Design-for-Test Methodology for Non-Scan At-Speed Testing*

Mainak Banga, Nikhil Rahagude and Michael S. Hsiao
Bradley Department of ECE, Virginia Tech, Blacksburg, Virginia - 24061
Email: {banga, rahagude, mhsiao}@vt.edu

***Abstract*—While scan-based testing achieves a high fault coverage, it requires long test application times and substantial tester memory, in addition to the overhead in chip area and high test power. Functional testing, on the other hand, suffers from low coverage but can be applied at-speed. In this paper, we propose a novel three-step design-for-test (DFT) methodology which enhances the performance of functional testing to a great extent. In the first step we expand the state space of the circuit beyond functionally reachable space without scan or reset. These new states create conditions to activate/propagate fault effects that are otherwise hard-to-detect. Since structural correlation between D flip-flops (DFFs) of a circuit restricts its state space variation, the second step consists of partitioning the DFFs into different groups that helps to break such correlations. In the third step, we make internal hard-to-observe points in the circuit more observable by directly XORing them with selected primary outputs. This method can be applied at-speed (since no scan shifting is involved) saving significant amount of test application time, with comparable area overhead as scan-based DFT. Our experiments on large ISCAS'89 and ITC'99 benchmarks show that we are able to achieve very high non-scan fault coverages while simultaneously reducing the test application time** ($114\times$) **as compared to scan based techniques.**

I. Introduction

Ensuring that an IC is free from manufacturing defect(s) is a challenge for test engineers. To sieve out defective parts from good ones, test engineers run multiple test patterns on each IC and check their responses against expected values, discarding those which fail the test, to ensure a satisfactory quality of the shipped ICs. Since every single IC needs to be tested against defect(s), test application time of a single IC is a critical factor determining the test cost and profitability of the company.

The quality of a test set is measured as the number of modeled faults it can detect on a given device under test (DUT), and the ratio of faults detected to total number of faults in the DUT is called as *fault coverage*. Special design for testability (DFT) features are often incorporated during the pre-silicon design stage to ease post-manufacturing testing.

There are two major DFT approaches in practice. *Scan based techniques* achieve a very high *fault coverage* at the cost of large test application time, test data volume, and additional hardware overhead. In such a technique, all the DFFs are connected back-to-back in a chain during test mode. Every scan pattern needs to be shifted into the scan chains before they can be applied to the circuit. This shifting is done on the *Test Clock* (T_{CK}) which is often slower than *Functional Clock* (F_{CK}) and hence translates to increased test application time. Variations in scan architecture have been proposed to improve test application time of traditional scan architecture. For example, in [1], [2] and [3], the authors have proposed *Illinois Scan Architecture* (ILS) to speed up scan testing. In ILS, the single scan chain is partitioned into multiple smaller chains. Test data can be broadcast and shifted in parallel to these scan chains thereby reducing the total test application time and test data volume. Since parallel shifting of test data can make some of the otherwise detectable faults undetectable, variations in ILS exist to support single scan chain mode as well. In [4] the authors proposed *California Scan Architecture* (CSA). CSA employs a traditional scan-architecture to shift in test patterns with adjacent filling such that circuit power consumption remains low during scan shift. In test application, it employs selected $\overline{Q}$ signals instead of all Qs from the DFFs.

On the other hand, *functional testing* can be applied at-speed because no scan shifting is involved which is more economical both in terms of test application time and test power. In addition, at-speed testing also helps in detecting delay-related defects. But often, values required on DFFs for exciting/propagating many fault effect(s) are very difficult to achieve in the functional mode. In [5], [6], [7] and [8], the authors have proposed novel sequential ATPG (Automatic Test Pattern Generation) algorithms targeted to improve the fault coverage of sequential circuits. Their results show that achieving a high level of fault coverage for sequential circuits is still a major problem, especially for large circuits with huge state spaces. In [9] and [10], the authors have proposed non-scan DFT techniques for improving the *fault coverage* of non-scan sequential circuits. Our current work falls under this theme, although it involves less overhead in terms of additional hardware as compared to these works.

In this paper, we propose a novel three-step non-scan DFT approach to enhance the *fault coverage* of a DUT. We use both Q and $\overline{Q}$ signals of a subset of DFFs to help activating/propagating many more fault effects to primary output(s). We propose a heuristic based approach to partition the DFFs into multiple groups and select a specific group using test inputs which we call as *ENABLE* pins. Incorporating the $\overline{Q}$ values of only a subset of DFFs (instead of all DFFs) sufficiently help to diversify the state space. Normal scan based testing uses *scan enable*(SE), *scan in*(SI), *scan out*(SO) and *scan mode*(SM)(Single chain or ILS mode) input pins to test the DUT. We also use a maximum of 4 input pins for our proposed DFT architecture so that there is no additional overhead in terms of pin count. Finally, by selecting a set of hard-to-observe points and making them observable at outputs, we further enhance the *fault coverage*. No additional output pins are needed as we use existing outputs for observing fault effects. Since no scan shifting is involved, this methodology can be applied at-speed that helps saving test application time as well as allowing for catching delay-related defects. Our

* This work is supported in part by NSF grant 1016675
978-3-9810801-7-9/DATE11/©2011 EDAA

results on ISCAS'89 and ITC'99 benchmarks show promising results in terms of the *fault coverage* achieved, with significantly reduced test application times.

The paper is organized as follows: Section II explains the three steps of our methodology in detail. Section III talks about the setup of our experiments and IV discusses the *fault coverage* and speedup achieved on benchmarks for both random/sustained random as well as state-of-art ATPG tool generated test set. Finally, Section V concludes the paper and discusses possible future directions.

II. Our Approach

Our approach consists of three steps. In the first step, we modify the DFFs to make $\overline{Q}$ usable for creating new state vectors in the circuit. In the next step, we partition the DFFs into multiple groups to selectively use their $\overline{Q}$ signal, thereby breaking the DFF correlations and enriching the state space. In the final step, we increase the observability of a set of selected hard-to-observe internal nodes by making them directly observable at existing outputs of the circuit.

A. Enabling $\overline{Q}$

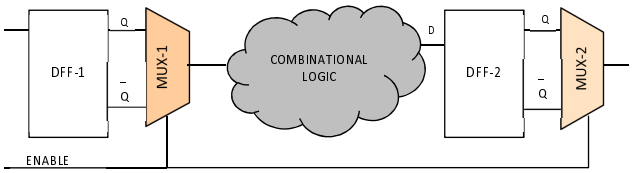


Fig. 1. Modifying flip-flops to enable D-BAR

In this work, we expand the state space of the DUT by utilizing $\overline{Q}$ values of DFFs along with Q. We modify the output of a conventional DFF by multiplexing the Q and $\overline{Q}$ signals through a 2×1 multiplexer and selecting either of them based on *ENABLE* signal of the multiplexer (refer to Fig. 1). The *ENABLE* signal is determined by input pin(s) of the DUT and is used to select the normal or complemented value from the DFFs. This modification does not add any hardware overhead as compared to normal scan DFF. Essentially, we are taking the multiplexer which was placed in front of the DFF in normal scan architecture to the output end of the DFF. By allowing for either Q or $\overline{Q}$ values, we effectively increase the controllability of many hard-to-control signals, thereby significantly enhancing the testability of the logic. In other words, the inclusion of $\overline{Q}$ along with Q helps creating new conditions in the combinational logic for excitation/propagation of fault effects which were otherwise difficult to detect.

B. Flip-flop Partitioning

Controlling all the DFFs using a single *ENABLE* restricts the state space variation to an extent. In a circuit, many DFFs are interdependent, i.e., their values are correlated. Such correlations must be broken to create more variations in state space. For example, if a pair of DFFs A and B are equivalent, $Q_A \leftrightarrow Q_B$; then $(1,1)$ and $(0,0)$ are the only two possible combinations possible using a single *ENABLE* pin. $(1,0)$ and $(0,1)$ would not be achievable under this single *ENABLE* configuration. To resolve this problem, we partition the DFF set into different groups so that a minimum number of correlated DFFs are placed in a group. We use multiple input pins and use a decoder to create several *ENABLEs* each of which is used to select a specific group. Selecting $\overline{Q}$ from a part of the DFFs instead of the entire DFF set helps in creating variation in states obtained on DFF outputs which are not achievable using a single *ENABLE*. Conceptually the achievable state space under all such configuration(s) (single/multiple *ENABLES*) lies in between the total state space and functional state space. This is shown in Fig. 2

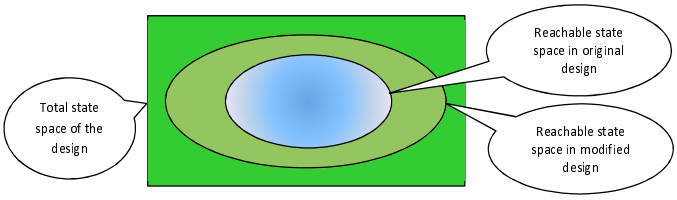


Fig. 2. Extent of reachable state space using our approach

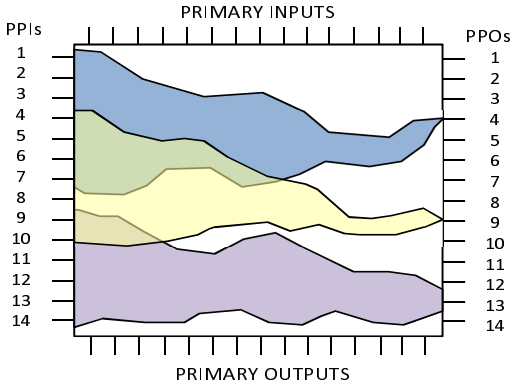


Fig. 3. Partitioning scheme for flip-flops into different groups

Fig. 3 shows the fanin cone of three DFFs viz. 4, 9 and 13. It is evident that the overlap in fanin cones of DFFs 4 and 9 is more than that between DFFs 9 and 13. This implies DFFs 4 and 9 have more common inputs than DFFs 9 and 13. When two gates are controlled by a common set of inputs, chances are high that their logic values are correlated. To break this structural correlation we place DFF 4 in one group and 9 in the other. Since DFF 13 has no fanin cone overlap with DFF 4, it is placed in the same group containing DFF 4. DFF partitioning has been used in the past to reduce test data volume in scan based testing [11].

The DFF grouping procedure is explained using ISCAS'89 benchmark circuit s298. This circuit has 14 DFFs viz. 1 through 14 and their fanin cone matrix is shown in Fig 4 (i). An entry of 1 at the intersection of DFF A (represented as row index) and DFF B (represented as column index) indicates that DFF B is in the fanin cone (FC) of DFF A. The fanin cone of a DFF X is defined by the following set:

$$FC_X = \{g \mid g\, is\, a\, gate\, in\, the\, fanin\, cone\, of\, X\} \quad (1)$$

Thus the fanin cone of DFF 9 consists of DFFs $FC_9 = \{2, 3, 4, 5, 6, 9, 13\}$ and is represented by the following vector $\langle 01111100100010 \rangle$ where the index 9 refers to the DFF number.

For a group of DFFs G, the fanin cone (FC) of G is defined as the union of the fanin cones of individual DFFs that comprise G.

$$FC_G = \bigcup_{n=1}^{n=i} FC_n \tag{2}$$

We define the fanin count C of a DFF or a group of DFFs as the cardinality of its fanin cone.

$$C_X = |FC_X| \; or \; C_G = |FC_G| \tag{3}$$

For the DFFs in s298, their cardinalities are $C_1 = 1, C_2 = 4, C_3 = 3, C_4 = 4, C_5 = 6, C_6 = 6, C_7 = 7, C_8 = 7, C_9 = 7, C_{10} = 8, C_{11} = 8, C_{12} = 7, C_{13} = 1, C_{14} = 1$.

Overlap between two DFF groups G_x and G_y is defined as the summation of the dot product of their fanin cone vectors

$$O_{G_x}^{G_y} = \Sigma FC_{G_x} \cdot FC_{G_y} \tag{4}$$

This is to note here that a group can consist of a single DFF as well.

Given a target number of partitions to be formed within a set of DFFs, we begin by placing the DFF with maximum cardinality in GROUP 1 (G_1). For example, in s298 both DFFs 10 and 11 have cardinality of 8. For creating 3 different groups we select the first one, i.e., DFF 10 and place it in G_1. We define *selected group* (SG) as the group of DFFs that have already been selected in some group. Currently SG for s298 consists of DFF 10 only. Thus the fanin-cone $FC_{SG} = \langle 11111100010010 \rangle$. For selecting the first DFF of all subsequent groups G_2 and G_3 in our running example), we compute the overlap of all remaining DFFs with SG individually. For s298, the fanin cone overlap for DFF 1 with current SG is given by $O_{SG}^1 = \Sigma \langle 11111100010010 \rangle \cdot \langle 10000000000000 \rangle = 1$. Similarly the overlap with all other DFFs with SG is $O_{SG}^2 = 4, O_{SG}^3 = 3, O_{SG}^4 = 4, O_{SG}^5 = 5, O_{SG}^6 = 6, O_{SG}^7 = 6, O_{SG}^8 = 6, O_{SG}^9 = 6, O_{SG}^{11} = 7, O_{SG}^{12} = 6, O_{SG}^{13} = 1, O_{SG}^{14} = 0$. This is to note that there is no entry for DFF 10 as it has already been selected in G_1. The highest overlap of SG is with DFF 11 and so DFF 11 is placed in G_2. SG now consists of DFF 10 and 11. Thus, $FC_{SG} = \langle 11111100011010 \rangle$. The next DFF is selected based on the individual overlap of the remaining DFFs with current SG. This overlap comes out to be $O_{SG}^1 = 1, O_{SG}^2 = 4, O_{SG}^3 = 3, O_{SG}^4 = 4, O_{SG}^5 = 5, O_{SG}^6 = 6, O_{SG}^7 = 6, O_{SG}^8 = 6, O_{SG}^9 = 6, O_{SG}^{12} = 6, O_{SG}^{13} = 1, O_{SG}^{14} = 0$. Hence the first DFF of G_3 is DFF 6. (Although DFF 7, 8, 9 and 12 have the same overlap, we select the first one).

Once each group has been assigned a DFF, the overlap criteria for selecting subsequent DFFs in any group is reversed from maximum overlap to minimum overlap. For selecting the next DFF in G_1, we compute the overlap of the remaining DFFs (those which are yet to be selected) with the existing DFF(s) in G_1, selecting one with minimum overlap. For s298, the overlap of remaining DFFs with G_1 is given by $O_{G_1}^1 = 1, O_{G_1}^2 = 4, O_{G_1}^3 = 3, O_{G_1}^4 = 4, O_{G_1}^5 = 5, O_{G_1}^7 = 6, O_{G_1}^8 = 6, O_{G_1}^9 = 6, O_{G_1}^{12} = 6, O_{G_1}^{13} = 1, O_{G_1}^{14} = 0$. Since DFF 14 has the minimum overlap, it is placed in G_1 along with DFF 10. All the rest of the DFFs are selected in the same manner. Selecting the DFFs with minimum overlap helps in breaking structural correlations among DFFs in a group. In other words, DFFs which are structurally correlated get segregated into different groups. Since each group has independent *ENABLE* signals, such a grouping scheme ensures that their $\overline{Q}$ output are independently selected irrespective of the DFF to which it is correlated. Fig 4 (iii) shows the composition of the groups for a partition count of 7.

FF IDs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	0	0	0	0	0	0	0	0	1
6	0	1	1	1	1	1	0	0	0	0	0	0	1	0
7	0	1	1	1	1	1	1	0	0	0	0	0	1	0
8	0	1	1	1	1	1	0	1	0	0	0	0	1	0
9	0	1	1	1	1	1	0	0	1	0	0	0	1	0
10	1	1	1	1	1	1	0	0	0	1	0	0	1	0
11	1	1	1	1	1	1	0	0	0	0	1	0	1	0
12	0	1	1	1	1	1	0	0	0	0	0	1	1	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(i) Fanin-cone matrix for flip-flops in s298

G1	10	14	3	5	9
G2	11	1	2	7	12
G3	6	13	4	8	

(ii) Flip-flop groups with enable count 2

G1	10	14
G2	11	1
G3	6	13
G4	7	3
G5	8	2
G6	9	4
G7	12	5

(iii) Flip-flop groups with enable count 3

Fig. 4. (i) Flip-flop fanin cone matrix for s298 (ii) Flip-flop grouping for 2 enable pins for s298 (iii) Flip-flop grouping for 3 enable pins for s298

C. Observability Enhancement

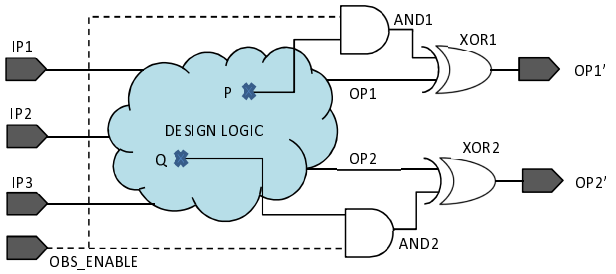


Fig. 5. Observability modification for hard-to-observe nodes

In this step, we attempt to increase the observability of a selected set of hard-to-observe internal nodes. The concept of *Observability Enhancement* is shown in Fig. 5. P and Q are two hard-to-observe points in the given circuit. For each signal, we use a AND gate to select the point in the test

mode and XOR them with an existing primary output to make them observable. The first input of the AND gate is a hard-to-observe signal and the second input is OBS_ENABLE signal. In the normal functional mode, OBS_ENABLE is set to 0 and hence only functional logic value reaches the final output $OP1'$ and $OP2'$. In the test mode, OBS_ENABLE is set to 1. If any fault effect appears on either P or Q, it will immediately propagate to $OP1'$ or $OP2'$ because the XOR gate will not block the fault effect (unless the other input of the XOR gate also contains a fault effect from the same fault).

Selecting suitable hard-to-observe points in a circuit can be tricky. It may be the case where there are more hard-to-observe points than the number of outputs in a circuit. One approach could be building multiple XOR trees to include all such points and associating each XOR tree with one of the outputs. But XOR trees involve much hardware overhead and is not preferred. Alternatively, if we associate each output with single internal hard-to-observe point, the number of points we can observe is exactly the number of outputs.

To tackle this problem we select our hard-to-observe points based on hard-to-observe *regions*. For each gate in the circuit we compute its fanin cone till a depth of 3. We run a random + sustained random vector set on the circuit with modified and partitioned DFFs and obtain a list of excited but undetected faults. We grade the gates present in this list by the number of fault instances that are undetected in its *region*. For example, if a 2 input gate T has 2 faults on its input and 1 fault on its output as excited but undetected, then the *fault weight* (FW) assigned to this gate is 3. For a *region* we define its *fault weight* as the summation of fault weights of all gates included in that *region*.

$$FW_R = \sum_{i=1}^{i=N} FW_{T_i} | T_i \epsilon R \tag{5}$$

where FW_R represents the *fault weight* of *region* R, FW_{T_i} represents the *fault weight* of i^{th} gate in *region* R and R contains a total of N gates.

As an example, let's consider the *region* shown by the dotted line in Fig. 6. This *region* stems out from the fanin cone of gate 9 till a depth of 3 levels and contains gates $2, 3, 4, 5, 6, 7, 8, 9$ and $DFF-2$. Gates which have undetected faults on them are tabulated in top right corner of Fig. 6. Gates $2, 3, 4$ and 5 have undetected faults on them and *fault weight* corresponding to each one of them is shown under the $FaultWeight$ column in the figure. So, *fault weight* of the *region* is computed as $FW_{R_9} = FW_2 + FW_3 + FW_4 + FW_5 = 1 + 2 + 3 + 3 = 9$.

We arrange all the *regions* in the circuit in descending order of their *fault weights* and select regions from the top of the list. In our implementation, the number of *regions* selected is equal to the number of outputs. In essence, we are trying to maximize the number of hard-to-observe points that can be made observable without incurring much hardware overhead compared to an XOR tree.

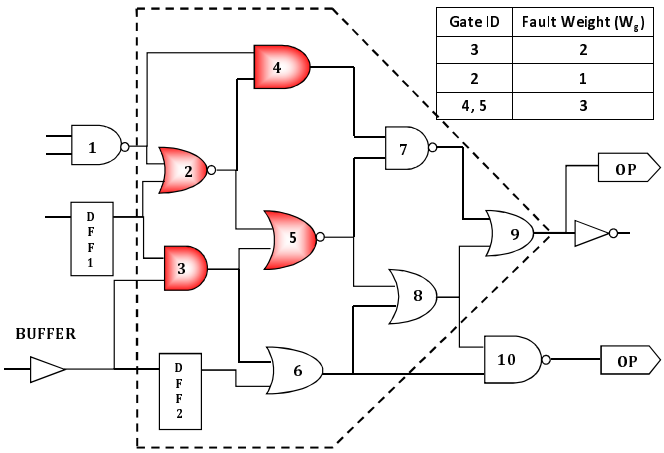


Fig. 6. Computing *fault weight* of a *region*

III. Experimental Setup

We used ISCAS'89 and ITC'99 sequential circuits to evaluate the effectiveness of our approach. We report the results only for those circuits with more than 50 DFFs, i.e., a potential state space of $> 2^{50}$ states. For each circuit, we modified the existing DFFs by adding a multiplexer, thus allowing Q and $\overline{Q}$ signals to propagate to the combinational logic. We experimented using 1, 2 and 3 input *ENABLE* pins for creating the *ENABLE* signals to select DFF groups. A decoder structure is used to create the actual *ENABLEs* for each group. Thus, for k *ENABLE* input pins, we can uniquely select 2^k groups. Out of these 2^k possible selects, we reserve one for the functional mode in which none of the groups are selected to use $\overline{Q}$ signal. The remaining are assigned to address different groups. Thus the number of groups addressable for k *ENABLE* input pins is $2^k - 1$.

We create a vector set consisting of random and sustained random vectors. In sustained random vectors we do not change the input vector for a specific number of time frames (we sustain it)! and allow current state in the circuit to create the next state. This concept was introduced in [12] in which the authors sustained pseudo random sequence generated by a LFSR to make the DUT transition through different states. Later in [13], the authors showed it to be very effective in enhancing *fault coverage* of some hard-to-test circuits. For exercising each group (including the group containing entire DFF set in normal functional mode), we generate 25,000 random vectors and 1000 sustained random vectors, each sustained 25 times making a vector set size of 50,000 vectors. Thus, total vector set for a circuit with single *ENABLE* pin consists of 100,000 (once with *ENABLE* high and once with *ENABLE* low) vectors. To make the comparison of *fault coverage* fair, for the original circuit (which doesn't have an *ENABLE* input) we repeat the same random + sustained random vector set twice (also making it 100,000 vectors). The runtime of our algorithm is in the order of minutes for large circuits but considering this is a one time procedure it is not a major concern.

Our experiments are conducted in two stages. In stage 1

we modify the DFFs, create the groups using the partitioning algorithm, add *ENABLE* pins and create the decoder logic to utilize the *ENABLE* pins for selecting different DFF groups. We fault simulate a random + sustained random vector set to create a list of faults that are excited but undetected. In the second stage, we use this list of undetected faults to select the observation points using step-3 of II. After we modify the hard-to-observe points to make them observable, we again fault simulate the new circuit. Our experimental results report the final coverage achieved after all the modifications have been done on the circuit. We also run the circuit with 3 *ENABLE* input pins through a state-of-art sequential ATPG tool to improve the *fault coverage* even more.

IV. Experimental Results

Table I shows the *fault coverage*, *fault efficiency* and reduction in test time for large ISCAS'89 and ITC'99 benchmarks using our proposed methodology. "CKT STATS" contains the circuit attributes viz. the circuit name represented under "CKT NAME" column, number of inputs and outputs in the original circuit under the "INPUT/OUTPUT" column and the number of DFFs under the "DFF COUNT" column. "FAULT COVERAGE" reports the *fault coverage* achieved by using different number of *ENABLE* pins as proposed in the theory. Under this section, "NO-ENABLE", "1-ENABLE", "2-ENABLE" and "3-ENABLE" represent the *fault coverage* achieved in % by using 0, 1, 2 and 3 *ENABLE* pins respectively. We run the sequential ATPG only in the "3-ENABLE" configuration. For this, we report both *fault coverage* and *fault efficiency* which are the sub-columns under "SEQ ATPG W/ 3-ENABLE". The column "TEST TIME RED." represents the reduction in test application time for our methodology relative to a scan configuration (single scan chain or ILS structure depending on DFF count of the circuit). For all the configuration with one or more *ENABLE* pins, we have one OBS_ENABLE pin to control the observability of hard-to-observe points. In comparison to the original circuit, additional gates are required to enable observability of internal nodes in the modified circuits. For three different configurations of modified circuits, the number of gates in the decoder logic is different. Hence total fault count in the fault list of a given circuit is different for all the four different configurations (*NO-ENABLE*, *1-ENABLE*, *2-ENABLE* and *3-ENABLE*). To make fair comparisons we report the total *fault coverage* including all those new faults which are introduced in the circuit because of the addition of extra hardware.

In Table I, results on "FAULT COVERAGE" show the general trend that *fault coverage* increases as we increase the number of *ENABLE* pins. This is expected because as we increase the granularity of the DFF sets it creates new states which help exciting/propagating more fault effects to the output. In general, sequential ATPG is able to achieve a higher *fault coverage* for all the circuits with 3 *ENABLE* signals.

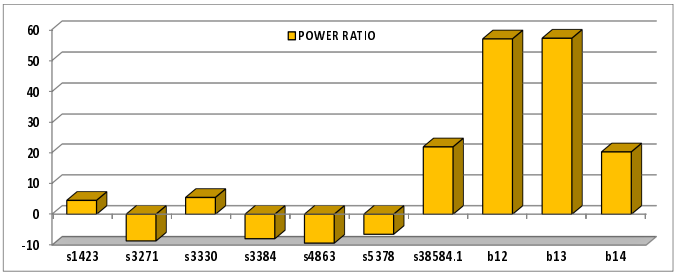
For circuits like $s1423$, $s3330$, $s5378$, $s38584.1$, $b12$, $b13$ and $b14$ which were otherwise hard to test using traditional sequential ATPG, we have been able to improve the *fault coverage* by more than 10% using our modified circuit configuration; sometimes the improvement is as high as 68% for $b12$, nearly 30% for s38584.1 and 27% for s5378. For smaller benchmark circuits our results were comparable but since they are highly testable with existing approaches we did not explicitly report their numbers in Table I. We also didn't report the results for $s35932$ because it is already highly testable without any modification. For such circuits, adding testability features would be an overhead. Likewise, we did not report circuits such as s15850, etc., since they are not initializable; thus, both Q and $\overline{Q}$ would remain unknowns. On an average, the final test set obtained by sequential ATPG achieves a *fault coverage* of 97% and *fault efficiency* of 98%.

For computing the reduction in test application (shown in "TEST TIME RED." column) we made the following assumptions. Circuits with less than 100 DFFs are assumed to have a single scan chain and those with more than 100 DFFs have 16 balanced scan chains. Thus $s3271$, $s3330$, $s3384$, $s4863$, $s5378$, $s38584.1$, $b12$ and $b14$ have 16 chains for broadcasting the test patterns. If there are N patterns in the functional ATPG test set, we would need $N * F_{CK}$ time to run the complete test set, where F_{CK} is the period of the functional clock. For a circuit with single scan chain, assuming we have M test patterns and D DFFs in the circuit; total test time required is $((M + 1) * D * T_{CK}) + M * F_{CK})$. For circuits with multiple scan chains, the time required will be $((M + 1) * D/16 * T_{CK}) + M * F_{CK})$. The first part of the above expression(s) $((M+1)*D*T_{CK}))$ represent scan shift-in time (also shift-out which is multiplexed with shift-in) while the second part $(M*F_{CK}))$ represent test application time. For M test patterns we need to scan in all of them and scan out the last pattern after the test is applied. So the multiplying factor is $(M + 1)$. Since T_{CK} is much slower than F_{CK}, the scan-shift in time dominates the total test time required. Test clock T_{CK} in our experiments is assumed to be 10 times [14] slower than the functional clock. The test sets for full scan testing is obtained using a combinational ATPG tool with fault dropping. Although for ILS mode we have assumed that the scan chains can be partitioned without dropping *fault coverage*, this may not be entirely possible. Thus the actual time required for scan testing in ILS mode will be little higher because all faults are not testable in ILS broadcast mode. For our functional test set generated by sequential ATPG, we have used compaction techniques to reduce the size of the test set before computing the test application time. In our results, we can see orders of magnitude reduction in the test application time for circuits like $s1423$, $s3384$, $s38584.1$ and $b13$. When millions of ICs are subjected to test, such a reduction in testing time of a single part translates to a huge saving in overall test time.

Fig. 7 shows the increase/decrease in average test power using our approach. In [15],[16] the authors have shown that scan shift in power could be significantly high as compared to the functional power. For most circuits, our experiments show that the test power is comparable to the power in functional mode. In fact, the test power could sometimes be lower, as in

TABLE I
FAULT COVERAGE AND TEST TIME REDUCTION FOR ISCAS'89 AND ITC'99 BENCHMARKS USING OUR METHODOLOGY

CIRCUIT STATS			FAULT COVERAGE						TEST
CKT NAME	INPUT/ OUTPUT	DFF COUNT	NO-ENABLE FC(%)	1-ENABLE FC(%)	2-ENABLE FC(%)	3-ENABLE FC(%)	SEQ ATPG W/ 3-ENABLE		TIME RED.
							FC(%)	FE(%)	
s1423	17/5	74	80.00	86.42	96.04	96.11	**96.42**	97.06	**167.48**
s3271	26/14	116	99.20	98.49	99.40	99.43	**99.98**	99.98	**58.62**
s3330	40/73	132	73.45	89.32	92.47	94.94	**96.65**	97.34	**90.05**
s3384	26/14	183	91.15	96.64	96.79	96.99	**97.94**	97.94	**165.83**
s4863	49/16	104	95.89	96.51	98.14	98.81	**99.83**	99.93	**66.37**
s5378	35/49	179	69.56	83.62	81.97	86.65	**96.96**	97.81	**58.51**
s38584.1	38/304	1426	66.79	88.57	91.50	91.42	**96.25**	98.90	**301.70**
b12	6/6	121	30.17	39.19	57.81	66.04	**98.18**	98.18	**12.67**
b13	11/10	53	60.41	78.65	91.00	91.81	**96.85**	97.58	**194.94**
b14	33/54	247	64.29	85.71	87.60	88.87	**95.15**	97.49	**19.24**
AVERAGE			73.09	84.31	89.27	91.11	**97.42**	**98.22**	**113.54**



Note: The disproportionate power in b12 and b13 is due to the fact that original circuits were very untestable with low signal toggles in the circuit

Fig. 7. Ratio of average power in test mode using our approach to normal functional mode

s3271, s3384, s4863 and s5378. In two circuits, $b12$ and $b13$, the test power was unusually higher. This is because these two circuits are originally very hard to test (as shown by their low *fault coverage* without using our approach), as certain portions of logic in these circuits are very hard to toggle. With our modified circuit structure, those portions become sufficiently active, hence improving *fault coverage* and thereby adding to the average circuit power.

When compared to [9] and [10], they required several more input and output pins; on the other hand, our approach uses a fixed count (4) of input pins and no additional output pins. For example, in $s38584.1$, 11 additional inputs and 12 outputs have been used in [10] which added to considerable hardware overhead. With less pins, we are still able to achieve a higher *fault coverage* and *fault efficiency* for the same circuit. In our case the test application time is improved by a factor of $300\times$ as compared to $10\times$ improvement in the previous works. For the remaining circuits, the *fault coverages* are comparable, but with a speed up in test application time in our approach.

V. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a three step technique to boost the performance of non-scan DFT substantially. In the first step we modified the existing usage of DFFs in a circuit by utilizing $\overline{Q}$ and thus extending the state space of the DUT beyond the functional limit. Partitioning the DFFs into different groups and using their $\overline{Q}$ selectively helps create more state variations helping in enhanced *fault coverage*. In stage three we select hard-to-observe points in the circuit and make them easily observable at the output so that any fault effect propagated to these points are not masked. Experimental results on ISCAS'89 and ITC'99 benchmarks shows marked improvement in the coverage of those circuits which were earlier hard to test. At-speed testing is possible using this methodology thus reducing the test application time by 2 orders of magnitude.

As a future direction, variation in group selection heuristic would be interesting to investigate. Selection of hard-to-observe point using other heuristic is also possible. Moreover, constructing a XOR tree to include a larger set of hard-to-observe points should further enhance the *fault coverage*, but one has to strike a balance between the hardware overhead incurred in building such a tree and the improvement achieved.

REFERENCES

[1] F. F. Hsu, K. M. Butler and J. H. Patel, *A case study on the implementation of the Illinois Scan Architecture*, ITC 2001, pp. 538-547.

[2] I. Hamzaoglu and J. .H. Patel, *Reducing test application time for full scan embedded cores*, Fault-Tolerant Comp 1999, pp. 260-267.

[3] K-J. Lee, J-J. Chen and C-H. Huang, *Using a single input to support multiple scan chains*, ICCAD 1988, pp. 74-78.

[4] K. Y. Cho, S. Mitra, E. J. McCluskey, *California scan architecture for high quality and low power testing*, ITC 2007.

[5] T. Niermann and J. H. Patel, *HITEC: a test generation package for sequential circuits*, DATE 1991, pp. 214-218.

[6] M. S. Hsiao, E. M. Rudnick and J. H. Patel, *Sequential circuit test generation using dynamic state traversal*, DATE 1997, pp. 22-28.

[7] X. Lin, I. Pomeranz and S. M. Reddy, *MIX: a test generation system for synchronous sequential circuits*, VLSI Design Conf. 1998, pp. 456-463.

[8] A. Giani, S. Sheng, M. S. Hsiao, and V. Agrawal, *Efficient spectral techniques for sequential ATPG*, DATE 2001, pp. 204-208.

[9] V. Chickermane and J. H. Patel, *An optimization based approach to the partial scan design problem*, ITC 1990, pp. 377-386.

[10] M. S. Hsiao and M. Banga, *Kiss the Scan Goodbye: A Non-Scan Architecture for High Coverage, Low Test Data Volume and Low Test Application Time*, ATS 2009, pp. 225-230.

[11] M. Ashouei, A. Chatterjee, A. Singh, *Test volume reduction via flip-flop compatibility analysis for balanced parallel scan*, Current and Defect Based Testing 2004, pp.105-109.

[12] L. Nechman, K. K. Saluja, S. Upadhyaya and R. Reuse, *Random Pattern Testing for Seq. Ckts Revisited*, Fault Tolerant Comp. 1996, pp. 44-52.

[13] R. Guo, S. Reddy and I. Pomeranz, *PROPTEST: A Property Based Test Pattern Gen. for Seq. Ckts. Using Test Comp.*, DAC, 1999, pp. 653-659.

[14] H. Hashempour, F. J. Meyer and F. Lombardi, *Test time reduction in a manufacturing environment by combining BIST and ATE* Defect and Fault Tolerance in VLSI Systems 2002, pp 186-194.

[15] S. Almukhaizim and O. Sinanoglu, *Peak power reduction through dynamic partitioning of scan chains*, ITC 2008, pp 1-10.

[16] S. Remersaro, X. Lin, Z. Zhang, S. M. Reddy, I. Pomeranz and J. Rajski, *Preferred fill: a scalable method to reduce capture power for scan-based designs*, ITC 2006, pp 1-10.