Improving The Test Quality for Scan-based BIST Using A General Test Application Scheme

Huan-Chih Tsai Kwang-Ting Cheng
Department of ECE
University of California
Santa Barbara, CA 93106

Sudipta Bhawmik
Bell Laboratories
Lucent Technologies
Princeton, NJ 08542

Abstract

In this paper, we propose a general test application scheme for existing scan-based BIST architectures. The objective is to further improve the test quality without inserting additional logic to the Circuit Under Test (CUT). The proposed test scheme divides the entire test process into multiple test sessions. A different number of capture cycles is applied after scanning in a test pattern in each test session to maximize the fault detection for a distinct subset of faults. We present a procedure to find the optimal number of capture cycles following each scan sequence for every fault. Based on this information, the number of test sessions and the number of capture cycles after each scan sequence are determined to maximize the random testability of the CUT. We conduct experiments on ISCAS89 benchmark circuits to demonstrate the effectiveness of our approach.

1 Introduction

Agrawal *et al.* classified the test scheme of scan-based BIST as either *test-per-clock* or *test-per-scan* [1]. In *test-per-clock* BIST, a test vector is applied and its response is compressed *every clock cycle*. The examples of test-per-clock BIST are BILBO-based design [2] and circular BIST [3]. In *test-per-scan* BIST, a test vector is applied and its response is captured into the scan chains *only after the test is scanned into the scan chains*. The well-known STUMP architecture [4] falls into this category.

There are tradeoffs between these two test application schemes in terms of area overhead, performance degradation, and test application time. The test-per-clock BIST typically has shorter test time but incurs higher area and performance overheads than test-per-scan BIST. Recently, PSBIST has been proposed to incorporate partial-scan and pseudo-random testing into the scan-based BIST [5]. The test application scheme of PSBIST is a combination of test-per-clock BIST and test-per-scan BIST. It results in shorter test time without increasing the area and performance overheads comparing to the conventional test-per-scan BIST. This work is based on the PSBIST architecture and we'll give an overview of the PSBIST in the next section. Unlike PSBIST (or any other test-per-scan BIST) which always applies a *single* capture cycle after scanning in a new test pattern, we propose to apply *multiple* capture cycles after each scan sequence. It has been observed that applying a different number of capture cycles per scan can help to detect a different subset of faults. We'll illustrate this concept and discuss the motivation in Section 3. We then propose a general test application scheme — multiple test sessions with a different number of capture cycles per scan in each session — for PSBIST. A procedure of finding the optimal parameters (i.e. the number of test sessions and the number of capture cycles per scan in each session) of the test scheme for a given circuit is described in Section 4. The proposed scheme has been implemented and experimented on ISCAS89 benchmark circuits using an industrial scan-based BIST system, **psb2**. The results presented in Section 5 illustrate the effectiveness of the proposed approach.

2 PSBIST

Fig. 1 is the schematic of PSBIST which is similar to STUMP [4]. The BIST capability are incorporated into the circuit through the following steps:

1. Replace crucial flip-flops with scan flip-flops, then connect them into the scan chains. For full-scan BIST, all flip-flops are replaced; for partial-scan BIST, the crucial flip-flops are defined as those, if scanned, that will remove all sequential loops with length *greater than one*.

2. Optionally, add test points (control points or observation points) to increase the random testability of the circuit.

3. Add a Test Pattern Generator (TPG) add an Output Data Compactor (ODC). The TPG consists of a Linear Feedback Shift Register (LFSR) and a Phase Shifter (PS). The PS is used to avoid the structure dependency among the outputs of TPG [6]. The ODC consists of a Multiple Input Signature Register (MISR) and a Space Compactors (SC) [7].

During the test, the patterns are continuously applied to the network N from the primary inputs and through the scan chains. The responses at the primary outputs are compressed

DAC 99, New Orleans, Louisiana

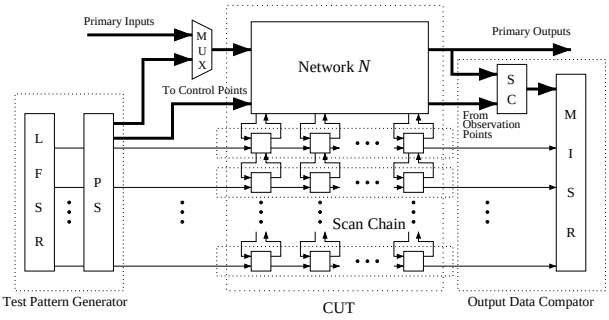


Figure 1: PSBIST architecture

by the MISR *every clock cycle* which is similar to the *test-per-clock* scheme. However, the responses are captured into the scan chains only after a new test pattern is scanned in, and then compressed by the MISR when they are scanned out. This is similar to the *test-per-scan* BIST. By doing so, the desired fault coverage can be reached earlier than the conventional test-per-scan approach without extra hardware.

3 Motivation

Several techniques have been developed to improve the test quality of pseudo random testing. Among them, weighted random testing uses multiple weight sets to provide different signal probability profiles. it can achieve the desired fault coverage with a reasonable test length [8][9][10][11]. However, if the number of weight set is large, the storage and extra hardware required for the pattern generator becomes costly. Instead of providing different signal probability profiles purely from the test pattern generator, we found that under the PSBIST architecture, test patterns with different signal probability profiles can be generated by changing the test application scheme. Specifically, using *multiple capture cycles after each scan sequence* results in different profiles for patterns in the scan flip-flops.

Before further discussion, we define the following terms.

- A *scan cycle* is the period in which a test pattern is shifted into (or the response is shifted out of) the scan chains. If the length of the longest scan chain is l, then one scan cycle corresponds to l clock cycles.
- A *functional cycle* is the period between two scan cycles. If we use k capture cycles per scan, then one functional cycle corresponds to k clock cycles.
- A *test cycle* is one scan cycle followed by one functional cycle.

A graphical representation of the entire test procedure and one test cycle is shown in Fig.2. Let's now consider the example in Fig.3: A, B and C are the primary inputs and their signal probabilities are 0.5 assuming they are driven by an LFSR. *PSI* is the pseudo-input driven by the scan flip-flop FF. During the scan cycle, the values appeared at *PSI* are random, therefore, its signal probability is 0.5. If we only apply

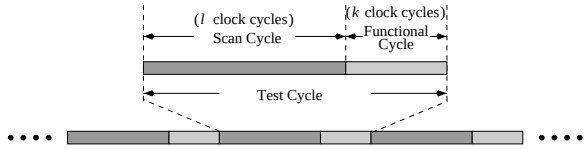


Figure 2: Entire test procedure and one test cycle

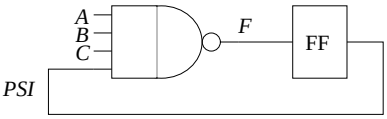


Figure 3: Different signal probability profiles at PSI

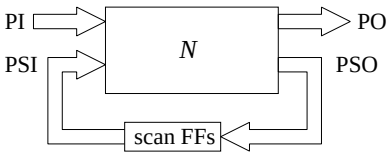
one capture cycle after each scan sequence, the signal probability of *PSI* remains 0.5 in the functional cycle because its value is derived from the scan chain. As a result, only *one* signal probability profile is produced using *one capture cycle per scan*. Notice that at the end of the scan cycle, the signal probability of F is 0.9375. If we use *two capture cycles* after the scan cycle, the value of F will be latched into FF after the first capture cycle and thus the signal probability of *PSI* changes to 0.9375 at the second capture cycle. Consequently, *two* profiles are generated at *PSI* during the functional cycle.

In this example, the effects of using *two capture cycles per scan* are the following. At the second capture cycle:

- Faults $A_{s/1}$, $B_{s/1}$ and $C_{s/1}$ become more observable (side input *PSI* has a greater chance of being a non-controlling value).
- Fault $F_{s/1}$ becomes easier to be activated (signal probability decreases from 09375 to 0.8828125).
- Fault $PSI_{s/1}$ becomes harder to be activated (signal probability increases from 0.5 to 0.9375).

This result suggests that the optimal number (k) of capture cycles vary from one fault to another. To achieve the best test result, a set of k's needs to be applied during the entire test. Thus, it is intuitive to divide the entire test process into *multiple test sessions with a different k in each session.* Using multiple test sessions has shown to be effective for test point insertion [12]. In [12], different subsets of control points are activated in different test sessions and each subset of control points only targets on a subset of faults. Here, instead of adding test points to increase the random testability of the CUT, we explore different test application schemes to be used in different test sessions. Each test session also only targets on a subset of faults.

In addition, it has been observed that under the PSBIST architecture, the test quality can be improved by increasing the observability of the scan flip-flops' data inputs [13]. Using *multiple capture cycles per scan* increases the chance of latching the fault effects thus increases the possibility of observing them at the primary outputs through functional logic *during the functional cycle.*



(a) A scanned circuit

Phase	Cycle	cntl(PI)	cntl(PSI)	obs(PO)	obs(PSO)
1	1st~ lth	0.5	0.5	1.0	0.0
2	(l+1)th~ (l+k -1)th	0.5	cntl(PSO) in previous clock cycle	1.0	obs(PSI) in next clock cycle
3	(l+k)th	0.5	cntl(PSO) in previous clock cycle	1.0	1.0

(b) Computing testability for one test cycle

Figure 4: Testability computation model

Implementing multiple capture cycles per scan only requires a minor modification to the BIST controller. In PS-BIST, we adjust the mode switch signal of scan flip-flops to put them in the functional mode for multiple clock cycles. The area overhead incurred by this modification is negligible comparing to the original single capture cycle per scan scheme.

4 Multiple test sessions with multiple capture cycles

In this section, we first introduce a testability computation model to find the detection probability of a fault using multiple capture cycles per scan. Based on this information, a procedure is proposed to find the required number of test sessions and the corresponding number of capture cycles per scan in each session for achieving the highest possible test quality.

4.1 Testability computation

Under stuck-at fault model, the detection probability Pd_i of a fault i can be estimated by one of following two equations:

$$Pd_{s/0} = C_s \cdot O_s \text{ , for s-a-0 at signal } s; \quad (1)$$

$$Pd_{s/1} = (1 - C_s) \cdot O_s \text{ , for s-a-1 at signal } s. \quad (2)$$

where C_s and O_s are 1-controllability and observability of signal s, respectively. They can be estimated very efficiently using COP [14]. Note that COP can only be applied to a combinational circuit or a sequential circuit with only self loops in its sequential graph without excessive iterations [15]. However, the CUT is sequential during the functional cycle, thus computing COP testability measures of the CUT in the functional cycle can be costly because the iterations may not converge quickly. Fortunately, the length of the functional cycle is fixed and relatively small in this application. By dividing one test cycle into multiple phases, we can calculate the testability measures using the original COP method with proper boundary conditions (controllabilities of the inputs and observabilities of the outputs).

Fig. 4(a) shows a scanned circuit where the primary inputs and outputs are denoted as PI and PO, respectively. Each scan flip-flop contributes a pseudo-input (PSI) and a pseudo-output (PSO). The entire test cycle is divided into 3 phases as shown in Fig. 4(b) assuming using k capture cycles per scan. Then C_s and O_s are computed with different boundary conditions in different phases. PIs are continuously driven random patterns, thus the controllability of a PI, *cntl*(PI), is 0.5 for all 3 phases. The observability of a PO, *obs*(PO), is always 1 because the values at POs are observed all the time. Unlike PIs and POs, the controllability of a PSI, *cntl*(PSI), and the observability of a PSO, *obs*(PSO), are different in different phases. The first phase is the scan cycle. In this phase, the PSI receives random patterns, thus *cntl*(PSI) is set to 0.5. Meanwhile, the responses at the PSOs are *not* captured, therefore *obs*(PSO) is 0. The second phase is the first k-1 capture cycles that the scan flip-flops are in the functional mode. In this phase, the patterns at PSIs are the responses captured at PSOs at the previous clock cycle. Consequently the *cntl*(PSI) is set to the corresponding *cntl*(PSO) at the previous clock cycle. Similarly, *obs*(PSO) at n-th cycle is equal to the corresponding *obs*(PSI) at (n+1)-th cycle. The last phase corresponds to the last capture cycle. The response captured by the scan flop-flops will eventually be scanned out and compressed by the MISR, therefore, *obs*(PSO) is 1. Once we know the proper *cntl*(PSI) and *obs*(PSO) in each phase, C_s and O_s for every signal s can be computed accordingly using the original COP method.

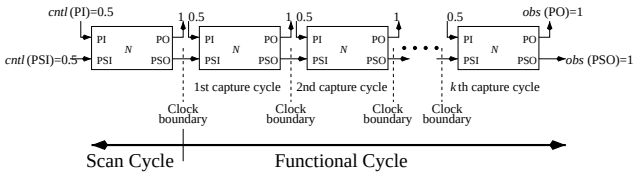


Figure 5: Circuit after time frame expansion

This testability computation model can also be illustrated using a time frame expansion model as shown in Fig. 5. For k capture cycles, the number of time frames required to be expanded is k+1 (1 for the scan cycle and k for the functional cycle). This expanded circuit is combinational thus original COP method can be applied directly. The controllabilities are computed from the inputs of the 1st copy of the circuit (the leftmost one in Fig. 5) toward the (k+1)-th copy of the circuit (the rightmost one in Fig. 5). Similarly, the observabilities are computed backward from the outputs of the (k+1)-th copy of the circuit toward the inputs of the 1st copy.

4.2 Finding the optimal number of capture cycles for each fault

After computing the COP testability measures, we can find the detection probability of the fault during the *functional*

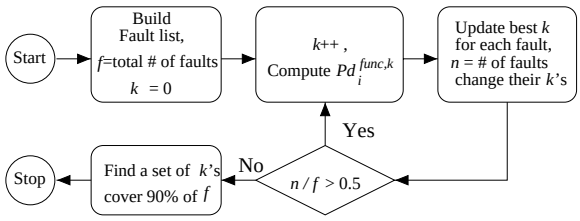


Figure 6: Determining the best test scheme

cycle using the following equation:

$$Pd_i^{func,k} = 1 - \prod_{j=1}^{k} (1 - Pd_i^j) \quad (3)$$

In Equation (3), $Pd_i^{func,k}$ is the detection probability of fault *i* in the *functional cycle* with *k capture cycles per scan*, and Pd_i^j is the detection probability of fault *i* at the *j*-th capture cycle. Pd_i^j is computed using Equation (1) or (2). By neglecting detecting faults in the scan cycle, $Pd_i^{func,k}$ approximates the detection probability of fault *i* in *one test cycle*. Given two different *k*'s, k_1 and k_2, we can easily compute Pd_i^{func,k_1} and Pd_i^{func,k_2} for each fault *i*. However, it would not be appropriate to compare them directly for deciding the optimal *k* value for fault *i*. This is because different *k*'s mean there are different numbers of clock cycles in one test cycle. Therefore, we normalize $Pd_i^{func,k}$ by dividing it by the number of clock cycles in one test cycle, i.e. $\frac{1}{l+k}$, where *l* is the length of the scan chain. For each fault *i*, we compare $\frac{Pd_i^{func,k}}{l+k}$ for different *k*'s and the best *k* for fault *i* is the one having the largest $\frac{Pd_i^{func,k}}{l+k}$.

4.3 Determining test scheme

With metric $\frac{Pd_i^{func,k}}{l+k}$, we propose a procedure to find the optimal number of test sessions and the corresponding number of capture cycles per scan in each session for a circuit as shown in Fig. 6. Note that the metric for determining the optimal *k* (i.e. $\frac{Pd_i^{func,k}}{l+k}$) for a fault is derived by neglecting the chance of detecting it in the scan cycle. Thus, only faults that are hard-to-detect during the scan cycle should be considered in this procedure. To identify these faults, we first compute COP measures of the circuit in scan cycle. Faults associated with those signals which have zero observabilities can never be detected in the scan cycle and thus must be targeted in the *functional cycle*. In addition, we also include faults with very low detection probabilities in the scan cycle, say $< 10^{-10}$, into the fault list. Finally, we record the total number of considered faults (f). After that, the procedure becomes an iterative process. At each iteration, *k* is first incremented (initially set to 0) and then the corresponding $Pd_i^{func,k}$ is computed. The new $Pd_i^{func,k}$ is normalized and compared with the previous recorded best normalized $Pd_i^{func,k}$ to determine if the best *k* value of fault *i* should be updated. After updating *k* for each fault, the number (n) of faults which change their best *k*'s is recorded. Because having too many test sessions may cause non-trivial area overhead on the BIST controller, thus the iteration continues only if a significant fraction of the considered faults change their best *k* values. In the work, we set the threshold to 50%, i.e. $n/f > 0.5$. Once the iterative process stops, the best *k* value of each fault is determined and we say a fault is covered by k_1 if its best *k* value is k_1. After that, a small set of *k*'s which covers a pre-determined percentage of considered faults is selected. This pre-determined percentage should cover most of the considered faults. In the work, we set it to 90%. Choosing a higher percentage may not be cost-effective because the increased area overhead for the extra test sessions is only devoted to a small number of faults (less than 10% in this case). Finally, the number of test sessions is equal to the number of *k*'s selected; the corresponding numbers of capture cycles are these *k*'s.

Circuit	# of Test sessions	# of capture cycles	CPU time (s.)
s953	4	2,3,4,5	15.0
s1423	2	1,2	15.3
s1512	3	1,2,3	15.5
s3271	3	2,3,4	18.7
s3330	4	1,4,5,6	19.3
s3384	3	2,3,4	19.0
s4863	3	1,4,5	21.6
s5378	2	1,2	19.5
s6669	3	2,4,5	25.4
s9234.1	3	2,3,4	35.6
s15850.1	3	1,2,3	51.9
s35932	3	1,2,3	276.9
s38417	3	1,2,3	211.1
s38584	3	1,2,3	271.0

Table 1: Test schemes

5 Experimental Results

We have implemented the algorithm and conducted experiments on ISCAS89 benchmark circuits. An industrial tool, **psb2** [1], is used to automatically insert the BIST circuitry. While adding the BIST capability, we set the length of the longest scan chain to 10 and use a 21-stage LFSR as the test pattern generator (19-stage LFSR for s953). The scanned circuits are fault simulated for 500K clock cycles using both single test session (STS) — one capture cycle per scan — and multiple test sessions (MTS) schemes. Note that fault simulation is done by issuing commands to the BIST controller so that the circuit tests itself. Therefore, the fault list contains faults not only in the CUT but also in the added BIST circuitry. To ensure a fair fault coverage comparison, we keep the fault lists identical for both STS and MTS schemes by excluding faults associated with the extra circuitry for MTS in the BIST controller.

We first determine the number of test sessions and the corresponding number of capture cycles per scan using our algorithm. The results are shown in Table 1. For example, circuit s3330 requires 4 test sessions with *1, 4, 5 and 6 capture cycles per scan* in each test session, respectively. Given

[1] **psb2** is a product of Lucent Technologies [5].

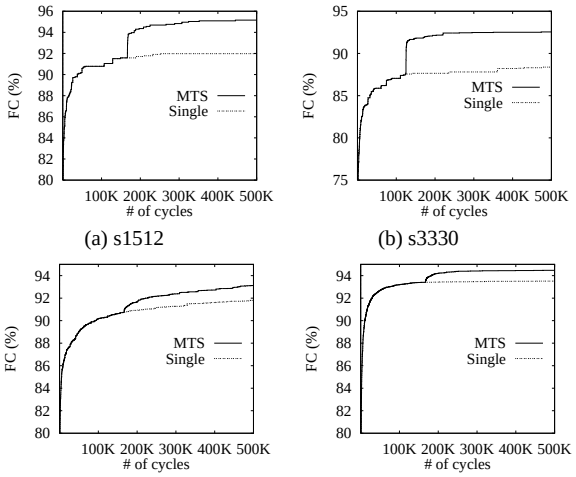
Circuit	FC (%)		Area					
			Original	Overhead (%)				
					Other		Overall	
	Single	Multiple	(grid count)	Scan	Single	Multiple	Single	Multiple
s953	96.63	99.92	1687	10.3	77.7	91.7	88.0	102.0
s1423	96.81	98.50	3339	13.3	42.1	45.7	55.4	59.0
s1512	91.98	95.16	3230	10.6	46.5	52.0	57.1	62.5
s3271	98.94	99.84	6656	10.5	22.8	25.4	33.2	35.9
s3330	88.38	92.55	7513	10.5	21.7	24.8	32.2	35.4
s3384	96.89	97.15	8389	13.1	20.5	22.6	33.6	35.7
s4863	97.16	97.42	9698	6.4	17.7	19.5	24.1	26.0
s5378	96.47	96.65	10094	10.6	16.2	17.4	26.9	28.0
s6669	99.72	99.89	14318	10.0	14.6	15.8	24.6	25.9
s9234.1	86.31	86.79	18330	6.9	9.0	9.9	15.9	16.8
s15850.1	87.19	88.23	34290	9.3	6.3	6.8	15.6	16.2
s35932	89.02	89.65	77166	13.4	2.4	2.6	15.8	16.0
s38417	91.82	93.12	84763	11.6	2.0	2.2	13.6	13.8
s38584	93.51	94.47	79022	10.4	1.9	2.1	12.4	12.6
Ave.	93.63	94.95	25918	11.0	6.5	7.1	17.5	18.1

Table 2: Fault simulation results and area overheads

a total number of clock cycles for BIST, we simply distribute them into four equal-test-length sessions. The run time to determine this test scheme is 19.3 seconds on a Sun SparcStation 20. The BIST controller is then modified accordingly to perform the desired operations — 1, 4, 5 and 6 capture cycles per scan.

The fault simulation results and area information are shown in Table 2. The 2nd and 3rd columns are the fault coverages using STS and MTS schemes, respectively. Both schemes run the same number of clock cycles (500K) for BIST. Although the fault coverage improvement varies, using MTS gives us higher fault coverages in all cases. We obtained an average 94.95% fault coverage with MTS scheme as opposed to 93.63% with STS scheme. The area information in Table 2 does not include routing area. The circuit size without BIST capability is in Column 4 in unit of grid using Lucent's 0.5μ CMOS standard cell library. The area overhead is divided into scan and other BIST circuitry overheads. Both are normalized by dividing them by the original circuit size. Scan overhead in Column 5 is the cost of converting flip-flops into scan flip-flops. This part is identical for both test schemes. Other BIST circuitry overhead includes BIST controller, TPG, ODC and multiplexers. Because STS and MTS differ only in the BIST controller, this part is slightly different for STS and MTS schemes and shown in Columns 6 and 7, respectively. The last 2 columns show the overall area overhead. The extra area overhead of MTS depends only on the number of test sessions and the maximum number of captures cycles per scan. For larger circuits, the overall area overhead of MTS is similar to that of STS.

The fault coverage curves of some circuits are shown in Fig. 7. Unlike the relatively smooth curves using STS, using MTS creates "jumps" at the beginning of a new test session. The "magnitude" of the "jump" is significant at the beginning of the 2nd test session and then diminishes afterwards. This diminishing phenomenon may be caused by the greedy nature of determining the k value for each fault. Given two k values, k_1 and k_2, we say k_1 is better than k_2 for fault i if $\frac{Pd_i^{func,k_1}}{l+k_1}$ is greater than $\frac{Pd_i^{func,k_2}}{l+k_2}$. However, if



(a) s1512 (b) s3330

(c) s38417 (d) s38584

Figure 7: Fault coverage curves

Circuit	Single		Multiple		Test length reduction[2] (%)
	FC (%)	Cycle	FC (%)	Cycle	
s953	96.63	214025	96.63	20262	90.5
s1423	96.81	127712	96.81	127712	0
s1512	91.98	252737	91.98	166700	34.0
s3271	98.94	381512	98.94	13762	96.4
s3330	88.38	476637	88.82	125112	73.8
s3384	96.89	498350	96.89	31350	93.9
s4863	97.16	429337	97.24	359287	16.3
s5378	96.47	326137	96.47	323400	0.8
s6669	99.72	79537	99.72	1625	98.0
s9234.1	86.31	496900	86.41	365075	26.5
s15850.1	87.19	440437	87.19	254337	42.3
s35932	89.02	2850	89.02	2850	0
s38417	91.82	498375	91.82	204400	59.0
s38584	93.51	455162	93.51	167187	63.3
Ave.	93.63	412836	93.68	154504	62.6

Table 3: Test length comparison

$|\frac{Pd_i^{func,k_1}}{l+k_1} - \frac{Pd_i^{func,k_2}}{l+k_2}|$ is relatively small, both k_1 and k_2 may work equally well for fault i. In this case, if both k_1 and k_2 are included in the final test application scheme and k_1 is applied before k_2, then faults which are intended to be targeted in the session with k_2 capture cycles per scan may also likely be detected in the session with k_1 capture cycles per scan.

Table 3 shows comparison of test length using STS and MTS. Here we set the target fault coverage to be the final fault coverage of STS. We then record the earliest clock cycle at which the target fault coverage is reached for both STS and MTS. Columns 2 and 3 show the final fault coverages and corresponding number of clock cycles for STS; Columns 4 and 5 are the results of MTS. With comparable fault coverages, the test length reduction (in %) using MTS is calculated and shown in the last column. The reductions are very significant for many circuits. The average test length reduction is approximately 62.57%. Note that we do not obtain any improvement for s1423 and s35932. This is because the first test session for both circuits uses one capture cycle per scan which is the same as STS. Moreover, the target

[2] [(earliest cycle for single)–(earliest cycle for multiple)]/(earliest cycle for single).

Circuit	Single		Multiple	
	# of TPs	FC (%)	# of TPs	FC (%)
s953	1	98.61	0	99.92
s1423	2	98.15	0	98.50
s1512	6	98.16	4	99.21
s3330	30	98.28	5	98.84
s3384	1	98.97	1	99.17
s4863	5	98.03	3	98.75
s5378	25	96.78	15	96.81
s9234.1	30	94.44	25	94.47
s15850.1	30	95.63	25	95.96
s35932	10	99.44	5	99.94
s38417	30	97.11	15	97.70
s38584	30	96.66	15	96.87

Table 4: Results after adding test points

fault coverages (96.81% for s1423 and 89.02% for s35932) are reached in the first test session. Therefore, even though the final fault coverages (after 500K clock cycles) are higher with MTS, they are not reached until the later test sessions. This indicates the timing to switch test sessions greatly affects the test length.

Although using the proposed MTS scheme increases the fault coverage, it may still require other DFT techniques to obtain a desired level of fault coverage. Therefore, we combine the proposed approach with a test point selection algorithm [16] to study the effects of using MTS on the required number of test points. Test points are added to circuits which have fault coverages lower than 98% with STS. Note that the test points are selected by assuming *obs*(PSO) equals to 1. This represents the case of applying infinite number of test cycles. Thus, the selected test points should improve the fault detection in every test cycle regardless the number of capture cycles used in each test cycle. We can expect that the proposed test application scheme can reduce the number of test points to achieve a given fault coverage. The results in Table 4 validate this point. MTS always obtains higher fault coverages than STS does with *a fewer* number of test points. The difference in the number of test points becomes much more significant for larger circuits. This is because when the circuit reaches a high fault coverage level, the undetected faults tends to be scattered over the entire circuit. Thus, it usually requires a separate test point to target each one of them. If we can detect extra faults with the proposed test application scheme, then the number of required test points can be drastically reduced. Furthermore, it reduces the chance of causing performance degradation due to the test points.

6 Conclusion

A general test application scheme is proposed to improve the test quality of the scan-based BIST. Instead of capturing the responses into the scan chain once every scan cycle, capturing the responses multiple times can increase the chance of detecting a subset of faults. We introduce a testability computation model for finding the detection probability of a fault in the functional cycle. A metric is developed to find the optimal number of capture cycles per scan for each fault. We propose the use of multiple test sessions with multiple capture cycles per scan for better test quality. A procedure is used to determine the best test scheme for a given circuit. Experimental results show significantly improvement on both fault coverage and test length. The experimental results also indicates that the proposed test application scheme and test point insertion are complementary — A higher fault coverage is achieved with fewer test points. The difference in the number of test points is more significant for larger circuits. The timing of switching test sessions greatly influences the test length and this issue is currently under investigation.

References

[1] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications," *IEEE Design & Test of Computers*, vol. 10, no. 22, pp. 69–77, June 1993.

[2] B. Konemann, J. Mucha, and C. Zwiehoff, "Built-In Logic Block Observation Technique," *Digest of Papers 1979 Test Conf.*, pp. 37–41, Oct. 1979.

[3] A. Krasniewski and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits," *IEEE Trans. on CAD*, vol. 8, no. 1, pp. 46–55, Jan. 1989.

[4] P.H. Bardell and W.H. McAnney, "Selt-Testing of Multichip Logic Modules," *Digest of Papers 1982 Int'l Test Conf.*, pp. 200–204, Nov. 1982.

[5] C.-J. Lin, Y. Zorian, and S. Bhawmik, "Integration of Partial Scan and Built-In Self-Test," *JETTA*, vol. 7, no. 1–2, pp. 125–137, Aug. 1995.

[6] P.H. Bardell, "Design Considerations for Parallel Pseudo-Random Pattern Generators," *JETTA*, vol. 1, pp. 73–87, Feb. 1990.

[7] Y. Zorian and A. Ivanov, "Programmable Space Compaction for BIST," *Proc. of Int'l Symp. on Fault-Tolerant Computing*, pp. 340–349, June 1993.

[8] J.A. Waicukauski and E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns," *Proc. of ITC*, pp. 245–255, 1988.

[9] I. Pomeranz and S.M. Reddy, "3-weight Pasudo-random Test Generation Based on A Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Trans. on CAD*, vol. 24, pp. 1050–1058, July 1993.

[10] M. Bershteyn, "Calculation of Multiple Sets of Weighted Random Testing," *Proc. of ITC*, pp. 1031–1040, Oct. 1993.

[11] R. Kapur, S. Patil, T.J. Snethen, and T.W. Williams, "A Weighted Random Pattern Generation System," *IEEE Trans. on CAD*, vol. 15, no. 8, pp. 1020–1025, Aug. 1996.

[12] N. Tamarapalli and J. Rajski, "Constructive Multi-Phases Test Point Insertion for Scan-Based BIST," *Proc. of ITC*, pp. 649–658, Oct. 1996.

[13] H.-C. Tsai, S. Bhawmik, and K.-T. Cheng, "An Almost Full-scan BIST Solution — Higher Fault Coverage and Shorter Test Application Time," *Proc. of ITC*, pp. 1065–1073, Oct. 1998.

[14] F. Brglez, "On Testability of Combinational Networks," *Proc. of ISCAS*, pp. 221–225, May 1984.

[15] K.-T. Cheng and C.-J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," *Proc. of ITC*, pp. 506–514, Oct. 1995.

[16] H.-C. Tsai, K.-T. Cheng, C.-J. Lin, and S. Bhawmik, "A Hybrid Algorithm for Test Point Selection for Scan-Based BIST," *Proc. of DAC*, pp. 478–483, June 1997.