IEEE 19th Asian Test Symposium (ATS'10), pp. 313-318, December 2010.

Seed Ordering and Selection for High Quality Delay Test

Tomokazu Yoneda†*, Michiko Inoue†*, Akira Taketani†* 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-reseeding-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, SDQM.*

1 Introduction

Nanometer technologies have led to 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 troublesome phenomenon in the field. It is well known that aging causes gradual delay increase and finally lead 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 tends 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], the 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-reseeding-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 filed and cause a failure of the device [4]. Secondly, only [13] considered the seed selection problem in LFSR-reseeding-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-reseeding-based BIST architecture as in [13]. Our contributions are summarized as follows.

- We present a seed ordering and selection technique for 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 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 show how the 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

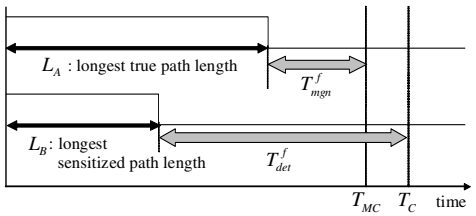


Figure 1. Timing relations for Longest true path length and longest sensitized path length.

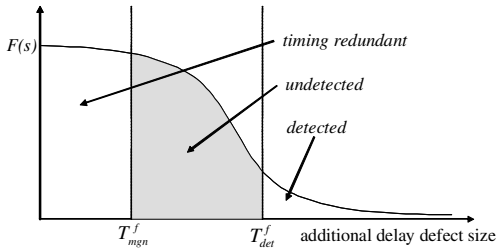


Figure 2. Delay defect distribution function and SDQL for a fault.

seed set on the final test quality of selected seeds.

The rest of the paper is organized as follows. We introduce SDQM and SDQL in Section 2, and Section 3 presents a seed ordering and selection method. Experimental results are shown in Section 4, and Section 5 evaluates the several mixed-mode BIST approaches and selection of base seed sets. Finally, Section 5 concludes this paper.

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 $T^f_{mgn} = T_{MC} - L_A$ is the minimum delay defect size that can affect system behavior and therefore should be detected. The difference $T^f_{det} = T_C - L_B$ 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 follow, where N is the total number of faults and $F(t)$ is a delay defect distribution function.

$$SDQL = \sum_{f \in N} \int_{T^f_{mgn}}^{T^f_{det}} F(t)dt \quad (1)$$

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

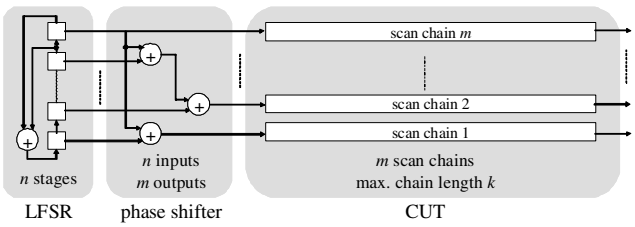


Figure 3. BIST architecture.

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.

SimulationBased

S_{base}: base seed set

S: ordered seed set (empty initially)

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

Table 1. CPU time of fault simulation.

circuit	#gates	#faults	#patterns	fault sim. CPU time(s) transition	TA
b15	8985	17329	727	0.8	47.4
b17	27766	65218	1375	5.2	194.4
b18	79400	172403	3293	49.2	1680.5
b19	152599	353301	6131	185.6	6415.8

In the ith iteration, it requires $|S_{base}|-(i-1)$ times fault simulation to obtain SDQL for each test pattern set expanded from $S+\{s\}$, which includes $\sum_{s_i \in S+\{s\}} d_i$ 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 SunFireX4100 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 SimulationBased 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 $F(t)$ and fault simulation, once we obtain the length of the longest path sensitized by each seed. Let l_f^S and l_f^s be the length of the longest path sensitized by the expanded patterns from seed set S and seed s for a fault f, respectively. Let L_S denote the sum of the longest sensitized path lengths for the expanded patterns from S. $L_{S+\{s\}}$ is obtained as follows.

$$L_{S+\{s\}} = L_S + \sum_{f \in N} \max(l_f^s - l_f^S, 0) \tag{2}$$

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

$$Gain_{S,s} = L_{S+\{s\}} - L_S = \sum_{f \in N} \max(l_f^s - l_f^S, 0) \tag{3}$$

The proposed method selects a seed s with the largest $Gain_{S,s}$ 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_{det}^f. 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
S: ordered seed set (empty initially)

1. For each $s \in S_{base}$, run fault simulation for the expanded patterns from s to obtain SDQL and l_f^s.
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 $Gain_{S,s}$.

Table 2. Characteristics of benchmark circuits.

circuit	#gates	#FFs	#faults	B in $F(t)$
b15	8985	417	17329	1.19
b17	27766	1317	65218	1.19
b18	79400	3020	172403	0.71
b19	152599	6042	353301	0.71

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

In Step 1, we run fault simulation to obtain l_f^s since TetraMAX can provide it through timing-aware fault simulation on and it is independent of delay defect distribution function $F(t)$. If $F(t)$ is available, then we can obtain a SDQL value for each seed as a by-product 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 $F(t)$ 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 ITC'99 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 mode. We can provide a delay defect distribution function $F(t)$ to TetraMAX to calculate SDQL in the form described as the following equation.

$$F(t) = A \cdot e^{-Bt} + C \tag{4}$$

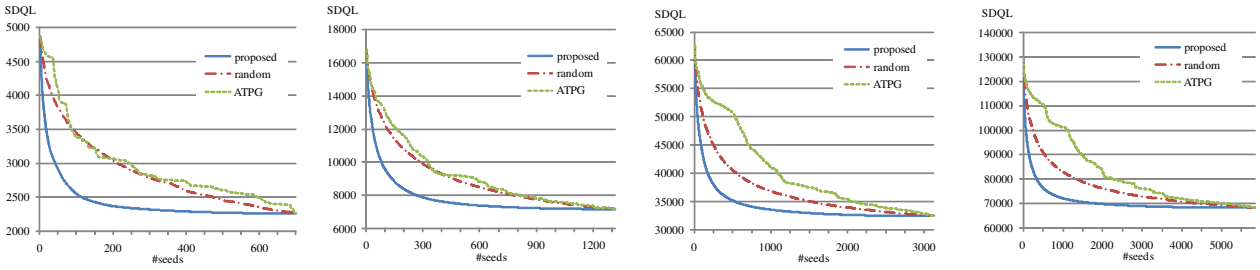
In all the experiments in this section, we set $A = 1, C = 0$, and set B so that $F(T_{MC}) = 0.1$ holds where T_{MC} denotes the system clock timing of the circuit. The values of B are shown in the column "B in $F(t)$" in Table 2.

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 we assumed, and the base seed set generation results. "X ratio" and "C_{max}" denote the average percentage of X's and the maximum number of specified bits in the generated patterns, respectively. "#schains" and "c_sep." 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 ordering (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 "fsim", "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.

circuit	ATPG results TGT(m)	FC(%)	SDQL	#patterns	X ratio(%)	C_{max}(bit)	BIST architecture #schains	#LFSR stages	c_sep.	seed gen. results #seeds	SC(%)
b15	1.9	82.0	2498.0	727	87.9	258	8	96	2048	700	96.3
b17	9.2	86.2	7841.8	1375	90.2	655	26	240	2048	1319	95.9
b18	43.8	79.7	33986.1	3293	94.4	703	60	384	2048	3129	95.0
b19	115.4	79.0	70768.2	6131	96.9	1258	120	608	2048	5850	95.4



(a) b15 (b) b17 (c) b18 (d) b19

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

Table 4. CPU time of seed ordering.

circuit	fsim(m)	other(m)	total(m)
b15	0.6	0.0	0.6
b17	2.2	0.1	2.3
b18	19.1	0.6	19.7
b19	72.5	2.4	74.9

Table 5 shows the results of the seed selection based on the three different ordering methods: (1) ordered by ATPG (atpg), (2) random ordering (rand) and (3) proposed ordering (pro) under several SDQL constraints to be satisfied. We set four SDQL constraints for each circuit: (1) 0% loss, (2) 2% loss, (3) 5% loss and (4) 10% loss. Let y_{base} and y_{empty} denote the SDQLs for the base seed set and empty seed set, respectively. Then, α% SDQL loss $y_{\alpha\%loss}$ is given as follows.

$$y_{\alpha\%loss} = y_{base} + (y_{empty} - y_{base}) \times \frac{\alpha}{100} \quad (5)$$

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

From Table 5, we can 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 SDQL 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 (i.e., only one deterministic pattern is expanded from each seed). However,

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

circuit	SDQL constraint		#seeds selected atpg	rand	pro	relative diff(%) Δatpg	Δrand
b15	2262.86	(0% loss)	700	700	593	-15.3	-15.3
	2314.93	(2% loss)	692	636	308	-55.5	-51.6
	2393.04	(5% loss)	663	566	177	-73.3	-68.7
	2523.23	(10% loss)	590	457	107	-81.9	-76.6
b17	7164.50	(0% loss)	1318	1319	1243	-5.7	-5.8
	7357.64	(2% loss)	1202	1144	629	-47.7	-45.0
	7647.35	(5% loss)	969	954	385	-60.3	-59.6
	8130.21	(10% loss)	756	728	240	-68.3	-67.0
b18	32476.02	(0% loss)	3129	3129	2879	-8.0	-8.0
	33080.83	(2% loss)	2925	2549	1361	-53.5	-46.6
	33988.04	(5% loss)	2433	1966	789	-67.6	-59.9
	35500.06	(10% loss)	1946	1347	450	-76.9	-66.6
b19	68390.51	(0% loss)	5850	5850	5323	-9.0	-9.0
	69609.96	(2% loss)	5379	4835	2185	-59.4	-54.8
	71439.14	(5% loss)	4258	3689	1194	-72.0	-67.6
	74487.78	(10% loss)	3358	2458	655	-80.5	-73.4

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 $d-1$ pseudo-random patterns 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 mix-mode BIST environments.

circuit	SDQL constraint	mix-mode BIST type	mix-mode BIST d	#seeds	Δseeds (%)	#patterns dp	#patterns rp	#patterns total	Δpatterns (%)	ordering CPU time (m)
b18	32476.02	I	(base case) 1	2879	-	2879	0	2879	-	19.7
	(0% loss		2	2173	-24.5	2173	2173	4346	51.0	30.8
	for base case)		4	1803	-37.4	1803	5409	7212	150.5	57.7
			8	1572	-45.4	1572	11004	12576	336.8	118.5
		II	1024	2144	-25.5	2144	1023	3167	10.0	26.3
			2048	1980	-31.2	1980	2047	4027	39.9	33.1
			4096	1960	-31.9	1960	4095	6055	110.3	44.5
		III	1024	1749	-39.2	1749	2046	3795	31.8	32.1
			2048	1755	-39.0	1755	4094	5849	103.2	44.5
			4096	1520	-47.2	1520	8190	9710	237.3	70.9
b19	68390.51	I	(base case) 1	5323	-	5323	0	5323	-	74.9
	(0% loss		2	4249	-20.2	4249	4249	8498	59.6	123.4
	for base case)		4	3483	-34.6	3483	10449	13932	161.7	237.0
			8	2854	-46.4	2854	19978	22832	328.9	493.2
		II	1024	3897	-26.8	3897	1023	4920	-7.6	90.0
			2048	3687	-30.7	3687	2047	5734	7.7	102.8
			4096	3595	-32.5	3595	4095	7690	44.5	133.2
		III	1024	3274	-38.5	3274	2046	5320	-0.1	105.3
			2048	3219	-39.5	3219	4094	7313	37.4	135.4
			4096	2820	-47.0	2820	8190	11010	106.8	195.9

s_1, 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 s_1 and S_2, and 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 SDQL constraint for $b18$ and $b19$. The columns "#seeds" and "Δseeds" denote the number of selected seeds and the relative difference to "type I widh $d = 1$ (base case)", respectively. The columns "dp", "rp", "total" and "Δpatterns" 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 relation 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 are 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 SDQL since type II achieved similar reduction in the seed count as type I with smaller increase in the pattern count to satisfy the same SDQL 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_i for each seed under SDQL 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 order for fair comparison, we also prepared another timing-aware 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 SDQL 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", "Δseeds" 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 test.

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 SDQL. Experimental results show that the proposed method intelligently selects seeds and obtains significant

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

	b15			b17			b18			b19		
	SDQL constraint = 2555.59 (0% loss for 1-detect)			SDQL constraint = 7854.65 (0% loss for 1-detect)			SDQL constraint = 35107.89 (0% loss for 1-detect)			SDQL constraint = 74411.53 (0% loss for 1-detect)		
base seed set	#seeds	Δseeds (%)	CPU (m)	#seeds	Δseeds (%)	CPU (m)	#seeds	Δseeds (%)	CPU (m)	#seeds	Δseeds (%)	CPU (m)
1-detect	464	-	0.4	942	-	1.6	1865	-	12.1	2933	-	39.5
2-detect	167	-64.0	0.7	543	-42.4	3.4	1130	-39.4	24.7	2021	-31.1	87.5
4-detect	152	-67.2	1.4	453	-51.9	6.9	828	-55.6	50.4	1640	-44.1	175.5
8-detect	139	-70.0	2.8	432	-54.1	13.7	749	-59.8	101.3	1381	-52.9	353.7
TA	99	-78.7	0.6	304	-67.7	2.3	509	-72.7	19.7	662	-77.4	74.9
TA_limit	229	-50.6	0.5	449	-52.3	1.8	683	-63.4	17.7	825	-71.9	70.3

Table 7. ATPG and seed generation results for different ATPG methods.

		ATPG results				seed gen. results	
circuit	ATPG	TGT (m)	FC (%)	SDQL	#patterns	#seeds	SC (%)
b15	1-detect	0.4	78.2	2668.0	508	488	96.1
	2-detect	0.8	78.2	2666.2	1016	976	96.1
	4-detect	1.5	78.2	2666.2	2032	1952	96.1
	8-detect	3.0	78.2	2666.2	4064	3904	96.1
	TA	1.9	82.0	2498.0	727	700	96.3
	TA_limit	1.7	78.4	2584.7	627	600	95.7
b17	1-detect	2.8	82.9	8527.9	966	951	98.5
	2-detect	6.1	82.3	8571.5	2024	1997	98.7
	4-detect	12.0	82.3	8572.5	4057	4002	98.6
	8-detect	23.8	82.3	8576.7	8120	8016	98.7
	TA	9.2	86.2	7841.8	1375	1319	95.9
	TA_limit	7.9	82.9	8188.0	1114	1060	95.2
b18	1-detect	11.7	74.8	36648.1	1989	1920	96.5
	2-detect	22.7	75.1	36488.3	4042	3896	96.4
	4-detect	45.8	75.4	36315.1	8198	7933	96.8
	8-detect	93.2	75.5	36276.6	16416	15844	96.5
	TA	43.8	79.7	33986.1	3293	3129	95.0
	TA_limit	37.8	74.9	35624.2	2909	2797	96.2
b19	1-detect	26.7	73.6	76644.7	3144	3004	95.6
	2-detect	52.9	73.4	76845.4	6766	6466	95.6
	4-detect	97.4	73.4	76839.7	13533	12933	95.6
	8-detect	204.2	73.4	76839.7	27069	25859	95.5
	TA	115.4	79.0	70768.2	6131	5850	95.4
	TA_limit	104.6	73.6	74453.7	5529	5361	97.0

reduction in seed count under SDQL constraint within a reasonable time. We also have explored several mixed-mode BIST 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 the timing-aware 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 SDQL and test time constraints in the mixed-mode BIST.

Acknowledgments

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research (B)(No.20300018) 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 Ohtake in Nara Institute of Science and Technology for their invaluable discussions.

References

[1] W. Wang, V. Reddy, A. Krishnan, R. Battikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Trans. Device and Materials Reliability*, vol. 7, pp. 509–517, Dec. 2007.

[2] T. W. Chen, K. Kim, Y. M. Kim, and S. Mitra, "Gate-oxide early life failure prediction," in *Proc. VLSI Test Symposium*, pp. 111–118, Apr. 2008.

[3] Y. Sato, S. Kajihara, Y. Miura, T. Yoneda, S. Ohtake, M. Inoue, and H. Fujiwara, "A circuit failure prediction mechanism (DART) for high field reliability," in *Proceedings of International Conference on ASIC, IEEE*, pp. 581–584, 2009.

[4] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Timing-based delay test for screening small delay defects," in *Proc. Design Automation Conference*, pp. 320–325, July 2006.

[5] X. Lin, K. H. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware ATPG for high quality at-speed testing of small delay defects," in *Proc. Asian Test Symposium*, pp. 139–146, Nov. 2006.

[6] Synopsys, *TetraMAX ATPG User Guide, Version C-2009.06-SP2*, September 2009.

[7] A. Uzzaman, M. Tegethoff, B. Li, K. M. Cauley, S. Hamada, and Y. Sato, "Not all delay tests are the same - SDQL model shows true-time," in *Proceedings of the 15th Asian Test Symposium*, (Washington, DC, USA), pp. 147–152, IEEE Computer Society, 2006.

[8] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern grading and pattern selection for small-delay defects," in *Proceedings of VLSI Test Symposium*, (Los Alamitos, CA, USA), pp. 233–239, IEEE Computer Society, 2008.

[9] Z. Wang and K. Chakrabarty, "Test-quality/cost optimization using output-deviation-based reordering of test patterns," *IEEE Transactions on Computer-Aided Design on Integration Circuits and Systems*, vol. 27, no. 2, pp. 352–365, 2008.

[10] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Interconnect-aware and layout-oriented test-pattern selection for small-delay defects," in *Proc. International Test Conference*, Oct. 2008.

[11] K. Peng, M. Yilmaz, M. Tehranipoor, and K. Chakrabarty, "High-quality pattern selection for screening small-delay defects considering process variations and crosstalk," in *Proc. Design, Automation and Test in Europe*, Mar. 2010.

[12] K. Peng, J. Thibodeau, M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "A novel hybrid method for SDD pattern grading and selection," in *Proc. VLSI Test Symposium*, pp. 45–50, Apr. 2010.

[13] M. Yilmaz and K. Chakrabarty, "Seed selection in LFSR-reseeding-based test compression for the detection of small-delay defects," in *Proc. Design, Automation and Test in Europe*, pp. 1488–1493, Apr. 2009.

[14] S. Hellebrand, H. G. Liang, and H. J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters," in *Proc. International Test Conference*, pp. 778–784, 2000.

[15] A. Jas, C. V. Krishna, and N. A. Touba, "Hybrid BIST based on weighted pseudo-random testing: a new test resource partitioning scheme," in *Proc. VLSI Test Symposium*, pp. 2–8, 2001.

[16] A. A. Al-Yamani, S. Mitra, and E. J. McCluskey, "BIST reseeding with very few seeds," in *Proc. VLSI Test Symposium*, pp. 69–74, Apr. 2003.

[17] Y. Sato, S. Hamada, T. Maeda, A. Takatori, and S. Kajihara, "Evaluation of the statistical delay quality model," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, (New York, NY, USA), pp. 305–310, ACM, 2005.

[18] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proc. European Test Conference*, pp. 237–242, 1991.