Seed Ordering and Selection for High Quality Delay Test

Tomokazu Yoneda*, Michiko Inoue**, Akira Takezawa*** and Hideo Fujiwara**

* Nara Institute of Science and Technology, Kansai Science City, 630-0192, Japan
** Japan Science and Technology Agency, CREST, Chiyoda-ku, Tokyo, 102-0075, Japan
*** yoneda, kounoe, fujiwara@is.naist.jp

Abstract

This paper presents a seed ordering and selection method in LFSR-seeding-based BIST for high quality delay test. The proposed method selects seeds based on the gain in the sum of the longest path lengths sensitized by seeds, which is highly correlated with statistical delay quality level (SDQL). We also evaluate the contributions of pseudo-random patterns in several mixed-mode BIST approaches and the impact of base seed set on the final quality of selected seeds. Experimental results show that the proposed method intelligently selects seeds and obtains significant reduction in seed count under SDQL-constraint within a reasonable time.

Keywords: BIST, seed ordering, delay test, SDQL.

1 Introduction

Nanometer technologies have led to a drastic increase in operational frequency, and screening timing-related defects has become more important to ensure product quality. Such timing-related defects caused by resistive opens, resistive shorts and process variations manifest themselves as small delay variations called small delay defects (SDDs). In addition, transistor aging has been known as a troublesome phenomenon in the field. It is well known that aging causes gradual delay increase and finally leads to a system failure [1, 2, 3]. Therefore, high quality delay test as well as built-in self-test (BIST) are required for ensuring high field reliability.

Though ATPG tools based on the traditional transition delay fault model is widely used, the delay test quality for SDDs has been questioned because they tend to activate the faults through short paths [4, 5]. Therefore, commercial timing-aware ATPG tools were introduced recently [5, 6, 7]. However, timing-aware ATPGs require long CPU run time for pattern generation and fault simulation, and result in a significantly large pattern count. Test data volume reduction is therefore essential especially for the field test where the resources (memory or hardware) for test data is limited.

Test pattern ordering methods, which rank test patterns and place the most effective test patterns at the top of the ordered sequence, can be effective to reduce test data volume. Several techniques have been proposed recently to reduce the pattern count for screening SDDs in this direction. In [8], authors use the output deviation [9] as a surrogate long-path coverage metric for SDDs. Similar methods were proposed to take into account the interconnect contribution [10] and process variations and crosstalk contributions [11] to the total delay of sensitized paths. In [12], an efficient pattern grading and selection method using standard delay format (SDF) timing information was proposed for screening SDDs, which has no output deviation saturation problem like [8, 10]. In [13], a SDD-aware seed selection technique was presented for LFSR-seeding-based test compression, which also utilizes the output deviation as a surrogate long-path coverage metric for SDDs.

However, the previous works have the following problems when we consider the field reliability. First, in all the previous work [8, 10, 11, 13], the “number of activated long paths” is considered to be a useful metric for evaluating the delay test quality. They defined long path limits (between 70-90% of the system clock period in [13]), and evaluated the number of activated distinct long paths within the limit. However, SDDs which can be activated only through short and intermediate paths are also important for reliability since a SDD escapes on such paths during test might magnify during subsequent aging in the field and cause a failure of the device [4]. Secondly, only [13] considered the seed selection problem in LFSR-seeding-based test compression for the detection of SDDs while the others tackled the pattern selection problem without test compression. Besides, in [13], only the deterministic patterns (i.e., one seed per pattern) are considered in the LFSR-based compression environment. However, when a seed is loaded in the LFSR, we can apply some pseudo-random patterns (known as mixed-mode BIST [14, 15, 16]) and there is a possibility that they will detect more SDDs so that some of the seeds are not needed.

In this paper, we address the seed ordering and selection problem for high quality delay test. We adopt “Statistical Delay Quality Model (SDQM)” proposed by Sato et al. [17] as a model of delay test quality, which is also adopted for commercial timing-aware ATPG tools [5, 6, 7]. The SDQM is based on a statistical delay defect distribution function and careful consideration on relations between defect sizes and activated path lengths. The model uses a metric called “Statistical Delay Quality Level (SDQL)” for each test set to evaluate its quality. The SDQL for a given test set represents the amount of delay defects that should be detected but cannot be detected by the test set. We also adopt a LFSR-seeding-based BIST architecture as in [13]. Our contributions are summarized as follows.

- We present a seed ordering and selection technique for LFSR-seeding-based BIST to achieve high quality delay test. The proposed method selects seeds based on the gain in the sum of the longest path lengths sensitized by seeds, which is highly correlated with statistical delay quality level (SDQL).
- Experiments for several ITC’99 benchmark circuits show the proposed seed ordering and seed selection method can obtain significant seed count reduction under SDQL-constraint within a reasonable time.
- We explore several mixed-mode BIST approaches where the ratio between deterministic and pseudo-random patterns are different and study the impact of pseudo-random patterns contribute to test data volume reduction in high quality delay test.
- We apply the proposed method to several base seed sets generated by different ATPGs, and investigate the impact of base
Figure 1. Timing relations for longest true path length and longest sensitized path length.

Figure 2. Delay defect distribution function and SDQL for a fault.

2 Statistical Delay Quality Model (SDQM)

In this section, we introduce statistical delay quality model (SDQM) and statistical delay quality level (SDQL) proposed by Sato et al [17]. The SDQM is proposed to evaluate test quality based on a delay defect distribution function which is derived from fabrication process. The SDQM considers rising and falling delay faults on each of input and output pins of each gate. Though the number of faults is the same as transition faults, a delay defect size is associated with each fault. Figure 1 shows a concept of delay defect sizes that should be detected and can be detected by a given test set. Let \( f \) be a fault, and let \( L_A \) and \( L_B \) be the lengths of the longest true path passing through \( f \) and the longest path passing through \( f \) that is actually sensitized by the given test set, respectively. Let \( T_{MC} \) and \( T_C \) be system clock timing and test timing, respectively. The difference \( T_{min} = T_{MC} - L_A \) is the minimum delay defect size that can affect system behavior and therefore should be detected. The difference \( T_{min} = T_C - L_B \) is the minimum delay defect size that can be actually detected by the given test set.

The SDQl for a given test set is the amount of delay defects that should be detected but cannot be detected by the test set, and defined as follows, where \( N \) is the total number of faults and \( F(t) \) is a delay defect distribution function.

\[
SDQL = \sum_{f \in F} \int_{0}^{T_{min}} F(t) dt
\]

A shaded area in Fig. 2 shows an amount of delay defect for one fault escaped during test. Therefore, smaller SDQl means better delay test quality.

3 Seed Ordering and Selection

In this paper, we target a scan BIST architecture that consists of LFSR, phase shifter and MISR, and we focus on the input side of the BIST shown in Figure 3. Our technique is applicable with any number of scan chains and any phase shifter. In the BIST, deterministic patterns are encoded into seeds of \( n \) bits where \( n \) is the number of FFs in the LFSR. A seed \( s_i \) is loaded into the LFSR and then expanded into the desired test pattern in the scan chains by running the LFSR for \( m \) cycles, where \( m \) is the maximum length of the scan chains. If the LFSR runs for another \( m \) cycles, a pseudo-random pattern is expanded in the scan chains. In this paper, we consider a mixed-mode BIST technique where \( d_i \) patterns are expanded from each seed \( s_i \). That is, only the first pattern is a deterministic pattern and the remaining \( d_i - 1 \) patterns are pseudo-random patterns.

We present a seed ordering and selection method for the above BIST architecture to obtain high quality delay test based on SDQL. The outline of the proposed method is as follows.

1. Generate deterministic patterns and encode them into a seed set \( S_{base} \), called base seed set.
2. Order the seeds in \( S_{base} \) so that the SDQl improves by the maximum amount with the inclusion of each additional seed.
3. If there is a seed count constraint \( k \), select the top \( k \) seeds from the ordered sequence (this is the case to minimize SDQL under the seed count constraint).
4. If there is a SDQl constraint, select the seed from the top of the ordered sequence until the constraint is satisfied (this is the case to minimize the number of seeds to satisfy the SDQl constraint).

The deterministic patterns are generated by existing ATPG tools, and the patterns are encoded into seeds by solving a linear system of equations, which is an algebraic representation of the linear expansion of the LFSR and the phase shifter into the scan chain FFs [18]. Therefore, we focus on the ordering method in Step 2 in the following subsection.

3.1 Simulation Based Ordering

For the seed ordering in Step 2, we can consider the following method based on timing-aware fault simulation.

**Simulation Based**

1. For each \( s \in S_{base} \) run fault simulation for the expanded patterns from \( S + \{s\} \) to calculate SDQl.
2. For each \( s \in S_{base} \) run fault simulation for the expanded patterns from \( S + \{s\} \) to calculate SDQl.
3. For each \( s \) with the minimum SDQl, set \( S = S + \{s\} \) and \( S_{base} = S_{base} - \{s\} \).
In the $i$th iteration, it requires $|S| + (i-1)$ times fault simulation to obtain SDQL for each test pattern set expanded from $S + \{s\}$, which includes $\sum_{\ell=3}^{|S|} d_{i}$ patterns. However, timing-aware fault simulation to calculate SDQL is time-consuming. Table 1 shows CPU times required for timing-aware and non-timing-aware fault simulations for ITC benchmark circuits, where we used Synopsys TetraMAX and Synopsys Verification4300 with AMD Opteron256 3.0GHz and 16GB memory (Sun Microsystems). In timing-aware fault simulation, we have to consider not only fault detection but also the length of a sensitized path, and a fault can be dropped only when a test pattern detects the fault by the longest path passing through the fault. Therefore, it cannot be accelerated by fault dropping like non-timing-aware fault simulation, and Simulation-Based method is impractical for large circuits.

### 3.2 Proposed Seed Ordering

**SimulationBased** uses SDQL values to select a seed in each iteration, and therefore needs to run time-consuming fault simulation. In contrast, the proposed method uses the sum of the longest sensitized path lengths for all the faults instead of SDQL values. For each fault, the length of the longest path sensitized by a seed set can be easily calculated without delay defect distribution function $F(d)$ and fault simulation, once we obtain the length of the longest path sensitized by each seed. Let $L_f$ and $L_f'$ be the length of the longest path sensitized by the expanded patterns from seed set $S$ and seed set $S_f$ for a fault $f$, respectively. Let $L_f$ denote the sum of the longest sensitized path lengths for the expanded patterns from $S$, $L_{f\in S}$ is obtained as follows.

$$ L_{f\in S} = L_S + \sum_{\ell=3}^{|S|} \max(L_f' - L_{f\in S}, 0) $$

Let us define the gain $Gain_{f\in S}$ in the sum of the longest sensitized path lengths when $s$ is added to $S$ as follows.

$$ Gain_{f\in S} = L_{f\in S} - L_S = \sum_{\ell=3}^{|S|} \max(L_f' - L_{f\in S}, 0) $$

The proposed method selects a seed with the largest $Gain_{f\in S}$ as the $i$th test pattern. Though we do not directly evaluate SDQL values in each iteration like **SimulationBased**, the increase of the longest sensitized path length for a fault $f$ implies the decrease of $T_{delay}$. From Equation (1), it implies the decrease of SDQL. The outline of the proposed method is summarized as follows.

**SeedOrdering**

$S_{base}$ base seed set

1. For each $s \in S_{base}$ run fault simulation for the expanded patterns from $s$ to obtain SDQL and $F$.
2. For $s$ with the minimum SDQL, set $S = S + \{s\}$ and $S_{base} = S_{base} - \{s\}$.
3. Repeat Steps 4 and 5 until $S_{base}$ becomes empty.
4. For each $s \in S_{base}$, calculate $Gain_{f\in S}$.

For $s$ with the maximum $Gain_{f\in S}$, set $S = S + \{s\}$ and $S_{base} = S_{base} - \{s\}$.

In Step 1, we run fault simulation to obtain $F$ since TetraMAX can provide it through timing-aware fault simulation on and it is independent of delay defect distribution function $F(d)$. If $F(d)$ is available, then we can obtain a SDQL value for each seed as a byproduct of the fault simulation. That is why we select the first seed based on SDQL in Step 2. However, we can remove Step 2 if $F(d)$ is not available. In the proposed method, we apply fault simulation only one time for the above purpose, and therefore, it can order a seed set much faster than **SimulationBased**.

### 4 Experimental Results

In this section, we present experimental results using several ITC99 benchmark circuits. The characteristics of the circuits used in the experiments are summarized in Table 2. In the experiment, we used Synopsys TetraMAX ATPG with Small Delay Defect Test model. We can provide a delay defect distribution function $F(d)$ to TetraMAX to calculate SDQL in the form described as the following equation.

$$ F(d) = A \cdot e^{-Bd} + C $$

In all the experiments in this section, we set $A = 1$, $C = 0$, and set $B$ so that $F(T_{delay}) = 0.1$ holds where $T_{delay}$ denotes the system clock timing of the circuit. The values of $B$ are shown in the column "B" in $F(d)$ in Table 2.

To evaluate the test quality of the proposed ordering method, we first generate launch-off capture (LOC) test patterns with unspecified bits (X's) using the timing-aware ATPG, and encode them into a base seed set. Table 3 summarizes the ATPG results, the BIST architecture, and the base seed set generation results. "X ratio" and "C ratio" denote the average percentage of X's and the maximum number of specified bits in the generated patterns, respectively. "#scanchains" and "#escan" denote the number of scan chains and the channel separation between two adjacent scan chains implemented by the phase shifter, respectively. "SC" denotes the seed coverage which is the ratio of the number of the encoded seeds to the number of the generated patterns.

For the base seed sets in Table 3, we compared the proposed method with two different ordering methods: (1) ATPG-ordered (i.e., original order generated by ATPG) and (2) random ordering. In this experiment, we set $d_i$ (the number of the expanded patterns from seed $s_i$) to 1 for all seeds in the base seed set. Table 4 shows the CPU times required for the proposed method, and Figure 4 shows the SDQL transitions using the seeds ordered by ATPG, random, and the proposed ordering methods. The columns "Sim", "other" and "total" show CPU times for fault simulation, the other computation, and total computation, respectively. From these results, we can observe that the proposed method efficiently improves SDQL (i.e., achieves lower SDQL) compared to the ATPG and random ordering methods with the seed count constraints for all the circuits.
Table 3. Seed generation results for timing-aware patterns.

<table>
<thead>
<tr>
<th>circuit</th>
<th>TGT(μs)</th>
<th>PO (%)</th>
<th>SDDL</th>
<th>#patterns</th>
<th>X-rate (%)</th>
<th>Cmax(%)</th>
<th>BIST architecture</th>
<th>#trains</th>
<th>#LFSR stages</th>
<th>Gen.</th>
<th>seed gen. results</th>
<th>Area 1</th>
<th>Area 2</th>
<th>Area 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b15</td>
<td>1.9</td>
<td>2.8</td>
<td>784.6</td>
<td>217</td>
<td>8.9</td>
<td>228</td>
<td>8</td>
<td>9</td>
<td>98</td>
<td>200</td>
<td>796</td>
<td>564</td>
<td>564</td>
<td>564</td>
</tr>
<tr>
<td>b17</td>
<td>4.2</td>
<td>1.6</td>
<td>784.6</td>
<td>217</td>
<td>8.9</td>
<td>228</td>
<td>8</td>
<td>9</td>
<td>98</td>
<td>200</td>
<td>796</td>
<td>564</td>
<td>564</td>
<td>564</td>
</tr>
<tr>
<td>b18</td>
<td>43.8</td>
<td>7.6</td>
<td>33986.1</td>
<td>3293</td>
<td>94.4</td>
<td>703</td>
<td>60</td>
<td>384</td>
<td>2048</td>
<td>3129</td>
<td>956</td>
<td>564</td>
<td>564</td>
<td>564</td>
</tr>
<tr>
<td>b19</td>
<td>115.4</td>
<td>79.0</td>
<td>70038.2</td>
<td>6131</td>
<td>90.9</td>
<td>1298</td>
<td>120</td>
<td>608</td>
<td>2048</td>
<td>5890</td>
<td>954</td>
<td>564</td>
<td>564</td>
<td>564</td>
</tr>
</tbody>
</table>

Figure 4. SDL transition using the seeds ordered by ATPG, random method and the proposed method.

Table 4. CPU time of seed ordering.

<table>
<thead>
<tr>
<th>circuit</th>
<th>SDQL</th>
<th>PO (%)</th>
<th>SDDL</th>
<th>#patterns</th>
<th>X-rate (%)</th>
<th>Cmax(%)</th>
<th>BIST architecture</th>
<th>#trains</th>
<th>#LFSR stages</th>
<th>Gen.</th>
<th>seed gen. results</th>
<th>Area 1</th>
<th>Area 2</th>
<th>Area 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b15</td>
<td>0.6</td>
<td>0.0</td>
<td>0.6</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>2.3</td>
<td>2.3</td>
</tr>
<tr>
<td>b17</td>
<td>2.2</td>
<td>0.6</td>
<td>0.6</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>2.3</td>
<td>2.3</td>
</tr>
<tr>
<td>b18</td>
<td>19.1</td>
<td>0.6</td>
<td>19.1</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>2.3</td>
<td>2.3</td>
</tr>
<tr>
<td>b19</td>
<td>22.5</td>
<td>2.4</td>
<td>7.4</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>3.3</td>
<td>2.3</td>
<td>2.3</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Table 5. Seed selection results under SDL constraints: ATPG vs random vs proposed.

That is, 0% loss denotes the SDL value which can be obtained by selecting all seeds in the base seed set, and 0% loss denotes the SDL value which has α% smaller improvement than 0% loss SDL.

From Table 5, we observe that, even in the case of 0% loss (i.e., without sacrificing SDQL), the proposed method obtained up to 15% reduction in the number of selected seeds compared to other methods. This means that the proposed method can efficiently find unnecessary seeds which have no contribution to SDL in the base seed set. In case of 2% loss (i.e., if we are allowed to sacrifice SDQL by 2%), we can obtain significant reduction in seed count, and it can reach around 50% in average compared to other methods. The results show that the proposed ordering method can reduce test data volume in high quality delay test efficiently.

5 Evaluation of Mixed-Mode BIST and Base Seed Set Selection

In the previous section, we evaluated the effectiveness of the proposed method using (1) the base seed sets generated by timing-aware ATPG and (2) the BIST with d_i = 1 for all seed s (i.e., only one deterministic pattern is expanded from each seed). However, when a seed is loaded in the LFSR, we can apply some pseudo-random patterns and there is a possibility that they will detect more SDDs so that some of the seeds are not needed. Moreover, delay test quality of the seeds selected by the proposed method depends on the given base seed sets. Therefore, in this section, we investigate the following two questions for the same circuits used in Section 4.

- How do the pseudo-random patterns in the mixed-mode BIST contribute to test data volume reduction in high quality delay test?
- What is the impact of base seed sets on the final delay test quality after seed selection?

5.1 Mixed-Mode BIST

To evaluate the contribution of the pseudo-random patterns in the mixed-mode BIST to test data volume reduction in high quality delay test, we explored the three types of mixed-mode BIST approaches as follows.

Type I: d patterns are expanded from every seed s (i.e., 1 deterministic pattern and 1 pseudo-random pattern for every seed). d is set to 1, 2, 4 and 8.

Type II: d patterns are expanded only from the first selected seed
Table 6: Seed ordering and selection results for different mixed-mode BIST environments.

<table>
<thead>
<tr>
<th>circuit</th>
<th>SDQ constraint</th>
<th>mix-mode BIST</th>
<th>seeds</th>
<th>Aseeds (%)</th>
<th>#patterns</th>
<th>Apatterns (%)</th>
<th>ordering CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b18</td>
<td>0.05 VdD2 (40% loss for base case)</td>
<td>I (base case)</td>
<td>2173</td>
<td>2173</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2173</td>
<td>2173</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>1803</td>
<td>1803</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1572</td>
<td>1572</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>b19</td>
<td>0.05 VdD2 (40% loss for base case)</td>
<td>I (base case)</td>
<td>3219</td>
<td>3219</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>3219</td>
<td>3219</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>2340</td>
<td>2340</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1803</td>
<td>1803</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

s1, and 1 deterministic pattern is expanded from the other seeds. d is set to 1024, 2048 and 4096.

**Type III**: #d patterns are expanded only from the first two selected seed s1 and s2, 1 deterministic pattern is expanded from the other seeds. d is set to 1024, 2048 and 4096.

We applied the proposed method to the same base seed sets used in Section 4 in the above mixed-mode BIST environments. Table 6 shows the results of the seed selection method under a SDQ constraint for b18 and b19. The columns “#seeds” and “Aseeds” denote the number of selected seeds and the relative difference to “type I width d = 1 (base case)”, respectively. The columns “dp”, “ip”, “total” and “Apatterns” denote the number of deterministic patterns, random patterns, total patterns and the relative difference to the base case, respectively.

In each mixed-mode type, we can observe that there is a trade-off between the reduction in the number of selected seeds and the increase in the number of expanded patterns, which correspond to test time. In general, if d becomes large, the number of selected seeds is decreased while the number of expanded patterns is increased. However, in some cases (type II and III with d = 1024 for b19), we can obtain reduction both in the seed and pattern counts.

In comparison between type I and II, type II can generate more efficient pseudo-random patterns in terms of SDQ since type II achieved similar reduction in the seed count as type I with smaller increase in the pattern count to satisfy the same SDQ value. This shows that the long pseudo-random sequence expanded from one seed is more effective than the set of very short pseudo-random sequences expanded from every seed. In comparison between type II and III, type III can generate more efficient pseudo-random patterns compared to type II. This shows that the contribution of the pseudo-random sequence from one seed is saturated if it is too long.

There results suggest that it is worth using some pseudo-random patterns to minimize the seed count if the seeds are stored on-chip and test time is not expensive. However, we need a way to find a minimum seed set with value d for each seed under SDQ and test time constraints. This is one of our future works.

5.2 Base Seed Set

Since the delay test quality of selected seeds depends on base seed sets, in order to find a suitable base seed set, we applied the proposed method for several base seed sets generated from different ATPG patterns. In the experiment, we generated patterns using timing-aware ATPG and n-detect ATPGs for transition faults for n = 1, 2, 4, and 8. Table 7 shows the ATPG results and the base seed set generation results. For the seed generation, we assumed the same BIST architecture as shown in Table 3. We can observe that the timing-aware ATPG denoted “TA” in Table 3 always achieved higher transition fault coverage than n-detect ATPGs. In this case, we also prepared another timing-aware ATPG patterns denoted “TA-limit” in Table 3. “TA-limit” was generated by the same timing-aware ATPG as “TA” but it was terminated when the transition fault coverage became almost the same as n-detect ATPGs.

Table 8 shows the results of the seed selection for the five different base seed sets under a SDQ constraint. In the experiments, we assumed the mixed-mode BIST of type I with d = 1 (i.e., only the deterministic patterns are expanded from each seed). The columns “#seeds”, “Aseeds” and “CPU” denote the number of selected seeds, the relative difference to the 1-detect case and CPU time for seed ordering, respectively. From Table 8, we can observe that the base seed set generated from timing-aware ATPG “TA” produced the best results for all the circuits, and obtained up to 78% reduction in seed count compared to “1-detect” seed set. Except for “TA”, all the four base seed sets have almost same transition fault coverage. However, “TA-limit” is still the best among them with reasonable CPU time for the large two circuits, b18 and b19. These results show that timing-aware ATPG is suitable to obtain high quality delay tests.

6 Conclusions

In this paper, we have presented a seed ordering and selection method in LFSR-reseeding-based BIST to achieve high quality delay test. The proposed method selects seeds based on the gain in the sum of the longest path lengths sensitized by seeds, which is highly correlated with SDQ. Experimental results show that the proposed method intelligently selects seeds and obtains significant
Table 7. ATPG and seed generation results for different ATPG methods.

<table>
<thead>
<tr>
<th>circuit</th>
<th>ATPG results</th>
<th>seed gaps</th>
<th>results</th>
<th>SC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-circuit</td>
<td>0.8</td>
<td>78.6</td>
<td>266.8</td>
<td>0.16</td>
</tr>
<tr>
<td>4-circuit</td>
<td>1.5</td>
<td>78.6</td>
<td>266.8</td>
<td>1.02</td>
</tr>
<tr>
<td>8-circuit</td>
<td>3.0</td>
<td>78.6</td>
<td>266.8</td>
<td>1.02</td>
</tr>
<tr>
<td>TA</td>
<td>1.0</td>
<td>82.0</td>
<td>286.0</td>
<td>0.72</td>
</tr>
<tr>
<td>TAA</td>
<td>1.7</td>
<td>82.8</td>
<td>284.7</td>
<td>0.60</td>
</tr>
<tr>
<td>b16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-circuit</td>
<td>2.8</td>
<td>82.0</td>
<td>823.7</td>
<td>0.88</td>
</tr>
<tr>
<td>4-circuit</td>
<td>12.0</td>
<td>82.0</td>
<td>827.0</td>
<td>0.95</td>
</tr>
<tr>
<td>8-circuit</td>
<td>25.8</td>
<td>82.0</td>
<td>825.7</td>
<td>0.82</td>
</tr>
<tr>
<td>TA</td>
<td>9.2</td>
<td>85.6</td>
<td>744.8</td>
<td>1.35</td>
</tr>
<tr>
<td>TAA</td>
<td>7.9</td>
<td>81.8</td>
<td>818.0</td>
<td>0.93</td>
</tr>
<tr>
<td>b17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-circuit</td>
<td>11.7</td>
<td>78.8</td>
<td>369.8</td>
<td>1.69</td>
</tr>
<tr>
<td>4-circuit</td>
<td>22.7</td>
<td>78.8</td>
<td>368.4</td>
<td>4.08</td>
</tr>
<tr>
<td>8-circuit</td>
<td>45.8</td>
<td>78.8</td>
<td>365.4</td>
<td>1.98</td>
</tr>
<tr>
<td>TA</td>
<td>93.2</td>
<td>78.8</td>
<td>367.0</td>
<td>6.18</td>
</tr>
<tr>
<td>TAA</td>
<td>43.8</td>
<td>78.8</td>
<td>398.6</td>
<td>3.19</td>
</tr>
</tbody>
</table>

Table 8. Seed ordering and selection results for different base seed sets.

<table>
<thead>
<tr>
<th>base seed set</th>
<th>b15</th>
<th>b16</th>
<th>b17</th>
</tr>
</thead>
<tbody>
<tr>
<td>base seed set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 2SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>base seed set</th>
<th>b15</th>
<th>b16</th>
<th>b17</th>
</tr>
</thead>
<tbody>
<tr>
<td>base seed set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 2SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDOF constraint = 3SD,9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

reduction in seed count under SDOF constraint within a reasonable time. We also have explored several mixed-mode BST approaches and showed the contribution of pseudo-random patterns to test data volume reduction in high quality delay test. Furthermore, we have investigated the impact of base seed set using different ATPG methods and showed that the mixed-mode ATPG can lead to high delay test quality for the finally selected seed sets. One of the future works is to find a minimum seed set and the number of expanded patterns from each seed under SDOF and test time constraints in the mixed-mode BST.

Acknowledgments

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grant-in-Aid for Scientific Research (B) (No.20300318) and Young Scientists (B) (No.21700059). The authors would like to thank Prof. Seiji Kajihara and Prof. Yasuo Sato in Kyusyu Institute of Technology, and Prof. Satoshi Otake in Nara Institute of Science and Technology for their invaluable discussions.

References