A-1

Acceleration of Seed Ordering and Selection For High Quality VLSI Delay Test

Ratna Aisuwarya [*], Yuta Yamato [*], Tomokazu Yoneda [*], Kazumi Hatayama[*], Michiko Inoue [*]

Abstract

Seed ordering and selection is a key technique to provide high-test quality with limited resources in Built-In Self Test (BIST) environment. We present a hard-to-detect delay fault selection method to optimize the computation time in seed ordering and selection processes. This selection method can be used to select faults for test generation when it is impractical to target all delay faults resulting large test pattern count and long Computation time. Three types of selection categories are considered, ranged in the number of seeds it produced, which is useful when we consider computing resources, such as memory and storage. We also evaluate the impact of the selection method in mixed-mode BIST when seed are expanded to more patterns, and evaluate the statistical delay quality level (SDQL) with the original work. Experimental results show that our proposed method can significantly reduce computation time while slightly sacrificing test quality.

1. Introduction

New challenges emerge for testing field engineering, as VLSI technologies are scale down to nanometer. This leads to the increasing probability of timing-related defects to occur. Small Delay Defects (SDDs), which caused by resistive opens, resistive shorts and some other process variations might significantly impact the overall product quality especially for the 45nm scales and below when the sensitized path is a critical path. Thus, serious consideration are growing rapidly in targeting these SDDs to minimizes the test escape rate as well as improves defect coverage in some extend of in-field reliability [1]. In order to detect SDDs, propagation through long path is required in test application, since the minimum slack that such a small delay produced cannot exceed the slack in shorter paths. However, Traditional Automatic Test Pattern generation (ATPG) tools tend to generate a pattern sequences that targets the fault along the path that has the largest timing slack, which is the short path [2].

Therefore, commercial timing-aware ATPG tools, e.g., Synopsys TetraMAX have been developed to overcome the lack of coverage of traditional timing-unaware ATPGs [3]. In spite of the ability to targets each undetected fault along paths with minimal timing slack, they result in significantly large CPU runtime and pattern count. This means increasing in manufacturing cost and resources, and it is not practical to apply for the testing environment where the specification is limited. To avoid the high cost and complexity during testing, novel method are required to reduce the pattern count but effectively target SDDs.

Seed ordering algorithm based on exploiting the algebraic properties of the pseudorandom pattern generator (PRPG) to increase the number of patterns generated from one seed, can be efficient method to reduce the seed storage [4]. LFSR reseeding –based BIST was first introduced by Koenemann in [5] as a technique for coding test patterns into PRPGs. In terms of targeting SDDs the proposed method in [6] considered the test compression for seed selection problem in LFSR-reseeding-based BIST, however it only utilized one seed for one pattern, or deterministic pattern in the compression method. Since, we can apply some pseudo-random patterns combined with deterministic patterns (mixed-mode BIST), more seeds can be reduced and there is a chance that the patterns will increase the detection coverage of SDDs. Recent seed ordering and selection method, proposed by Yoneda et al. [7] selects seeds based on the gain in the sum of the longest path lengths sensitized by seeds, which is correlated with statistical delay quality level (SDQL). Experiment results show that this method can obtain significant seed count reduction under several mixed-mode BIST approaches. Yet, the method is still considered to be time consuming, since it generated all patterns from the fault list, which later encoded into seeds. This is why we need another solution considered test time constraints as a compromise between the SDQL and the resources.

2. Statistical Delay Quality Level (SDQL)

We applied statistical delay quality level (SDQL) proposed by Sato et al, in [8]. SDQL is modeled in measurable indicator of delay quality. Let f be a delay fault at some signaled f line. Let T^f_{mgn} denote the difference between system clock timing and the length of the longest path passing through f, and T_{det} denote the difference between test timing and the length of the longest sensitized path passing through f by a given test set. Let $F(s)$ be a delay defect distribution. SDQL represents the amount of delay defects escaped to be detected by the test set, and can be expressed by:

$$SDQL = \int_{T^f_{mgn}}^{T^f_{det}} F(S) \quad (1)$$

Since, the SDQL show the amount of test escape, we tried to reduce the value during testing.

3. Seed Ordering

The seed ordering proposed in [7] uses the sum of the longest sensitized path lengths for all the faults. First, for

* Nara Institute of Science and Technology

each fault, the length of the longest path sensitized by each seed is obtained.. Let l_f^{st} and l_f^s be the length of the longest path sensitized by the pseudo-random pattern which is expanded from seed set *St* and seed *S* for a fault *f*. Sum of the longest sensitized path length for the expanded patterns from *S* is denoted by L_s. Then, the sum of the longest sensitized path lengths when *S* is added to *St* is obtained as follows.

$$L_{St+\{S\}} = L_s + \sum_{f \in N} max\left(l_f^{St} - l_f^S, 0\right) \quad (2)$$

Then, the *Gain* is calculated as $L_{St+\{S\}} - L_s$. *Gain* is calculated for each seed *S* in the base seed set *St*. Seed with the largest *Gain* values will be removed from the base seed set S_{base} and included in the ordering set *St.* Fault simulation is applied for each seed to obtained l_f^S. The first seed that is selected into the ordering set is selected based on SDQL. However, the remaining seeds will be selected based on the *Gain* values.

4. Proposed Hard-to-detect Fault Selection

In this section, we describe the proposed hard-to-detect fault selection method. These kinds of faults are faults with relatively detected by few test patterns. It is very important in order to save computation time on the timing-aware ATPG. Since, it wasted a lot of time in sensitized faults that do not contribute to SDDs coverage resulting in a large number of test patterns. Furthermore, a large number of faults mean a large number of long paths to sensitize. Therefore, our idea is to reduce the number of faults for generating patterns in timing-aware ATPG.

The proposed method selects faults based on detection count constraint. We set *h* values as 1, 3, and 5, in other words in fault simulation, we count the number of times that a fault is detected by pattern set and create subsets of h (up to 1 time, 3 times, and 5 times). Then, for each fault that is falls into these categories will be included. The fault selection procedure is as follows:

1. Generates patterns based on transition fault model with all faults set.
2. Run fault simulation using the above patterns.
3. Create subsets of fault list based on *h* as result of fault simulation.

Furthermore, the hard-to-detect subset fault list will be used in the timing-aware ATPG to generate patterns for targeting SDDs. Therefore it can generate faster since the fault list base is reduced.

5. Experimental Results

5.1 Seed Ordering

In order to evaluate the effectiveness of our proposed method, we conducted experiments using several ITC '99 benchmark circuits. Table 1. Show the characteristics of the circuits used in the experiments.

Table 1. Characteristics of benchmark circuit

Circuit	#gates	#FFs	B in *F(t)*
b15	8985	417	1.19
b17	27766	1317	1.19
b18	79400	3020	0.71
b19	152599	6042	0.71

Synopsis TetraMAX ATPG with Small Delay Defect Test mode were used with provided delay defect distribution function *F(t)* in order to calculate SDQL values with the following equation.

$$F(t) = A \cdot e^{-Bt} + C \quad (3)$$

We set *A*, and *C* to 1 and 0 respectively, then calculated *B* so that function of system clock timing of the circuit, $F(T_{MC}) = 0.1$ holds. The calculated values of *B* are shown in the Table 1.

Table 2. Fault base set for generation pattern

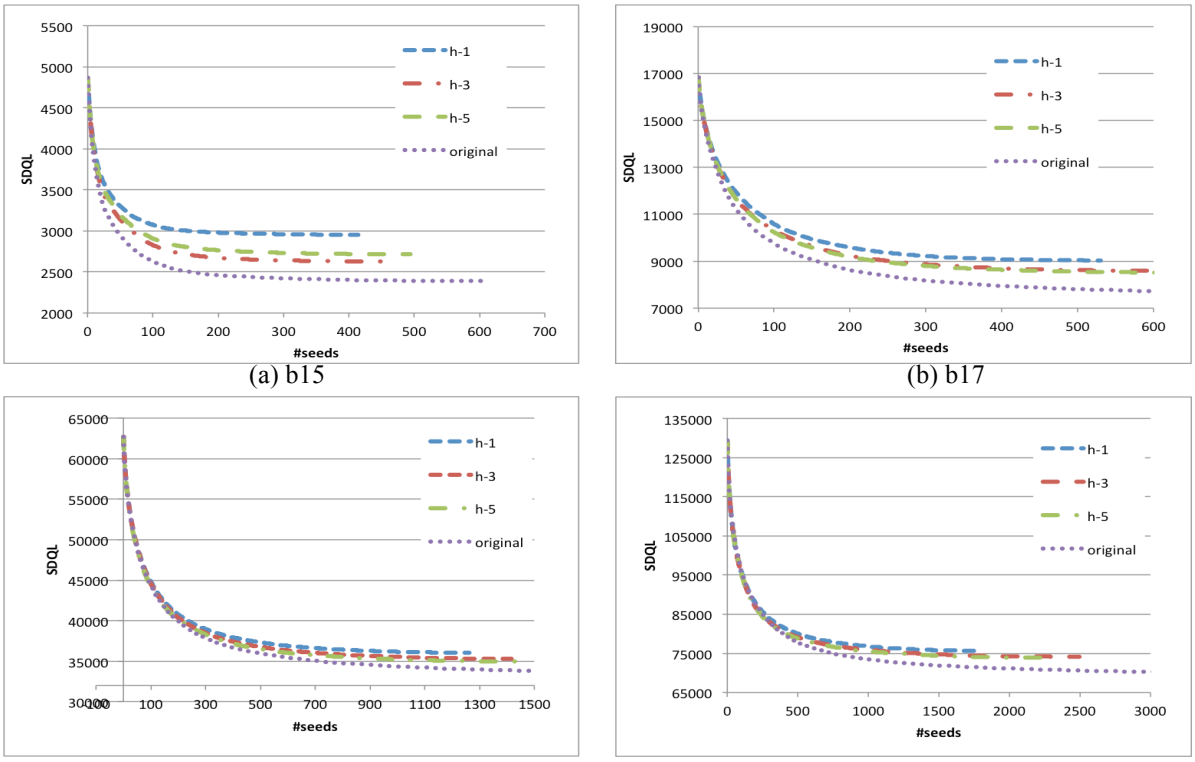
Circuit	#Faults			
	h-1	h-3	h-5	original
B15	3,930	6,423	7,245	17,329
B17	15,868	25,942	29,062	65,218
B18	39,265	60,812	67,453	172,403
B19	78,454	121,472	134,392	353,301

To evaluate the test quality of our proposed method in the ordering process, we compare hard-to-detect fault selection base with three different constrains (*h*-1, *h*-3, and *h*-5) to the original work which included all fault in the patterns generation. The fault base set after the selection process applied is shown in table 2. Test patterns with unspecified bits (X's) are generated by timing-aware ATPG using the faults in table 2, then these patterns encoded into a base seed set. We can see in the results, the selection method significantly reduced the number of faults, which mean less number of long path to sensitized.

Seed generation results are shown in Table 3. "#schains" denote the number of scan chains and 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 the Table 3, we compared the proposed hard-to-detect fault selection method with the original methods targeting all faults during fault simulation. The number of expanded patterns from seed S_{ii} is set to 1 (d =1) for all seeds in the base seed set. We can observe that the proposed method obtained significant reduction in the number of seeds compared to original methods. Figure 2 shows the SDQL transitions using the selection categories compare to the original. We can observe from the figures that our proposed method even with smaller number of seed can obtained effective SDQL coverage. Thus, we have to sacrifice SDQL coverage varied among the selection categories

Table 3. Seed generation results for timing-aware patterns

Circuit	BIST Architecture		#patterns				#seeds			
	#LFSR	#schains	h-1	h-3	h-5	original	h-1	h-3	h-5	original
B15	96	8	490	543	568	727	478	528	553	700
B17	240	26	735	935	978	1,375	706	891	931	1,319
B18	384	60	1,479	1,690	1,760	3,293	1,415	1,609	1,689	3,129
B19	608	120	2,006	2,681	2,908	6,131	1,906	2,560	2,784	5,850



(a) b15 (b) b17

(c) b18 (d) b19

Figure 2. SDQL transition of the ordered seeds from the proposed selection and original

Table 4. Seed ordering results for different mixed-mode BIST

Circuit	Type	d	SDQL				Ordering Computation time (m)			
			h-1	h-3	h-5	original	h-1	h-3	h-5	original
b18	I	1	34,894.94	34,165.45	33,789.73	32,510.24	6.21	7.14	7.66	13.48
		2	34,735.76	34,030.25	33,665.68	32,428.76	9.49	10.87	11.78	20.77
		4	34,556.28	33,866.16	33,516.78	32,315.38	17.14	19.72	21.02	37.95
		8	34,390.90	33,731.80	33,389.95	32,203.47	34.34	39.71	41.66	76.41
	II	1024	34,685.55	34,012.80	33,640.54	32,426.86	10.09	11.13	11.31	17.24
		2048	34,600.98	33,937.69	33,564.83	32,384.50	14.12	15.06	15.50	21.15
		4096	34,483.89	33,819.18	33,462.49	32,323.50	21.96	22.88	23.30	28.94
	III	1024	34,599.11	33,943.39	33,575.58	32,387.12	13.92	15.04	15.48	21.15
		2048	34,477.68	33,831.20	33,478.04	32,324.42	21.79	22.90	23.34	29.63
		4096	34,330.71	33,687.38	33,351.41	32,232.54	37.73	38.88	39.23	46.09
b19	I	1	74,077.77	72,316.43	72,123.42	68,274.49	17.27	22.99	25.01	50.70
		2	73,755.27	72,093.86	71,891.89	68,156.53	27.73	37.09	39.72	80.03
		4	73,406.32	71,798.39	71,595.62	67,944.48	51.51	68.79	74.57	153.38
		8	73,050.98	71,530.23	71,310.90	67,688.74	103.17	139.48	151.21	315.89
	II	1024	73,772.55	72,077.48	71,913.30	68,137.08	26.23	31.77	32.49	58.94
		2048	73,592.33	71,950.11	71,786.90	68,069.92	35.03	40.34	42.32	68.00
		4096	73,338.09	71,760.22	71,605.27	67,950.66	52.52	57.84	59.83	85.33
	III	1024	73,610.69	71,972.19	71,793.07	68,072.68	34.83	40.46	42.41	67.73
		2048	73,363.38	71,789.97	71,625.00	67,970.68	52.78	58.69	59.82	86.23
		4096	73,027.66	71,542.90	71,378.98	67,804.02	88.37	92.96	95.94	121.05

5.2 Mixed-mode BIST

In mixed-mode BIST contribution of pseudo-random patterns to delay test quality is evaluated. Three types of mixed-mode BIST approaches are used.

Type I: for every seed S, d patterns are expanded; d is set to 1, 2, 4, and 8.

Type II: d patterns are expanded only from the first selected seed S_1, and 1 deterministic pattern is expanded from the other seeds. d is set to 1024, 2048, and 4096.

Type III: the first two seeds S_1 and S_2 will be expanded

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

Table 4 shows seed ordering results in mixed-mode BIST for *b*18 and *b*19. We can observe that when *d* become large, SDQL value is decrease and the number of test pattern is increased. This also correlated to test time. Furthermore, we compare the results between type I, type II and type III. In this case, type III generate more pseudo-random pattern compared to two other types. The results shows that the long pseudo-random patterns expanded from one seed are more effective than the very short expanded pattern for every seed in type I.

In comparison between our proposed method and the original work, our proposed method can achieve reduction in SDQL values slightly different compare to the original but significant gap in computation time. This mean, under the same test time constraint we can expand more patterns with our proposed method to achieve more coverage.

5.3 Computation Time

We evaluate the optimization in computation time for our proposed method. In the experiments, two additional times is needed to get the hard-to-detect fault base set. First, computation time for generating patterns for transition delay fault. Second, Computation time for fault simulation to create fault list based on the detection counts. Table 5 summarizes Computation times for fault selection.

To compare between the original work and our proposed method, we evaluate each computation cost in:

1. Fault selection. (Time to select faults in the proposed method).
2. ATPG. (Time for generating patterns).
3. Ordering. (Time for fault simulation in seed ordering).

Since, fault selection only applied in our proposed method, for original work we set this time to 0. Table 6 shows Computation time for all process. The results show that the original work consumed longer time due to the fact that it targeted all faults during timing-aware ATPG. Therefore, if the test time is expensive, our proposed method can be applied to accelerate testing time.

Table 5. Computation time for fault selection

Circuit	P.generation (m)	Fsim (m)
b15	0.11	0.07
b17	1.08	0.41
b18	4.49	1.46
b19	7.25	4.49

6. Conclusions

We have presented a hard-to-detect fault selection method in the seed ordering and selection for high quality delay test. The proposed method selects fault based on detection count constraint. We set several constraints and relax the hardness and evaluate the effectiveness compare to the original work. Experimental results show that the proposed method significantly reduced seed count in the base seed set. We evaluate the effectiveness based on SDQL values, and found that the delay test quality is slightly decreased. However, if we expand more patterns from seeds in the mixed-mode BIST environment we can increase the coverage. Furthermore, our method can obtain significant test timing reduction.

Table 6. Computation time

Circuit	Type	h-1	h-3	h-5	original
b15	Selection	0.18	0.18	0.18	0
	ATPG	0.19	0.33	0.37	1.29
	Ordering	0.31	0.32	0.3	0.47
	Total (m)	0.68	0.83	0.85	1.76
b17	Selection	1.49	1.49	1.49	0
	ATPG	1.71	2.69	2.88	5.95
	Ordering	0.86	1.06	1.14	1.7
	Total (m)	4.06	5.24	5.51	7.65
b18	Selection	5.95	5.95	5.95	0
	ATPG	7.59	8.88	9.59	27.08
	Ordering	6.21	7.14	7.66	13.48
	Total (m)	19.75	21.97	23.2	40.56
b19	Selection	11.75	11.75	11.75	0
	ATPG	14.85	20.75	23.81	70.87
	Ordering	17.27	23.99	25.01	50.7
	Total (m)	43.87	56.49	60.57	121.57

7. References

[1] R.Mattiuzo, D. Appello C. Allsup, "Small Delay Defect Testing," http://www.tmworld.com/article/CA6660051.html, Test &Measurement World, 2009.

[2] 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.

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

[4] A. A. Ahmad, M. Subhasish, and M.J. Edward, "Optimized Reseeding by Seed Ordering and Encoding" in *IEEE on Computer-aided Design of Integrated Circuits and Systems*", vol. 24, pp 264-270, Feb. 2005.

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

[6] 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.

[7] T.Yoneda, M. Inoue, A. Taketani, H. Fujiwara, "Seed ordering and selection for high quality delay test" in *Proc. Asian Test Symposium,* pp. 313-318, Dec. 2010.

[8] 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 Pasific Design Automation Conference,* (New York, NY, USA), pp. 305-310, ACM, 2005.