don’t-care bits from a fully-specified transition delay test set while preserving its delay test quality by keeping the longest paths originally sensitized for fault detection, and (2) an X-filling method, called justification-probability-based fill (JP-fill), that is both effective and scalable for reducing launch-induced switching activity. This scheme can be easily implemented into any ATPG flow to effectively reduce power supply noise, without any impact on delay test quality, test data volume, area overhead, and circuit timing.

1. Introduction

Shrinking feature size, growing circuit complexity, increasing clock speed, and decreasing power supply voltage have made timing-related defects, usually of small delays, the dominant failure mechanism in the deep submicron (DSM) era [1]. As a result, high-quality delay testing, characterized by high small-delay-defect detecting capability, is required in order to reduce the defect level of DSM circuits [2].

1.1 At-Speed Scan Testing

Delay testing is mostly conducted by at-speed scan testing [3]. This is due to its strong fault diagnosis support, easy implementation, and high fault coverage.

As illustrated in Fig. 1 (a), the essence of at-speed scan testing is to launch a transition at the start-point of a path (FF output) and capture its response at the end-point of the path (FF input) at the system speed. That is, the test cycle (T) is at-speed in order to directly check the delay of the path. As shown in Fig. 1 (b), a path may have three types of possible delays: nominal path delay (ND), defect-induced delay (DD), and power-supply-noise-induced delay (PD). A chip is identified to be defective if \( ND + DD + PD > T \), that will cause an incorrect logic value to be loaded into the end-point FF due to the unsatisfied timing requirement.

The launch of transitions in at-speed scan testing can be realized by either the launch-off-shift (LOS) through a shift pulse (\( SE = 1 \)) or the launch-off-capture (LOC) scheme through a capture pulse (\( SE = 0 \)) [3]. This paper focuses on LOC as shown in Fig. 2, since it is widely applied due to the use of a slow \( SE \) signal and conventional scan FFs.

1.2 High-Quality At-Speed Scan Testing

The quality of at-speed testing is measured by defect level (the fraction of defective chips passing a test), which depends on test vectors and test timing [4], as illustrated in Fig. 3.

Fig. 1 At-Speed Scan Testing.

Fig. 2 Launch-off-Capture (LOC) Scheme.
Another approach for reducing higher than that of functional operation [11-13]. This activity caused by the launch operation (Fig. 1 (a)) is much long paths for fault detection. This can be directly achieved of sensitized paths by biasing ATPG towards sensitizing However, in that it is proportional to the number of nodes in a circuit. model [3] is widely used since its fault count is manageable is prohibitively high. In practice, the transition delay fault test escape occurs since the defect can not be detected by needed in order to sensitize as long paths as possible in transition delay fault detection, for the purpose of achieving timing-aware transition delay test generation is needed in order to sensitize as long paths as possible in transition delay fault detection, for the purpose of achieving high small-delay-defect detecting capability [2, 7-9].

From Fig. 3, it is also clear that the delay test quality of a transition delay test set for at-speed scan testing depends on the set of longest paths sensitized for fault detection [2, 4, 9]. In this paper, this set of paths is called the characteristic path set (CPS) of the transition delay test set.

1.3 Power Supply Noise

When switching activity occurs in a circuit, current flows through the equivalent RLC network, resulting in IR and L-di/dt voltage drop at logic cells. This is called power supply noise [10], which has severe impact on high-quality at-speed scan testing due to the following factors:

- High Launch-Induced Switching Activity: The switching activity caused by the launch operation (Fig. 1 (a)) is much higher than that of functional operation [11-13]. This results in voltage drop at cells, increasing cell delays and thus path delay. This means larger PD in Fig. 1 (b).
- Long Path Sensitization: Long paths are sensitized for fault detection in high-quality at-speed scan testing. This means larger ND in Fig. 1 (b).
- Short Test Cycle: The time between launch and capture is short. This means smaller T in Fig. 1 (a).

All these factors cause ND + PD > T to occur frequently, which often makes a normal circuit to fail in testing. Since this results in power-supply-noise-induced yield loss, power supply noise has become a critical issue in high-quality at-speed scan testing [14, 15].

The impact of power supply noise can be analyzed in terms of voltage drop and/or delay increase. Approximation methods [10, 14, 15] are often used since accurate analysis is time-consuming. Their results are used to guide test generation [14] or reduce the number of target test vectors before accurate analysis is conducted for sign-off [15].

Power supply noise reduction is more important, which requires to lower launch-induced switching activity. For LOC-based at-speed scan testing, this means that switching activity due to the first capture (C₁ in Fig. 2) should be reduced. Three approaches are available for this purpose:

1. Test Clocking: One-hot and multi-capture clocking schemes can reduce the number of clock domains that capture simultaneously [3].
2. Circuit Change: DFT methods, such as partial capture [16], can be used to allow only part of a circuit to capture.
3. Test Generation: Switching activity can be directly reduced by generating proper logic values in ATPG [17, 18], assigning logic values to don’t care bits by in-ATPG or post-ATPG X-filling [19-24], and test compaction [25].

Generally, the test generation approach is preferable. Especially, post-ATPG X-filling is now widely used in practice for power supply noise reduction [22, 24], since it can be easily implemented into any ATPG flow, without any impact on test data volume, area overhead, and circuit timing. More details will be described below.

1.4 Motivation

Post-ATPG X-filling for power supply noise reduction is illustrated in Fig. 4, which consists of two major steps:

1. Test Relaxation: This is the process to identify don’t care bits, referred to as X-bits, from a set of fully-specified test vectors under the condition that the property of the test set, e.g. its fault coverage, is preserved [26-28]. The result of this step is a set of test cubes with X-bits.
2. X-Filling: This is the process to assign logic values to the X-bits in a test cube so that the resulting fully-specified test vector has low launch-induced switching activity. Logic values can be determined by minimizing either the Hamming distance between a test vector and its response at FFs [19-22] or the weighted node transition count in a circuit [23].
The major advantage of post-ATPG X-filling is that it reduces power supply noise without inflating test data volume. As shown in Fig. 4, this is achieved by using test relaxation to obtain X-bits from a compact initial test vector set generated by ATPG with dynamic compaction and random-fill [3]. Forcing ATPG to leave unspecified bits as X-bits without using them to improve detection efficiency will result in much larger test sets. Note that, although over 90% of bits may remain unspecified even after dynamic compaction, not conducting random-fill on them will still significantly increase test data volume [21, 22].

However, previous post-ATPG X-filling methods for power supply noise reduction suffer from two major problems with respect to high-quality at-speed scan testing:

### Problem-1: Test Quality Degradation in Test Relaxation

Previous test relaxation methods [26-28] can be readily extended for the transition delay fault model to identify X-bits from a fully-specified transition delay test set while preserving its fault coverage. However, all of them totally ignore the paths sensitized for transition delay fault detection. Consequently, X-filling the test cubes obtained by such test relaxation methods results in a final test set with different sensitized paths. As illustrated in Fig. 5, if the initial test set has high delay test quality characterized by its characteristic path set, the delay test quality of the final test set can be very different, usually low, since it has a different characteristic path set with possibly shorter paths. Therefore, test relaxation ignoring path sensitization information may result in significant delay test quality degradation.

### Problem-2: Effectiveness vs. Scalability in X-Filling

Ideal X-filling should be both effective in reducing launch-induced switching activity as much as possible and scalable to large circuits in terms of short CPU time. Previous X-filling methods use time-consuming justification to maximize effectiveness [19, 21] or aggressive approximation based on signal probability to maximize scalability [22]. As illustrated in Fig. 6, there is a need for more balanced X-filling, which is highly effective and sufficiently scalable, especially for high-quality at-speed scan testing of industrial circuits. This is due to the facts of (1) growing circuit scale, (2) less X-bits identified by test relaxation since more properties, e.g. fault coverage as well as delay test quality, should be preserved, and (3) the need of using part of X-bits for other purposes, such as test compression [3].

### 1.5 Contributions

This paper tackles the above problems with a new post-ATPG X-filling scheme. (1) Problem-1 is solved by a unique test relaxation method, called path-keeping X-identification, which finds X-bits from a set of fully-specified transition delay test vectors while keeping the set of longest paths, i.e. characteristic path set, sensitized by the original fully-specified test vectors for fault detection. This preserves the delay test quality, as well as the fault coverage, of the original test set. (2) Problem-2 is solved by an effective and scalable X-filling method, called justification-probability-based fill (JP-fill), in which techniques based on justification and probability are used selectively, depending on how X-bits appear at the inputs and outputs of corresponding flip-flops. This achieves effectiveness and scalability in a more balanced manner.

The new post-ATPG X-filling scheme can be easily implemented into any ATPG flow to effectively reduce power supply noise without any impact on delay test quality, test data volume, area, and timing. This greatly improves the applicability of high-quality at-speed scan testing.

### 1.6 Organization

The rest of the paper is organized as follows: Section 2 describes the background. Section 3 outlines the new scheme. Sections 4 and 5 present the details of path-keeping X-identification and JP-fill, respectively. Section 6 shows experimental results, and Section 7 concludes the paper.

### 2. Background

We first discuss issues related to delay test quality. Then, we review test relaxation and X-filling in detail.

#### 2.1 Delay Test Quality

##### A. Characteristic Path Set

As illustrated in Fig. 1 (a), scan-based delay testing for a path is conducted by creating a start transition at the start-point of the path and establishing a sensitization path to propagate the transition effect to the end-point of the path.
A sensitization path can be established either explicitly by path delay test generation or implicitly by transition delay test generation [3]. In path delay test generation, a target path is explicitly specified and test generation is conducted to sensitize it directly. In transition delay test generation, a slow-to-rise or slow-to-fall transition fault is assumed at a node, which by itself does not specify any path. As illustrated in Fig. 7, transition delay test generation then tries to establish an excitation path from the start-point of the path to the fault site for creating a proper transition at the fault site and a propagation path for passing the fault effect from the fault site to the end-point of the path. This way, a sensitization path, which is the combination of the excitation path and the propagation path, is established implicitly. In this paper, we focus on transition delay test generation since it is more widely used in practice.

![Fig. 7 Paths in Transition Delay Test Generation.](image)

**Definition 1:** Suppose that $V$ is a transition delay test set and \{$f_1, f_2, \ldots, f_d$\} is the set of all transition delay faults detected by $V$. The longest path sensitized by vectors in $V$ for a fault $f_i$ is called the **characteristic path** of the fault $f_i$ under the test set $V$, denoted by $CP(f_i, V)$. The set of all characteristic paths under $V$, \{$CP(f_i, V)$ | $i = 1, 2, \ldots, d$\}, is called the **characteristic path set** of $V$, denoted by $CPS(V)$.

For example, a test set $V$ can detect two faults: $f_a$ and $f_b$. Suppose that $f_a$ can be detected with three paths: $p_{a1}(7)$, $p_{a2}(9)$, $p_{a3}(5)$, while $f_b$ can be detected with two paths: $p_{b1}(5)$, $p_{b2}(3)$. Here, the number in the parenthesis is the length of the corresponding path. In this case, $CPS(V) = \{p_{a2}, p_{b1}\}$.

There are two approaches for obtaining the $CPS(V)$ for a transition delay test set $V$, depending on what type of ATPG, timing-ignoring or timing-aware, is used to generate $V$.

**Case-1 (Timing-Ignoring ATPG):** In this case, a transition delay fault $f$ is declared “detected” by a test vector $v \in V$ whenever a sensitization path is established, no matter how short the path may be [3]. As a result, there is no guarantee that the path sensitized by $v$ is the longest for the detection of $f$ under the whole test set $V$. Therefore, timing-based simulation is needed in order to find the longest sensitization path for each detected fault in order to obtain $CPS(V)$.

**Case-2 (Timing-Aware ATPG):** In this case, the excitation and propagation paths by a test vector $v$ are made as long as possible to detect a fault $f$ [2, 7-9]. As a result, the path sensitized by $v$ is usually the longest for the detection of $f$ under the whole test set $V$. Therefore, it is only necessary to simply record the sensitization path at the time when the corresponding fault $f$ is detected in test generation. That is, $CPS(V)$ is obtained as a by-product of timing-aware ATPG, and post-ATPG timing-simulation is not needed.

**B. Delay Test Quality Metric**

Generally, the delay test quality of a transition delay test set can be quantified by the **Statistical Delay Quality Level (SDQL)** metric [4]. The SDQL value for a transition delay test set $V$, denoted by $SDQL(V)$, is defined as

$$SDQL(V) = \sum_{i=1}^{n} \int_{T_{mgd}(f_i)}^{T_{det}(f_i, V)} F(s) \, ds$$

where the set of all possible faults is \{$f_1, f_2, \ldots, f_n$\}, and $F(s)$ is the delay defect distribution function, i.e. the probability of a defect having the delay size of $s$. In addition, $T_{mgd}(f_i)$ and $T_{det}(f_i, V)$ are defined as follows [4]:

- $T_{mgd}(f_i)$ is the maximum redundant delay size with respect to fault $f_i$. As shown in Fig. 3, $T_{mgd}(f_i) = T_{SC} - (the \text{ delay of the actual longest path passing through } f_i \text{ in function mode}),$ where $T_{SC}$ is the system clock period. Note that $T_{mgd}(f_i)$ is determined by design, and is independent of the test set $V$.

- $T_{det}(f_i, V)$ is the minimum detectable delay size with respect to fault $f_i$ by the test set $V$. If $f_i$ is undetected, $T_{det}(f_i, V) = \infty$; otherwise, as shown in Fig. 3, $T_{det}(f_i, V) = T_{TC} - (the \text{ delay of } CP(f_i, V))$, where $T_{TC}$ is the test clock period. Note that $T_{det}(f_i, V)$ is determined by $CP(f_i)$, which depends on the test set $V$. Also note that $T_{SC} = T_{TC}$ in at-speed scan testing.

Thus, it is clear that the delay test quality of a transition delay test set $V$, measured by $SDQL(V)$, is determined by $CPS(V)$, if the test timing (the relation between $T_{SC}$ and $T_{TC}$) and the delay defect distribution $F(s)$ are given. Generally, the longer the characteristic paths in $CPS(V)$, the higher the delay test quality of the transition delay test set $V$.

**C. Test Quality Preservation**

**Definition 2:** Suppose that $V_1$ and $V_2$ are two transition delay test sets, both detecting faults $f_1, f_2, \ldots, f_d$. If the length of $CP(f_i, V_2)$ is equal to or longer than that of $CP(f_i, V_1)$ for $i = 1, 2, \ldots, d$, it is said that $CPS(V_2) \supseteq CPS(V_1)$, and that the delay test quality of $V_1$ is preserved by $V_2$.

Since each characteristic path corresponds to a detected fault, $CPS(V_2) \supseteq CPS(V_1)$ means that the fault coverage and the delay test quality, measured by the SDQL metric [4], of $V_2$ are guaranteed to be no lower than those of $V_1$.

**2.2 Test Relaxation**

**Test relaxation** is the process of identifying $X$-bits from a set of fully-specified test vectors, while preserving its property, such as fault coverage. The resulting test cubes are often used for test data reduction [26], test quality improvement [28], and test power/noise reduction [19-25].

The major advantage of obtaining test cubes by test relaxation, instead of simply keeping unspecified bits in ATPG, is that a compact initial test set can be obtained through dynamic compaction and random-fill; otherwise, the test vector count may even double [22]. After a compact initial test set is generated, test relaxation can then be applied to obtain $X$-bits, usually as much as over 50% [27,28].
Previous test relaxation methods are as follows: (1) *Bit-stripping* [26] changes one bit in a test vector into an X-bit and checks whether all the faults that are only detected by the vector are still detected. (2) The method in [27] identifies newly detected faults by a test vector, and marks all the lines whose values are required for each such fault to be detected. The unmarked input lines are set to X-bits. (3) *X-identification* [28] uses justification and implication to find input bits needed to detect each fault that is detected by only one test vector. Other input bits are turned into X-bits after adjustment is made so that all faults are detected.

However, all previous test relaxation methods only preserve fault coverage, but totally ignore the information on sensitization paths. Thus, the initial test set $V_1$ and the final test set $V_2$, obtained by X-filling the test cubes identified by such test relaxation, usually have different characteristic path sets, i.e. $CPS(V_1) \neq CPS(V_2)$. It is highly possible that $CPS(V_2)$ has shorter characteristic paths. As described in Section 2.1, this means that the delay test quality of the final test set $V_2$ is lower than that of the initial test set $V_1$.

Thus, test relaxation ignoring sensitization paths is not feasible for high-quality at-speed scan testing. This is tackled by a new test relaxation method, called *path-keeping X-identification*, whose details are described in Section 4.

### 2.3 X-Filling

*X-filling* is the process of properly assigning logic values to the X-bits in a test cube for achieving a specific goal. Here, the goal is to reduce launch-induced switching activity.

There are two major approaches to such X-filling, as shown in Figs. 8 and 9. Here, $c$ is a test cube and $f(c)$ is the response of the combinational portion $f$. Logic values need to be determined for the X-bits in $c$, such that the Hamming distance between $<p_1, q_1, r_1>$ and $<p_2, q_2, r_2>$ is reduced.

*Justification-Based X-Filling* [19,21]: This is a multi-pass approach illustrated in Fig. 8. PPI-PPO bit-pairs of the form $<X>$ are processed first, followed by those of the form $<X, X>$, one at a time. Here, $<p_1, p_2>$ is processed first by justifying 0 on $p_2$ since $p_1$ has 0. Suppose that this is achieved by setting 1 to $a$. Then, bit-pairs of the form $<X, X>$ are processed. Note that one such bit-pair may require two passes of justification. For example, for $<r_1, r_2> = <X, X>$, 1 is first justified on $r_2$ and 1 is assigned to $r_1$. If this fails, 0 is then justified on $r_2$ and 0 is assigned to $r_1$. This approach is highly effective. However, it is less scalable due to long run time, especially for bit-pairs of the form $<X, X>$.

*Probability-Based X-Filling* [22]: This is a single-pass approach illustrated in Fig. 9. The 0 and 1 probabilities of each node are first calculated by setting 0.50 as the 0 and 1 probabilities for each input X-bit and conducting probability propagation. Then, the logic value for a PPI X-bit is determined by using the relation between the 0 and 1 probabilities of its corresponding PPO X-bit. For example, $q_2 = 0.93, 0.07$ on $q_2$ means that $q_2$ is likely to be 0, thus, it is reasonable to set 0 to $q_1$. This approach is highly scalable. However, its effect may be damaged by approximation in probability calculation. Especially, if the difference between the 0 and 1 probabilities at a PPO X-bit is negligibly small, e.g. $(0.49, 0.51)$ on $r_2$, logic value determination for its corresponding PPI X-bit becomes highly inaccurate.

Thus, there is a need for more balanced X-filling, that is highly effective and sufficiently scalable. This is achieved by a new X-filling method, called *justification-probability-based fill* (JP-fill), whose details are described in Section 5.

### 3. New Post-ATPG X-Filling Scheme

The general flow of the new post-ATPG X-filling scheme is shown in Fig. 10. It is a post-processing procedure conducted on an initial transition delay test set $V_1$, such that the resulting final test set $V_2$ has lower launch-induced switching activity, thus lower power supply noise.

![Fig. 8 Justification-Based X-Filling.](image)

![Fig. 9 Probability-Based X-Filling.](image)

![Fig. 10 General Flow of New Post-ATPG X-Filling.](image)

The new scheme has two unique characteristics:

1. **Test Quality Preservation in Test Relaxation**: Delay test quality, as well as fault coverage, of the original test set is preserved, i.e. $CPS(V_2) \geq CPS(V_1)$. This is achieved by *path-keeping X-identification*, to be described in Section 4.
2. **Balanced Effectiveness and Scalability in X-Filling**: The effectiveness and scalability of X-filling are achieved in a more balanced manner by *justification-probability-based fill* (JP-fill), to be described in Section 5.
4. New Test Relaxation Method
As described in 2.1, the delay test quality of a transition delay test set \( V \) is largely determined by its characteristic path set \( CPS(V) \). Therefore, test relaxation with the delay test quality preservation capability can be achieved by path-keeping \( X \)-identification as stated below:

**Path-Keeping \( X \)-identification:** Given a fully-specified transition delay test set \( V \) and its characteristic path set \( CPS(V) \), find a partially-specified test cube set \( C \), such that \( CPS(C) = CPS(V) \).

Suppose that \( V_1 \) is the original fully-specified test set, and \( C \) is the partially-specified test cube set obtained by path-keeping \( X \)-identification. That is, \( CPS(C) = CPS(V_1) \). Also suppose that \( V_2 \) is the final fully-specified test set obtained by \( X \)-filling \( C \). Since \( X \)-filling cannot affect characteristic paths already sensitized by \( C \) but may sensitize even longer paths, we have \( CPS(C) \leq CPS(V_2) \). As a result, \( CPS(V_2) \geq CPS(V_1) \). This means that the delay test quality of \( V_2 \), measured by SDQL [4] discussed in Section 2.1, is not lower than that of \( V_1 \). Therefore, the delay test quality of \( V_1 \), as well as its fault coverage, is preserved by \( V_2 \) in post-ATPG \( X \)-filling because of path-keeping \( X \)-identification.

In the following, we first describe the basic idea of path-keeping \( X \)-identification. Then, we present its key operation. Finally, we show its general flow.

4.1 Basic Idea
The relation between \( V \) and its characteristic path set \( CPS(V) \) can be expressed by a sensitization table, which lists all test vectors and all characteristic paths sensitized by each test vector. An example is shown in Table 1.

<table>
<thead>
<tr>
<th>Test Vector</th>
<th>Sensitized Characteristic Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = 011 )</td>
<td>( p_1 ) ( p_2 )</td>
</tr>
<tr>
<td>( v_2 = 101 )</td>
<td>( p_4 )</td>
</tr>
<tr>
<td>( v_3 = 111 )</td>
<td>( p_5 )</td>
</tr>
</tbody>
</table>

Table 1  Sensitization Table

In Table 1, \( V = \{ v_1, v_2, v_3 \} \) and \( CPS(V) = \{ p_1, p_2, \ldots, p_5 \} \). Note that one characteristic path may be sensitized by two or more test vectors. For example, \( p_3 \) is sensitized by both \( v_2 \) and \( v_3 \). However, such cases are seldom for test vectors generated by timing-aware ATPG [2, 9].

The basic idea of path-keeping \( X \)-identification is as follows: For each fully-specified test vector \( v_i \) in a test vector set \( V = \{ v_i | i = 1, 2, \ldots, n \} \), create a partially-specified test cube \( c_i \), such that \( c_i \) sensitizes all characteristic paths sensitized by \( v_i \). Obviously, for the resulting test cube set \( C = \{ c_i | i = 1, 2, \ldots, n \} \), \( CPS(C) = CPS(V) \) holds.

Table 2 shows an example of the basic idea, for the test vector set \( V = \{ v_1, v_2, v_3 \} \) given in Table 1. Path-keeping \( X \)-identification turns \( V \) into the test cube set \( C = \{ c_1, c_2, c_3 \} \), such that \( c_1 \) sensitizes \( p_1 \) and \( p_2 \) as \( v_1 \), \( c_2 \) sensitizes \( p_3 \) and \( p_4 \) as \( v_2 \), and \( c_3 \) sensitizes \( p_5 \) as \( v_3 \). Obviously, \( CPS(C) = \{ p_1, p_2, p_3, p_4, p_5 \} \). This means that \( CPS(C) = CPS(V) \).

Note that if a characteristic path is sensitized by multiple test vectors, then it will be targeted by just one test vector in test relaxation, so that more \( X \)-bits can be identified. In Table 1, for example, \( p_1 \) is sensitized by both \( v_2 \) and \( v_3 \). As shown in Table 2, since \( c_2 \) sensitizes both \( p_3 \) and \( p_4 \), it is only necessary to make \( c_3 \) sensitize \( p_5 \), instead of both \( p_3 \) and \( p_5 \). As a result, \( c_3 \) contains more \( X \)-bits.

4.2 Test Cube Creation
From the basic idea described in Section 4.1, it can be seen that the key operation of path-keeping \( X \)-identification is **test cube creation**. Its purpose is to create a partially-specified test cube \( c \) from a fully-specified test vector \( v \), such that \( c \) sensitizes all the characteristic paths, \( p_1, p_2, \ldots, p_k \), that are originally sensitized by \( v \).

Test cube creation for obtaining such a test cube \( c \) is conducted by the following two steps:

**Step-1 (Creating Primary Test Cubes):** \( k \) primary test cubes, \( c_1, c_2, \ldots, c_k \), are created from the test vector \( v \) for the \( k \) characteristic paths, \( p_1, p_2, \ldots, p_k \), respectively, such that \( c_i \) also sensitizes \( p_i \) as \( v \) does \( (i = 1, 2, \ldots, k) \).

**Step-2 (Merging Primary Test Cubes):** All primary test cubes, \( c_1, c_2, \ldots, c_k \), are merged into a final test cube \( c \), such that \( c \) also sensitizes \( p_1, p_2, \ldots, p_k \) as \( v \) does. Here, \( c \) sensitizes \( p_i \) \( (i = 1, 2, \ldots, k) \).

The two steps are described in more detail in the following.

**A. Creating Primary Test Cubes**
Given a fully-specified test vector \( v \) and a characteristic path \( p_i \) sensitized by \( v \), a *primary test cube* \( c_i \) needs to be created from \( v \), such that \( c_i \) also sensitizes \( p_i \) as \( v \) does. This task can be accomplished by a simple but powerful cone-analysis-based technique. An example is shown in Fig. 11.

Fig. 11 (a) shows a full-scan circuit with \( b_2, b_3, \ldots, b_5 \) as inputs to its combinational portion, and Fig. 11 (b) shows its two-time-frame circuit model for the launch-off-capture-based at-speed scan testing (Fig. 2). A test vector \( v = <011 101> \) is applied to \( b_1, b_2, \ldots, b_5 \). Suppose that \( v \) sensitizes the characteristic path \( p_0 \), whose start and end points are denoted by \( S \) and \( E \), respectively.

In Fig. 11 (b), cone analysis is conducted from the end-point (E) of the characteristic path \( p_0 \) in both time-frames, in order to identify which inputs in \( \{ b_1, b_2, b_3, b_4, b_5 \} \) can affect the sensitization of \( p_0 \). Note that the cone analysis from the end-point of \( p_0 \) in the 2nd time-frame should be conducted in both time-frames as shown in Fig. 11 (b).
The result of the cone analysis shows that only $b_2$, $b_3$, and $b_4$ can affect the path $p_i$ for sensitization, and they are called the essential inputs for the characteristic path $p_i$. On the other hand, $b_1$ and $b_5$ have no impact on $p_i$, and they are called the non-essential inputs for the characteristic path $p_i$. Therefore, the 1st and 5th bits in $v = <0\;1\;1\;0\;1>$, which correspond to the non-essential inputs $b_1$ and $b_5$, respectively, can be turned into $X$-bits to create a test cube $c_i$ without affecting the sensitization state of the characteristic path $p_i$ at all. This way, a primary test cube, $c_i$, is created.

Note that primary test cube creation is based on fast cone analysis to identify $X$-bits. This makes the process highly efficient even for large industrial circuits. More aggressive techniques based on timing analysis may identify more $X$-bits, but are too time-consuming to be practical.

### B. Merging Primary Test Cubes

Generally, if a test vector $v$ sensitizes $k$ characteristic paths $p_1, p_2, \ldots, p_k$, the above cone-analysis-based technique can be applied to create $k$ primary test cubes $c_1, c_2, \ldots, c_k$, respectively. It is obvious that the merged test cube $c = c_1 \cap c_2 \ldots \cap c_k$ also sensitizes $p_1, p_2, \ldots, p_k$ as the test vector $v$.

That is, a final test cube $c$ is created that sensitizes the same set of characteristic paths as the test vector $v$. This way, test cube creation in path-keeping $X$-identification is completed.

Note that the $\cap$ operation is a bit-wise operation conducted on a pair of corresponding bits in two test cubes, based on the following rules:

$$X \cap X = X \quad X \cap 0 = 0 \quad X \cap 1 = 1$$

An example is shown in Fig. 12. Here, the test vector $v = <0\;1\;1\;0\;1>$ sensitizes characteristic paths $p_1$ and $p_2$. Two primary test cubes are created: $c_1 = <X\;1\;1\;X\;X>$ sensitizes characteristic paths $p_1$, and $c_2 = <X\;1\;X\;0\;X>$ sensitizes $p_2$. The final test cube is $c = c_1 \cap c_2 = <X\;1\;1\;X\;X> \cap <X\;1\;X\;0\;X> = <X\;1\;1\;0\;X>$. Obviously, $c$ sensitizes characteristic paths $p_1$ and $p_2$ as $v$.

### 4.3 General Flow

The flow of path-keeping $X$-identification is shown in Fig. 13. The inputs are a fully-specified test vector set $V$ and its characteristic path set $CPS(V)$, and the output is a partially-specified test cube set $C$, and $CPS(C) = CPS(V)$.

![General Flow of Path-Keeping X-Identification](image)

#### 5. New X-Filling Method

As described in Section 2.3, justification-based $X$-filling is effective but less-scalable, while probability-based $X$-filling is scalable but less-effective. This is tackled by justification-probability-based fill (JP-fill), based on two techniques:

1. **Limited Justification**: Justification-based $X$-filling is only conducted for PPI-PPO bit-pairs of the form $<\text{logic value}, X>$, but not for any PPI-PPO bit-pair of the form $<X, X>$ for which multiple passes of justification may be needed. This is to achieve higher scalability.

2. **Probability Re-Calculation**: Probability-based $X$-filling is conducted for PPI-PPO bit-pairs of the form $<X, X>$, but in multiple passes. That is, the logic value for a PPI $X$-bit is determined only if its corresponding PPO $X$-bit has significantly different 0 and 1 probabilities; otherwise, probabilities are re-calculated in the next pass. This is to achieve higher effectiveness through improved accuracy.

Obviously, compared to previous $X$-filling methods based on only justification [19, 21] or only probability (no re-calculation) [22], JP-fill can achieve both effectiveness and scalability in a more balanced manner.
5.1 Circuit Model and Bit-Pairs

Fig. 14 shows the circuit model of LOC-based at-speed scan testing (Fig. 2). c is a test cube, whose PPI part is denoted by <c: PPI>. The response of the combinational portion to c is f(c), whose PPO part is denoted by <f(c): PPO>. When the first capture (C1 in Fig. 2) is conducted, <f(c): PPO> is loaded into all scan FFs to replace <c: PPI>. This causes launch-induced switching activity.

![Fig. 14 Circuit Model for Launch-Induced Switching Activity.](image)

The corresponding bits in <c: PPI> and <f(c): PPO> forms a bit-pair, whose types are shown in Table 3. X-filling action for each type of bit-pair in JP-fill is also shown.

### Table 3 Types of Bit-Pairs

<table>
<thead>
<tr>
<th>Type</th>
<th>X-Pair</th>
<th>PPI</th>
<th>PPO</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-A</td>
<td>a</td>
<td>b</td>
<td></td>
<td>No Action</td>
</tr>
<tr>
<td>Type-B</td>
<td>X_{PPI}</td>
<td>b</td>
<td></td>
<td>Assignment</td>
</tr>
<tr>
<td>Type-C</td>
<td>a</td>
<td>X_{PPO}</td>
<td></td>
<td>Justification</td>
</tr>
<tr>
<td>Type-D</td>
<td>X_{PPI}</td>
<td>X_{PPO}</td>
<td>Probability</td>
<td>(X_{PPI}, X_{PPO})</td>
</tr>
</tbody>
</table>

5.2 The General Flow of JP-Fill

The general flow of JP-fill is shown in Fig. 15. The details are described in the following.

![Fig. 15 General Flow of JP Fill.](image)

A. Assignment-Based X-Filling

Assignment-based X-filling is conducted on all Type-B bit-pairs first, before any bit-pair of other type is processed.

As shown in Fig. 15-D, for a Type-B X-pair \{X_{PPI}, b\}, since any value can be set to X_{PPI} by scan shift, logic value b is assigned to X_{PPI} so that a capture transition is avoided.

B. Justification-Based X-Filling

Justification-based X-filling is conducted on all Type-C bit-pairs before Type-D bit-pairs are processed, after all Type-B bit-pairs are processed.

As shown in Fig. 15-E, for a Type-C bit-pair \{a, X_{PPO}\}, logic value a is justified for the X_{PPO} bit. If this justification succeeds, a is set to X_{PPO}, so that a capture transition is avoided. If this justification fails, ~a is set to X_{PPO}.

C. Probability-Based X-Filling

Probability-based X-filling is conducted on all Type-D bit-pairs after all bit-pairs of other types are processed.

First, the 0 and 1 probabilities, denoted by (0_{Prob}, 1_{Prob}), are calculated for each node. The initial 0 and 1 probabilities for an input with an X-bit are assumed to be 50%, and any probability propagation method can be used [22].

Suppose that the PPO X-bits are X_{PPO}^1, X_{PPO}^2, ..., X_{PPO}^n, and that their probabilities are (0_{Prob}^1, 1_{Prob}^1), (0_{Prob}^2, 1_{Prob}^2), ..., (0_{Prob}^m, 1_{Prob}^m). The average difference between their 0 and 1 probabilities is calculated as follows:

\[
\Delta = \left( \sum_{i=1}^{m} (0_{\text{Prob}}^i - 1_{\text{Prob}}^i) \right) / m
\]

Now consider a Type-D bit-pair \{X_{PPI}, X_{PPO}\}. Suppose that the 0 and 1 probabilities of X_{PPO} are (0_{Prob}, 1_{Prob}). As shown in Fig. 15-F, different X-filling strategies are used, depending on the relation between 0_{Prob} and 1_{Prob}:

**Case-1** (0_{Prob} - 1_{Prob} > \Delta): In this case, the 0 and 1 probabilities are significantly different at X_{PPO}; thus, the chance of X_{PPO} having the logic value corresponding to the higher probability (0_{Prob} or 1_{Prob}) is high. Therefore, if 0_{Prob} > 1_{Prob}, 0 is set to X_{PPO}; otherwise, 1 is set to X_{PPO}. This is the same as the preferred fill method [22].

**Case-2** (0_{Prob} - 1_{Prob} ≤ \Delta): In this case, the difference between the 0 and 1 probabilities at X_{PPO} is negligibly small, making it inaccurate to use them for determining a logic value for X_{PPO}. Thus, no action is taken. This leads to a new pass of X-filling, in which probabilities are re-calculated to reflect the newly assigned logic values at inputs. As a result, the accuracy of X-filling is improved.

5.3 Summary

In JP-fill, time-consuming justification for Type-D bit-pairs is avoided (Fig. 15-\(\text{G}\)), and the inaccuracy in probability calculation is alleviated by re-calculation (Fig. 15-\(\text{G}\)) when 0 and 1 probabilities are so close that proper logic value determination becomes impossible. Thus, more balanced effectiveness and scalability can be achieved by JP-fill.
6. Experimental Results

Path-keeping X-identification and JP-fill were implemented in C, and experiments were conducted on 10 ISCAS’89 benchmark circuits on a computer with a 2.6GHz CPU and 16GB memory. The transition delay fault model was used.

A. Test Relaxation Results

Table 4 shows the results of path-keeping X-identification on initial test sets generated by two types of ATPG: timing-ignoring and timing-aware [9], both developed internally. The test vector count, fault coverage, percentage of identified X-bits, and CPU time are shown under “Test Vec. #”, “Fault Cov. (%)”, “X (%)”, and “CPU (s)”, respectively.

From Table 4, it is clear that path-keeping X-identification is effective in that it identified 54.7% and 67.8% of bits as X-bits from fully-specified initial test sets. This method is also efficient in that its CPU time was fairly short since the cone-analysis-based technique used is very fast.

Table 4 Results of Path-Keeping X-Identification

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Timing-Ignoring ATPG</th>
<th>Timing-Aware ATPG</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test Vec. #</td>
<td>Fault Cov. (%)</td>
<td>X (%)</td>
<td>CPU (s)</td>
<td>Test Vec. #</td>
</tr>
<tr>
<td>s1196</td>
<td>26</td>
<td>90.8</td>
<td>41.6</td>
<td>0.0</td>
<td>24</td>
</tr>
<tr>
<td>s1238</td>
<td>26</td>
<td>90.8</td>
<td>41.6</td>
<td>0.0</td>
<td>24</td>
</tr>
<tr>
<td>s1423</td>
<td>76</td>
<td>85.8</td>
<td>24.7</td>
<td>0.1</td>
<td>148</td>
</tr>
<tr>
<td>s5378</td>
<td>178</td>
<td>84.8</td>
<td>53.3</td>
<td>1.0</td>
<td>379</td>
</tr>
<tr>
<td>s9234</td>
<td>376</td>
<td>81.3</td>
<td>50.8</td>
<td>3.7</td>
<td>839</td>
</tr>
<tr>
<td>s13207</td>
<td>323</td>
<td>79.5</td>
<td>67.4</td>
<td>4.4</td>
<td>838</td>
</tr>
<tr>
<td>s15850</td>
<td>221</td>
<td>70.2</td>
<td>54.3</td>
<td>4.0</td>
<td>703</td>
</tr>
<tr>
<td>s35932</td>
<td>337</td>
<td>82.5</td>
<td>92.0</td>
<td>12.6</td>
<td>763</td>
</tr>
<tr>
<td>s38417</td>
<td>270</td>
<td>98.0</td>
<td>47.1</td>
<td>12.9</td>
<td>2605</td>
</tr>
<tr>
<td>s38584</td>
<td>412</td>
<td>83.9</td>
<td>73.9</td>
<td>17.0</td>
<td>2111</td>
</tr>
<tr>
<td>Ave.</td>
<td>54.7</td>
<td>67.8</td>
<td>15.01</td>
<td>14.93</td>
<td></td>
</tr>
</tbody>
</table>

Path-keeping X-identification preserves the longest path sensitized by the initial test set for each detected fault when creating the intermediate test cube set. Therefore, the delay test quality, as well as the fault coverage, of the final test set obtained by X-filling the intermediate test cube set is guaranteed to be no lower than that of the initial test set.

We used the SDQL metric [4] to quantify the delay test quality of initial and final test sets (obtained by the JP-fill method) for timing-aware ATPG, and the results are also shown in Table 4. It can be seen that the final SDQL values are equal to or even smaller than the values for the initial test sets. This confirms that the delay test quality was not degraded due to test relaxation. Note that the smaller the SDQL value, the higher the delay test quality [4].

B. X-Filling Results

Table 5 shows the X-filling results by a justification-based method [21], a probability-based method [22], and the JP-fill method. Initial test vectors were generated by timing-aware ATPG [9], and test cubes were obtained by path-keeping X-identification. The WSA (Weighted Switching Activity) metric [12, 22] was used for estimating the launch-induced switching activity at the first capture (C1 in Fig. 2).

Table 5 Results of Various X-Filling Methods

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Max. WSA-FF</th>
<th>Max. WSA-Node</th>
<th>Ave. WSA-Node</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1196</td>
<td>43.2</td>
<td>22.8</td>
<td>29.8</td>
<td>21.8</td>
</tr>
<tr>
<td>s1238</td>
<td>50.0</td>
<td>20.1</td>
<td>27.5</td>
<td>15.7</td>
</tr>
<tr>
<td>s1423</td>
<td>17.1</td>
<td>4.6</td>
<td>8.7</td>
<td>8.4</td>
</tr>
<tr>
<td>s5378</td>
<td>30.1</td>
<td>16.7</td>
<td>45.7</td>
<td>23.2</td>
</tr>
<tr>
<td>s9234</td>
<td>31.2</td>
<td>7.1</td>
<td>23.4</td>
<td>18.9</td>
</tr>
<tr>
<td>s13207</td>
<td>30.1</td>
<td>15.5</td>
<td>23.4</td>
<td>18.9</td>
</tr>
<tr>
<td>s15850</td>
<td>32.1</td>
<td>7.0</td>
<td>24.5</td>
<td>16.4</td>
</tr>
<tr>
<td>s35932</td>
<td>6.2</td>
<td>6.3</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>s38417</td>
<td>30.9</td>
<td>15.4</td>
<td>56.8</td>
<td>80.1</td>
</tr>
<tr>
<td>s38584</td>
<td>10.5</td>
<td>10.5</td>
<td>35.8</td>
<td>45.2</td>
</tr>
</tbody>
</table>

Fig. 16 Comparison of Various X-Filling Methods.
Fig. 16 is the graphical form of the results given in Table 5. The effectiveness of the X-filling methods was evaluated by the launch-induced switching activity at all FFs (“WSA-FF”) and all nodes (“WSA-Node”), at maximum and in average, as shown in Fig. 16 (a) – (c). The scalability of the X-filling methods was evaluated by CPU time, as shown in Fig. 16 (d). Evidently, the justification-based method [21] is the best-performing for effectiveness, while the probability-based method [22] is the best-performing for scalability.

Fig. 16 (a) – (c) also show that, for effectiveness, the JP-fill method is much closer to the best-performing justification-based method [21] than the probability-based method [22]. Fig. 16 (d) also shows that, for scalability, the JP-method is much closer to the best-performing probability-based method [22] than the justification-based method [21]. This confirms that the JP-fill method proposed in this paper can indeed achieve both effectiveness and scalability in a more balanced manner than the previous X-filling approaches based on only justification or only probability.

7. Conclusions

This paper proposed a new post-ATPG X-filling scheme, featuring two novel and practical methods: (1) path keeping X-identification for finding don’t-care bits from a fully-specified transition delay test set while preserving its delay test quality by keeping the longest paths sensitized for fault detection, and (2) justification-probability-based fill (JP-fill) for X-filling test cubes in an effective and scalable manner. This scheme can be easily implemented into any ATPG flow to effectively reduce power supply noise in high-quality at-speed scan testing, without any impact on delay test quality, test data volume, area, and timing.

Experiments on large industrial circuits are currently under way, and the results will be included in the final version. Future research subjects include (I) applying the basic idea and techniques of this paper to path delay test generation, and (II) extending them for test compression schemes.

References


Paper 25.1 INTERNATIONAL TEST CONFERENCE 10