A Hybrid Algorithm For Test Point Selection For Scan-based Bist

Conference Paper · July 1997

DOI: 10.1109/DAC.1997.597195 · Source: IEEE Xplore

K.-T. Tim Cheng
University of California, Santa Barbara
477 PUBLICATIONS **12,986** CITATIONS
SEE PROFILE

Chih-Jen Lin
Samsung Electronic
15 PUBLICATIONS **480** CITATIONS
SEE PROFILE

Sudipta Bhawmik
Qualcomm
35 PUBLICATIONS **669** CITATIONS
SEE PROFILE

A Hybrid Algorithm for Test Point Selection for Scan-Based BIST

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

Chih-Jen Lin*
Intel Corporation
5200 N.E. Elam Young Pkwy.
Hillsboro, OR 97124

Sudipta Bhawmik
Bell Laboratories
Lucent Technologies
Princeton, NJ 08542

Abstract

We propose a new algorithm for test point selection for scan-based BIST. The new algorithm combines the advantages of both explicit-testability-calculation and gradient techniques. The test point selection is guided by a cost function which is partially based on explicit testability recalculation and partially on gradients. With an event-driven mechanism, it can quickly identify a set of nodes whose testability need to be recalculated due to a test point, and then use gradients to estimate the impact of the rest of the circuit. In addition, by incorporating timing information into the cost function, timing penalty caused by test points can be easily avoided. We present the results to illustrate that high fault coverages for both area- and timing-driven test point insertions can be obtained with a small number of test points. The results also indicate a significant reduction of computational complexity while the qualities are similar to the explicitly-testability-calculation method.

1 Introduction

Because of the relatively low hardware overhead and the simplicity of test pattern generation, pseudo-random testing is a favorable technique for BIST. Many practical circuits, however, contain random pattern resistant faults which cause unacceptable low fault coverages for a reasonable test length. One way to resolve this problem is to modify the circuit under test by Test Point Insertion (TPI). Several studies analyze the circuit structure to select the test points. First, fault simulation is performed and then circuit structure is examined to identify the reconvergences[2] and gates which block the activations and propagations of faults[4][11]. Those are possibly good test point candidates.

Because fault simulation is costly in terms of computational complexity, it may not be applicable to large designs. An alternative is to use testability measures. For instance, COP[1] testability measures are used to select test points[7]. In [8], a set of equations which make use of the gradients[6] is developed to select a set of candidates. A test point is then selected by explicitly recalculating the COP measures and the cost function for each candidate. This method is further generalized for partial-scan circuits with a novel testability calculation[3][5]. A performance-driven test point selection technique is also proposed in [3]. Here, we refer to the methods used in [3] and [8] as the gradient-based approaches. Recently, a divide-and-conquer approach is used to partition entire TPI process into multiple phases[10]. In each phase, a set of test points are inserted. Probabilistic fault simulation is used to guide the placement of the test point. Control points are enabled in different phases with a fixed value and shared a common driving source to reduce the area overhead.

This paper presents a new test point selection algorithm. It is highly efficient and the quality of selection is similar to that of the explicit-testability-calculation method as justified by the experimental results. The key idea is that we use the event-driven mechanism to propagate the changes of testabilities due to a test point. Once the changes become small, the gradients are used to estimate the testability changes of all other nodes to avoid high computational complexity of recalculating testability for the entire circuit. Additional timing information can be taken into account to avoid adding test points on the timing critical paths. Experiments have been conducted to show the effectiveness of our algorithm. The results indicate that the new hybrid algorithm is much faster and achieves a higher fault coverage than previous approaches with the same number of test points. For timing-driven TPI, zero performance degradation is always achieved. Comparison with other approaches was made and summarized in Sections 3.

The rest of this paper is organized as following: in Section 2, some basic testability analysis techniques and the problems of the gradient-based method are discussed. The concepts of the new hybrid method and computation of a new cost function called *Hybrid Cost Reduction* are presented in Section 3. Two heuristics are also proposed in Section 3 to further improve the performance. Experimental results are presented throughout Sections 3 to demonstrate the effectiveness of the new hybrid method and the trade-offs of various heuristics.

*This work was conducted when the author was with Lucent Technologies.

0-89791-847-9/97/0006/$3.50 DAC 97 - 06/97 Anaheim, CA, USA

Section 4 concludes this paper.

2 Background

Controllability/Observability Program (COP) is a well-known procedure to estimate the 1-controllability C_s and observability O_s of every signal s in a combinational network. Both C_s and O_s can be computed efficiently by sweeping the circuits once. Because the controllabilities and observabilities represent only the local testability impact, they cannot reflect the global effect of adding a test point. Therefore, a good cost function proposed in [6] is used to estimate the global circuit testability. This cost function is defined as

$$U = \frac{1}{|F|}\left(\sum_{i \in F} \frac{1}{Pd_i}\right) \tag{1}$$

Where F is the fault set and $|F|$ is the cardinality of F, and Pd_i is the detection probability of fault i. For the stuck-at fault model, Pd_i can be expressed as one of following two equations:

$$Pd_{s/0} = C_s \cdot O_s, \text{for stuck-at-0 fault at } s. \tag{2}$$
$$Pd_{s/1} = (1 - C_s) \cdot O_s, \text{for stuck-at-1 fault at } s. \tag{3}$$

$1/Pd_i$ can be viewed as the expected number of pseudo-random patterns needed to be applied to detect fault i. Therefore, we can use U as an indicator of circuit testability and the objective of TPI is to minimize it. The value of U changes once a test point is added, and the difference of the values of U before and after test point insertion is called actual cost reduction (ACR). Using a greedy approach, we can select a test point which results in the largest ACR. A trivial and exhaustive method is to explicitly compute ACR for each possible test point in the circuit, and then select the one with the largest ACR as the test point. The computational complexity of this approach is too high to be practical for large designs. Notice that ACRs computed by this method are not perfect indicators for selecting test points either due to following two reasons: (1)COP provides only an estimate of true controllability and observability because of the assumption of signal independence. (2)Pd_i is also an estimate of true detection probability because it assumes the controllability and observability are independent. However, our extensive experimental results show that ACRs computed by this method are accurate enough for large designs (they are our main targets). So we adopt them as reference points for verifying the quality of our new approach.

In [6], a linear time algorithm to calculate the gradients, G_{C_s} and G_{O_s}, of U with respect to an infinitely small change of node testability (i.e. $G_{C_s} = \frac{dU}{dC_s}$, $G_{O_s} = \frac{dU}{dO_s}$) was developed. G_{C_s} and G_{O_s} are called controllability gradient and observability gradient, respectively. Similar to C_s and O_s, the gradients are not suitable for test point selection either, because inserting a test point will drastically change the testability of a node that violates the assumption of the gradients. Therefore, *Cost Reduction Factor* (CRF)[8] was introduced to approximate ACR for a test point candidate. The CRFs, however, could be significantly deviated from the ACRs, especially when the circuit has a high fault coverage, for the following several reasons:

First, the observability changes due to a control point is completely ignored. It means that the observability changes of nodes in the fanin cone of primary outputs reachable from the control point are completely neglected.

Second, certain circuit structures and fault sets were assumed during the derivation of the CRF equations. A chain of AND gates or OR gates usually cause extremely low detection probabilities. By assuming that circuits under test contain these types of structures, a controllability value C_s can be factored out such that the rest of the terms in U are completely independent to C_s. This assumption is not generally true and could contribute errors to CRFs.

The inaccuracy is more significant when a high fault coverage is reached. Therefore, in [3][8], an additional step is added: explicit calculation of ACRs is performed for a set of test point candidates with high CRFs. The one with the highest ACR is then selected as a test point. Since the calculation of ACRs dominates the CPU usage of the gradient-based method, only a small number of candidates are allowed to have their ACRs computed to reduce the complexity. Therefore, some good candidates may be excluded from the second step. These three factors limit the effectiveness of the gradient-based method.

A good solution to TPI is to find an accurate approximation of ACR such that the second step of the gradient-based method can be eliminated. In the mean time, this estimate must be computed very efficiently. This bring us to the hybrid test point insertion algorithm.

3 Hybrid Test Point Insertion Algorithm

In this section, we first introduce a new cost function called *Hybrid Cost Reduction* (HCR) which is used to estimate the ACR of a test point. We then present the new algorithm and results. Two heuristics are also proposed to further improve the performance.

3.1 Computing Hybrid Cost Reduction

Given a fault set F, the ACR for a test point s can be expressed as

$$\begin{aligned} ACR^s &= \Delta U^s = U^s - U^{org} \\ &= \frac{1}{|F|}\left[\sum_{i \in F}\left(\frac{1}{Pd_i{}^s} - \frac{1}{Pd_i{}^{org}}\right)\right] \\ &= \frac{1}{|F|}\left[\underbrace{\sum_{i \in F_1}\left(\frac{1}{Pd_i{}^s} - \frac{1}{Pd_i{}^{org}}\right)}_{\text{the difference is large}} + \underbrace{\sum_{i \in F_2}\left(\frac{1}{Pd_i{}^s} - \frac{1}{Pd_i{}^{org}}\right)}_{\text{the difference is small}}\right] \end{aligned} \tag{4}$$

Where U^{org}, U^s are the U values (defined in Equation (1)), and $Pd_i{}^{org}$, $Pd_i{}^s$ are the detection probabilities of fault i before and after inserting s, respectively.

According to the effect on U caused by a test point for each fault, we further divide the fault set F into two subsets F_1 and F_2. For each fault in F_1, $\frac{1}{Pd_i{}^s} - \frac{1}{Pd_i{}^{org}}$ is large and therefore the gradient approximation is not accurate. On the other hand, for every fault in F_2, the difference of $\frac{1}{Pd_i{}^s}$ and $\frac{1}{Pd_i{}^{org}}$ is small enough such that the gradients can accurately estimate their impact on U. Typically, the size of F_1 is much smaller than that of F_2. Therefore, we can explicitly calculate $\frac{1}{Pd_i{}^s} - \frac{1}{Pd_i{}^{org}}$ for faults in F_1 (by computing $\frac{1}{Pd_i{}^s}$) and use gradients to evaluate the second term of Equation (4). The event-driven mechanism is used to propagate the controllability/observability changes caused by the test point candidate. When the changes drop below a threshold, the changes are not further propagated. Whether an event is scheduled for further propagation is determined by the ratio of $G_{C_i} \cdot \Delta C_i$ to U^{org}, not the value of $G_{C_i} \cdot \Delta C_i$ nor the value of ΔC_i itself. The region of nodes processed to identify set F_1 is very small for large design, therefore, the complexity for calculating the first term of Equation (4) is low. In addition, because the assumptions of the gradients are not violated for faults in F_2, gradients can provide a very good estimate for the second term of Equation (4). In the following, we show how HCRs are computed for an observation point and a control point, respectively.

Observation Point The propagation of the changes of the observabilities starts from the observation point a toward the primary inputs. The effect is shown in Figure 1. For each node i inside region I, the ratio of $G_{O_i} \cdot \Delta O_i$ to U^{org} is larger than a given threshold so that the new observabilities of its fanin nodes are further evaluated. Propagation stops at the boundary nodes (between region I and region II) whose $G_{O_i} \cdot \Delta O_i$ to U^{org} ratios are below the threshold. The HCR for an observation point comprises three parts:

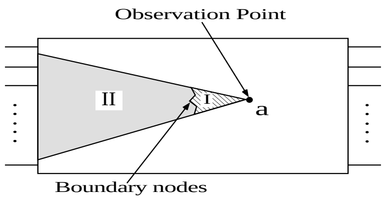

Figure 1: Computing HCR for an observation point

(i) $\sum_i \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}}\right)$ for every fault i inside region I. We explicitly compute the new O_i's (by the above procedure) and therefore the new Pd_i's.

(ii) For faults inside region II, $\sum_j G_{O_j} \cdot \Delta O_j$, where j's are the nodes *on the boundary* in Figure 1, is used to estimate their contribution to the ACR.

(iii) If the observation point a is not a fanout stem in the the original circuit, the contribution of the two new fanout branch faults ($\frac{1}{Pd_{a/1}}$ and $\frac{1}{Pd_{a/0}}$) are added.

The equation of computing HCR for an observation point is summarized in Equation (5). Notice that the estimate of contributions from nodes in region II can be viewed as adding a set of pseudo-observation points (with a delta change in observability) on the boundary simultaneously. The second term of Equation (5) simply represents the superposition of the effects of all these pseudo-observation points. Because the change is small, the assumption of the gradients holds and therefore the accuracy of this term would be high.

Control Point Computing HCR for a control point is more complicated The propagation of the new COP values has to proceed in both forward and backward directions. First, starting from the control point, the propagation of the new controllability proceeds forward toward the primary outputs shown as region I in Figure 3. At the end of processing each gate, the ratio of $G_{C_i} \cdot \Delta C_i$ to U^{org} is compared with a given threshold to decide if the controllabilities of its fanouts should be calculated.

Once the forward propagation stops, we obtain a set of nodes indicated as boundary I in Figure 3. The direction of propagation is then changed to backward toward the primary inputs. We use nodes at boundary I as the starting points and proceed the backward propagation similar to what is done in the case of adding observation points.

Again, while the ratio of $G_{O_i} \cdot \Delta O_i$ to U^{org} is small, the backward propagation stops, and a set of nodes indicated as boundary II in Figure 3 is identified. The HCR of a control point contains four terms:

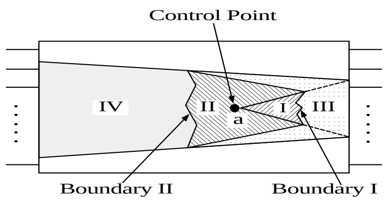

Figure 3: Computing HCR for a control point

(i) For every fault i inside regions I and II, we compute $\sum_i \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}}\right)$ explicitly. New C_i's, O_i's and therefore Pd_i's are computed by the above mentioned event-driven procedures.

(ii) $\sum_{k \in \text{boundary I}} (G_{C_k} \cdot \Delta C_k)$ is used to estimate $\sum_{i \in \text{region III}} \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}}\right)$.

(iii) $\sum_{j \in \text{boundary II}} (G_{O_j} \cdot \Delta O_j)$ is used to estimate $\sum_{i \in \text{region IV}} \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}}\right)$.

(iv) We need to add the contribution of the new faults introduced by the inserted gate: $\frac{1}{Pd_{a/0}}$ and $\frac{1}{Pd_{t/0}}$. These are s-a-0(s-a-1) faults at the extra input and the output of the added OR(AND) gate.

$$HCR_a{}^{OBS} = \sum_{i \in \text{region I}} \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}} \right) + \sum_{j \in \text{boundary}} \left(G_{O_j} \cdot \Delta O_j \right) + \left(\frac{1}{Pd_{a/1}} + \frac{1}{Pd_{a/0}} \right) \quad (5)$$

$$HCR_a{}^{OR} = \sum_{i \in \text{regions I \& II}} \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}} \right) + \sum_{k \in \text{boundary I}} \left(G_{C_k} \cdot \Delta C_k \right) + \sum_{j \in \text{boundary II}} \left(G_{O_j} \cdot \Delta O_j \right) + \left(\frac{1}{Pd_{a/0}} + \frac{1}{Pd_{t/0}} \right) \quad (6)$$

Figure 2: Equations for computing HCRs

The equation to compute HCR for an OR type control points is summarized in equations (6). By viewing every node j on boundary II as a pseudo-observation point, we use the superposition, $\sum_j \left(G_{O_j} \cdot \Delta O_j \right)$, to estimate $\sum_i \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}} \right)$ for faults i's inside region IV. Similarly, by viewing every node k on boundary I as a pseudo-control points, We use $\sum_k \left(G_{C_k} \cdot \Delta C_k \right)$ to estimate the contribution for faults inside region III. A control point, however, affects the testabilities of both its fanins and fanouts. Inserting a set of control points on boundary I should have effects on all faults in regions I$\sim$IV. This means $\sum_{k \in \text{boundary I}} \left(G_{C_k} \cdot \Delta C_k \right)$ also contains some contributions from region I, II, and IV. Because the contribution from region III usually dominates $G_{C_k} \cdot \Delta C_k$, the error by this approximation is negligible. Furthermore, unlike observation points which always reduce the value of U if they are inserted simultaneously, the effects of control points can cancel one another if they are added at the same time. For these reasons, using $\sum_{k \in \text{boundary I}} \left(G_{C_k} \cdot \Delta C_k \right)$ to estimate $\sum_i \left(\frac{1}{Pd_i{}^a} - \frac{1}{Pd_i{}^{org}} \right)$ for each fault i in region III is just a heuristic.

3.2 Evaluation of Accuracy of HCRs

In order to access the quality of the test points selected by the algorithm, we conducted experiments on two ISCAS85 benchmark circuits — C2670 and C7552 which are random-resistant. Twenty test points were selected sequentially for each circuit. At each iteration, we explicitly compute the ACRs for all possible test point candidates and sort them in a descending order. We then check the rank of the test point selected by the hybrid algorithm. The rankings of the test points selected by the hybrid algorithm are shown in Table 1. The results show that the test points selected by the new algorithm are almost identical with the ones with the highest ACRs (a rank of 1 means that the test point has the highest ACR among all possible test point candidates).

For comparison, we also use the gradient-based method to select the test points and check the ranks (based on ACRs). The results are shown in Table 2. The test points selected are sometime quite off, especially for the case that control points are selected.

3.3 Algorithm

In Algorithm 1, we summarize the hybrid test point insertion algorithm. The *slack*, defined as the difference of the required arrival time and actual arrival time of a signal, is computed by a timing analyzer. The first four steps in the while loop can be computed in linear time. The computation of HCRs is also highly efficient as explained earlier. Because HCRs are very accurate, we directly select the one with the highest HCR as the next test point. The algorithm shown avoids inserting test points to the nets on the timing critical paths. We call this version of the algorithm as the timing-driven TPI. The area-driven TPI does not use the timing information. Therefore, the slacks need not to be computed and the two if-conditions are removed.

Iteration	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
C2670	1	1	1	1	1	1	1	1	1	1
C7552	1	1	1	1	1	1	1	1	1	7
Iteration	11th	12th	13th	14th	15th	16th	17th	18th	19th	20th
C2670	1	1	1	1	1	1	1	1	2	1
C7552	1	1	1	2	2	1	3	2	1	1

Table 1: Rankings of test points selected by hybrid method

Iteration	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
C2670	106	8	4	6	1	3	1	1	1	1881
C7552	1	1	11	1	1	1	1	1	1	29
Iteration	11th	12th	13th	14th	15th	16th	17th	18th	19th	20th
C2670	1881	1885	1895	11	1901	1901	1911	1	1925	1925
C7552	1	1	3	2	1	1	1	2	1	1

Table 2: Rankings of test points selected by gradient method

Algorithm 1 Hybrid TPI algorithm

```
while ($FC$<desired $FC$) $\wedge$ (# of tp<$Max\_tp$) do
  compute slacks for all nodes in the circuit;
  compute the controllability $C_s$ for every node $s$;
  compute the observability $O_s$ for every node $s$;
  compute the gradients $G_{C_s}$ and $G_{O_s}$ for every node $s$;
  for each node $s$ in the circuit do
    if $Slack_s > Slack_{OBS\_threshold}$ then
      compute $HCR_s{}^{OBS}$;
    end if
    if $Slack_s > Slack_{CNTL\_threshold}$ then
      compute $HCR_s{}^{OR}$;
      compute $HCR_s{}^{AND}$;
    end if
  end for
  Select and insert the test point with the highest$HCR$;
end while
```

Experimental Results We have tried the algorithm on several large ISCAS85 and ISCAS89 benchmark circuits which are random-resistant. The circuits were first optimized for performance using Berkeley SIS

Circuit name	Original circuit		Area-driven TPI			Timing-driven TPI		
	FC[1](%)	Delay (ns)	#CP/#OP	FC (%)	Delay (ns)	#CP/#OP	FC (%)	Delay (ns)
C2670	83.23	40.6	4/1	98.31	40.6	4/1	98.31	40.6
C7552	95.12	62.7	3/7	98.23	62.7	3/7	98.23	62.7
s3330	86.54	13.4	7/6	99.92	14.3	16/9	99.95	13.4
s3384	96.82	20.4	0/9	99.78	20.8	5/4	99.78	20.4
s4863	99.22	27.5	1/1	99.64	27.5	1/1	99.64	27.5
s9234	87.99	15.1	16/3	96.10	16.1	20/2	96.13	15.1
s15850	90.56	24.5	26/8	97.41	29.1	23/25	97.14	24.5
s38417	87.37	18.2	30/16	99.19	18.3	36/12	99.18	18.2

Table 3: Results of area- and timing-driven test point insertion using the hybrid algorithm

synthesis system[9] and mapped into Lucent 0.9 micron CMOS cell library. Table 3 shows the results of area- and timing-driven TPI. In this experiment, we set $Slack_{OBS\_threshold}$ and $Slack_{CNTL\_threshold}$ to 1 ns and 3 ns, respectively and also set the threshold of scheduling an event to 0.1% for HCR^{OBS}, HCR^{OR} and HCR^{AND}. The results of area-driven TPI are shown in columns 4~6, and those of timing-driven TPI are shown in columns 7~9. The fault coverages shown in the table are computed after applying 32000 random patterns. As expected, some circuits suffer from performance degradation for area-driven TPI. The degradation could be as much as 19% (s15850) which is obviously unacceptable. For timing-driven TPI, there is no timing penalty for all circuits. The results also show that with a few more test points, the timing-driven TPI can achieve the same level of fault coverage as the area-driven TPI does.

Furthermore, we compare the results with those of the gradient-based method as shown in Table 4. Same number of test points are selected using both approaches. Fault coverages and CPU time are listed and the ratio of the CPU time of the hybrid method to that of the gradient-based method is shown in the last column. The CPU time is measured on a SUN SPARCstation 5. The results indicate that the hybrid algorithm achieves a higher fault coverage and uses much less CPU time than that of the gradient-based algorithm, especially for large circuits. The large reduction of the CPU usage mainly comes from the elimination of the second step of the gradient-based algorithm.

Circuit name	# of TP	Hybrid		Gradient-based		CPU time ratio(%)
		FC (%)	CPU time	FC (%)	CPU time	
C2670	5	98.31	78 sec.	98.31	264 sec.	30
C7552	10	98.23	3.4 min.	97.92	22 min.	15
s3330	13	99.92	4.9 min.	99.39	29 min.	17
s3384	9	99.78	128 sec.	99.78	659 sec.	19
s4863	2	99.64	61 sec.	99.61	201 sec.	30
s9234	19	96.10	12.3 min.	94.79	75 min.	16
s15850	34	97.41	34.4 min.	97.08	7.7 hr.	7
s38417	46	99.19	1.7 hr.	98.84	92.1 hr.	2
Ave.	-	98.57	-	98.22	-	17

Table 4: Comparison with the results of the gradient-based algorithm

3.4 Heuristics

Fault-list driven TPI Note that the fault set F used in Equation (1) can be any set of faults that we are interested in. When the fault coverage reaches high 90's, the value of U based on all faults may be dominated by the contributions of easy-to-detect faults. In such case, it would be beneficial to define U with respect to the set of hard-to-detect faults. The cost is the need of fault simulation to classify faults to be either easy-to-detect or hard-to-detect. To reduce the complexity of removing easy-to-detect faults from the fault set, only a small number, say 10000, of random patterns are applied. Those undetected faults are considered to be the hard-to-detect and used as the fault set for U calculation.

Because fault simulation is a costly process, it should be used at most once and only when the circuit already reaches a high fault coverage, say $\geq 95\%$. To automatically determine when to invoke fault simulation, we continuously monitor the detection probability of every fault at each iteration. When the detection probability of every fault is greater than a threshold, say 10^{-6}, it means that the circuit is now quite random testable, fault simulation is invoked.

The results of fault-list and non-fault-list driven TPI are shown in Table 5. The fault-list driven TPI generally achieves higher fault coverages, however, the portion of the CPU time used by fault simulation could be significant. The comparison with the results

Circuit names	w/o Fault list driven			w. Fault list driven		
	#CP/#OP	FC(%)	CPU time	#CP/#OP	FC(%)	CPU time
C2670	4/1	98.31	78 sec.	4/1	98.31	582 sec.
C7552	3/7	98.23	3.4 min.	4/6	98.23	17.4 min.
s3330	7/6	99.92	4.9 min.	2/11	99.92	14.4 min.
s3384	0/9	99.78	128 sec.	0/8	99.78	625 sec.
s9234	16/3	96.10	12.3 min.	16/3	96.45	27.6 min.
s15850	26/8	97.41	34.4 min.	24/10	97.64	58.9 min.
s38417	30/16	99.19	1.7 hr.	26/20	99.26	3 hr.

Table 5: Results of fault list and non-fault list driven TPI

in [10] is shown in Table 6. It shows the results of area-driven TPI for both methods. We would like to point out that the circuits we used were first optimized for performance and therefore, may be different from the circuits used in [10]. Thus, the comparison may not be very accurate. As suggested in Table 6, the fault coverages and the number of test points of the hybrid algorithm are comparable with the results in [10].

Hybrid TPI with Candidates As mentioned in

[1]FC = # of detected faults / Total number of faults

Circuit names	[Tamarapalli and Rajski, 96]							Hybrid algorithm	
	2-phase			Multi-phase					
	#CP/#OP	Ave. FC[2]	Max. FC	#Phases	#CP/#OP	Ave. FC	Max. FC	#CP/#OP	FC
C2670	1/5	100	100	-	-	-	-	4/1	100
C7552	18/2	99.49	99.60	6	18/2	100	100	3/7	99.98
s3330	1/4	99.97	100	-	-	-	-	7/6	99.92
s3384	0/5	100	100	-	-	-	-	0/8	100*
s4863	6/4	99.78	99.83	4	6/4	99.96	100	1/1	100
s9234	8/10	99.80	99.85	3	8/10	99.97	100	15/3	99.88
s15850	15/17	99.16	99.21	4	15/17	99.67	99.71	23/8	99.16*
s38417	18/30	99.79	99.82	3	18/30	99.81	99.85	28/20	99.79*

Table 6: Comparison with results in [9]

Section 2, one problem of the gradient-based method is the requirement of explicitly computing ACRs for a subset of test point candidates. In Section 3.2, we show that the HCR is a very accurate estimate of the ACR and can be computed very efficiently. Therefore, in the second step of the gradient-based method, instead of calculating ACRs, we calculate HCRs for the candidates selected in the first step. We call this modified method as heuristic hybrid algorithm. It significantly improves the runtime efficiency of the gradient-based method. It also runs faster than the hybrid method. Because of high efficiency of the second step, we could select more candidates in the first step to avoid excluding good candidates. We conducted the experiments on three large ISCAS89 benchmark circuits to demonstrate the efficiency of the heuristic hybrid method. As shown in Tables 7 and 8, the heuristic hybrid method runs about 2 times faster than the hybrid method which has been shown in Table 4 to be about 6 times faster than the gradient-based method on average. Unlike the gradient-based method, the fault coverages of the heuristic hybrid method are comparable to those of the hybrid method. It is because we can select more candidates in the first step of the algorithm.

Circuit name	Hybrid			Heuristic hybrid		
	#CP/#OP	FC (%)	CPU	#CP/#OP	FC (%)	CPU
s9234	16/3	96.10	12.3 min.	16/3	96.10	5.4 min.
s15850	26/8	97.41	34.4 min.	26/8	97.41	17.7 min.
s38417	30/16	99.19	1.7 hr.	30/16	99.19	0.62 hr.

Table 7: Comparison of area-driven TPI of hybrid algorithm and heuristic hybrid algorithm

Circuit name	Hybrid			Heuristic hybrid		
	#CP/#OP	FC (%)	CPU	#CP/#OP	FC (%)	CPU
s9234	20/2	96.13	10.7 min.	19/3	96.13	5.4 min.
s15850	23/25	97.14	22.5 min.	23/25	97.14	13.7 min.
s38417	36/12	99.18	1.9 hr.	36/12	99.17	0.61 hr.

Table 8: Comparison of timing-driven TPI of hybrid algorithm and heuristic hybrid algorithm

4 Conclusions

In this paper, we propose a hybrid algorithm for test point selection for scan-based BIST. The algorithm combines both the gradient-based and the explicitly-testability-calculation techniques. The experimental results show that the hybrid algorithm produces very high quality results and uses significantly less amount of CPU time. Several heuristics are also proposed to further improve the performance. The new method achieves a 10~15 times speedup as the gradient-based method and higher fault coverages as well. The experiments for timing-driven TPI were also conducted. The results shows that zero-performance degradation can always be achieved. With a few more test points, the same level of fault coverages as the area-driven version can be obtained. The testing scheme in this paper is assumed to be full scan BIST. The algorithm can be extended for partial scan BIST with slight modification to the equations for computing the HCRs. The partial scan version of the hybrid algorithm is currently under investigation.

[2]FC = (# of detected faults + # of redundant faults)/ Total number of faults

*Obtained by performing fault-list driven TPI

References

[1] F. Brglez, "On Testability of Combinational Networks," *Proc. of Int'l Symposium on Circuits and Systems*, pp.221-225, May 1984.

[2] A.J. Briers and K.A.E. Totton, "Random Pattern Testability By Fast Fault Simulation," *Proc. of Int'l Test Conf.*, pp.274-281, Sep. 1986.

[3] K.T. Cheng and C.J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," *Proc. of Int'l Test Conf.*, pp.506-514, Oct. 1995.

[4] V.S. Iyengar and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs," *Proc. of Int'l Test Conf.* pp.501-508, Aug. 1989.

[5] C.J. Lin, Y. Zorian and S. Bhawmik, "PSBIST: A Partial Scan Based Built-In Self-Test Scheme," *Proc. of Int'l Test Conf.*, pp.507-516, Oct. 1993.

[6] R. Lisanke et al, "Testability-Driven Random Test Pattern Generation," *IEEE Tran. on Computer-Aided Design*, vol. CAD-6, pp.1082-1087, Nov. 1987.

[7] Y. Savaria, M. Youssef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing," *Proc. of Int'l Symposium on Circuits and Systems*, pp.1960-1963, Jun. 1991.

[8] B. Seiss, P. Trouborst, and M. Schalz, "Test Point Insertion for Scan-Based BIST," *Proc. of European Test Conf.*, pp.253-262, Apr. 1991.

[9] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Bryant and A. Sangiovanni-Vincentelli "Sequential Circuit Design Using Synthesis and Optimization," *Proc. of Int'l Conf. on Computer Design*, pp.328-333, Oct. 1992.

[10] N. Tamarapalli and J. Rajski, "Constructive Multi-Phases Test Point Insertion for Scan-Based BIST," *Proc. of Int'l Test Conf.*, pp.649-658, Oct. 1996.

[11] N.A. Touba and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium* pp.2-8, Apr. 1996.