Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST

Kwang-Ting (Tim) Cheng
Dept. of ECE
University of California
Santa Barbara, CA 93016

Chih-Jen Lin
AT&T Bell Labs.
Engineering Research Center
Princeton, NJ 08542

Abstract - We propose timing-driven test point insertion methods for a full-scan based BIST scheme and for a partial-scan based BIST scheme, where the global flip-flop cycles have been broken by the scan flip-flops. The objective is to minimize the performance as well as the area impact due to the insertion of test points while achieving a high fault coverage under the pseudo-random BIST schems. The gradient-based method is used and extended to estimate the random-pattern testability improvement factors for the test point candidates of either full-scan based or partial-scan based BIST. We also propose a symbolic computation technique to compute testability for circuits under the partial-scan based BIST scheme. Our experimental results show that the performance degradation of test point insertion could be unacceptably high if the cost function used for test point selection does not include the performance penalty. Using our timing-driven algorithm, zero performance degradation and a high fault coverage can always be achieved using a small number of test points.

1. Introduction

The ever increasing costs of testing have made design for testability (DFT) a necessity. Among various structured DFT schemes proposed, full-scan, such as Level Sensitive Scan Design (LSSD) introduced two decades ago, has been used for random logic and studied most thoroughly. Lately, partial-scan has been proposed to reduce the area overhead by scanning only a small portion of all the flip-flops in a circuit. Both full-scan and partial-scan, however, require porting and maintaining of often huge test vector sets, which makes it difficult and costly to apply the test vector sets beyond IC level tests so as to achieve an acceptable vertical test integration. In addition, it is difficult to apply large scan vectors during environmental stress tests, including burn-in tests, due to the limited bandwidth for transferring test vectors. Various Built-In Self-Test (BIST) schemes for random logic have been proposed to contain the above mentioned problems of scan test. Also, BIST schemes provide other advantages, such as at-speed testing, easy reuse of the built-in test structure for diagnosis and repair. Finally, one salient feature of BIST that would make it stand out in design and test of VLSI and ULSI is as described below. Extremely complex devices call for top-down modular design methodology to contain the problems of cost and time to market. BIST suits this methodology very well because it is usually applied on individual structural blocks of VLSI (random logic, RAM, ROM, etc.) using a divide-and-conquer approach [2], in which every block has its own embedded BIST structure and is controlled by a chip-wide BIST controller.

An ideal BIST scheme, however, has to achieve not only very high fault coverage with minimum area overhead but also low or zero performance degradation. Full scan based random pattern testing BIST, called FSBIST, such as the one in [1] does not guarantee a very high fault coverage because many practical circuits contain some random pattern resistant faults. Therefore, clever methods to insert test points to guarantee a high fault coverage with a small area overhead were proposed in [3][4]. Recently, a partial-scan based BIST (PSBIST) scheme has been proposed [5], in which a method of selecting test points is also proposed to guarantee achieving a high fault coverage under pseudo-random patterns. Compared to full-scan BIST, however, PSBIST incurs less performance degradation because only a subset of the flip-flops are selected for scan—this is even more true if the flip-flops selection algorithm is timing-driven [6]. Nonetheless, in both full- and partial-scan based BIST test point insertion may cause performance degradation when test points are inserted on critical paths. In this paper, we propose timing-driven test point insertion methods, which have very low or zero performance degradation.

The objective is to minimize the performance as well as the area impact due to the insertion of test points while achieving a high fault coverage under the pseudo-random BIST scheme. Our method uses a greedy approach to select one test point at each iteration. The selection is based on a cost function of test point candidates which comprises two components: (1) the profit of global random pattern testability and (2) the penalty of timing. We use and extend the gradient-based method to estimate the random-pattern testability improvement factors for the test

INTERNATIONAL TEST CONFERENCE

point candidates of both full-scan based BIST and partial-scan based BIST. In addition, a symbolic computation technique is proposed to efficiently compute the testability of designs under partial-scan based BIST. Thorough experiments have been conducted to illustrate the effectiveness of our method and the trade-off among the performance degradation, the fault coverage, and the number of test points required. Our results indicate that the performance degradation of test point insertion could be unacceptably high if the cost function used for test point selection does not include the performance penalty. Most important, zero performance degradation and a high fault coverage can always be achieved by our method using a small number of test points.

2. Timing-Driven Test Point Insertion for Full Scan BIST

Figure 1 illustrates the structure of our version of full-scan based BIST, which is incorporated into the circuit through the following three steps:

1. Replace all flip-flops with scan flip-flops, then connect these scan flip-flops into scan chains.
2. Add a Test Pattern Generator (TPG) to supply random patterns to primary inputs and, via scan chains, to pseudo-inputs (outputs of those scanned flip-flops). Similarly, add an Output Data Compactor (ODC) to compact the data from primary outputs and, via scan chain, from pseudo-outputs (inputs of those scanned flip-flops). The TPG consists of a Linear Feedback Shift Register (LFSR) and a Phase Shifter (PS). PS is used to prevent the structure dependency from occurring among the outputs of TPG, thereby avoiding the degradation of randomness of those output signals [7]. The ODC consists of a Multiple Input Signature Register (MISR) and a Space Compactors (SC) [8].
3. Add test points to increase the random pattern testability of the circuit.

In the following, we describe briefly how test points, control points and observation points, are inserted into a circuit and how a test point affects the testability of a circuit.

An observation point is inserted at a node to improve the observabilities of the node and all the other nodes that feed the node directly or indirectly. The effect of inserting an observation point, as shown in Figure 2c, is illustrated by the shaded region in Figure 2a. Actual implementation of an observation point is done by simply connecting the node to the output data compactor. A control point is inserted at a node to improve controllabilities as well as observabilities of nodes in a circuit. Changing the controllability of a node changes also the controllabilities of nodes influenced by the node, as indicated in the shaded region of Figure 3a. In addition, the observabilities of nodes in the hatched area,

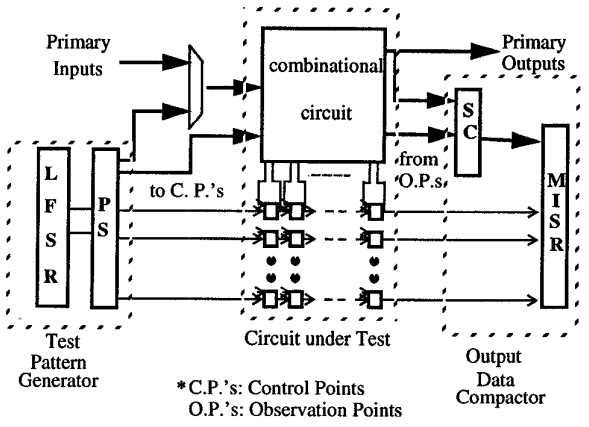

Figure 1: Full-scan based BIST structure

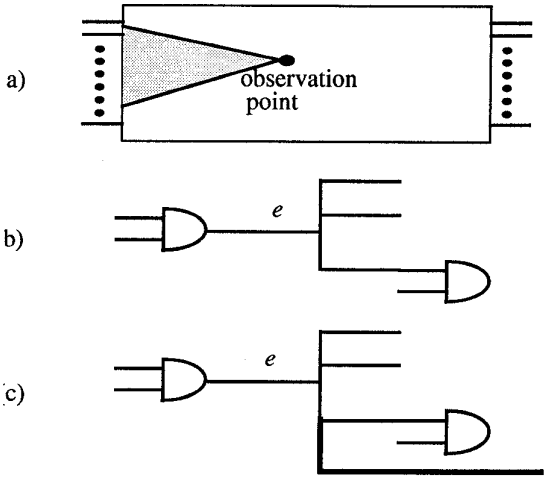

Figure 2: Adding an observation point
a) region influenced by an observation point
b) circuit before adding an observation point
c) circuit after adding an observation point

which includes the shaded area, are altered. Figure 3b and Figure 3c illustrate the circuits before and after a control point is inserted. Signal r is connected to a random source whose 1-controllability (defined as the probability of having a logic value 1) is 0.5 at the BIST mode and 0 at the normal mode. The added gate G in Figure 3c is either an OR gate or an AND gate. If the 1-controllability of e is too small, an OR gate is inserted such that during the BIST mode the 1-controllability of e' is higher than 0.5. On the other hand, if the 1-controllability of e is too large, an AND gate is added such that the 1-controllability of e' is smaller than 0.5. It is worth noting that, regardless of which gate is added, the observabilities of all the nodes that affect e are reduced–in the worst case reduced by half.

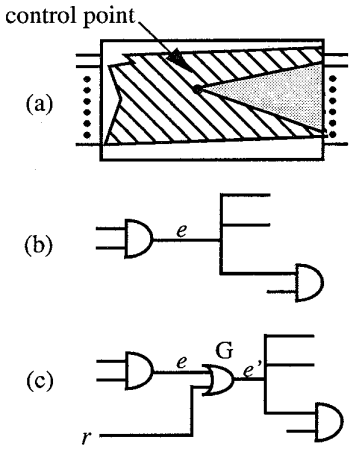

Figure 3: Adding a control point
a) region influenced by a control point
b) circuit before adding a control point
c) circuit after adding a control point

Our goal is to minimize the performance as well as the area impact due to the insertion of test points and to achieve a high fault coverage under the pseudo-random BIST scheme. The sketch of our timing-driven test point selection algorithm is the following. Given a circuit, timing analysis is first performed and the slack of each signal is recorded. The slack of a signal is defined as the difference of the required arrival time and the actual arrival time at the signal. For each signal, we calculate a testability improvement factor to estimate the improvement on random pattern testability obtained from the insertion of a control or an observation point. The calculation of this factor for all signals, which is based on the *gradient* method (that will be reviewed in the next subsection), can be done in *linear* time. A set of signals whose timing slacks as well as the testability improvement factors are larger than given thresholds are selected as candidate test points. Simulation is then performed for each of the candidates to calculate its actual testability improvement. We then select the signal that achieves the biggest improvement as the test point for insertion. After a new test point is inserted, we recalculate the slacks for all signals. The process continues until a pre-specified number of test points are selected. If the slack threshold is properly chosen, zero performance degradation can be achieved.

The testability computationU is performed at the primitive-gate level (AND, OR, NAND, NOR, NOT, etc.) while the timing analysis is done at the cell level. Therefore, we should restrict the test points to be only at the boundaries of standard cells but not inside the cells. If we allow insertion of test points inside the cells, re-mapping the logic would be required and the timing derived during the test point selection process could be inaccurate. An added control circuitry is mapped into a two-input AND or OR gate. An added observation point is assumed to add an extra load which is equivalent to the load of a flip-flop at the net and thus will also cause some extra delay.

2.1 Testability Computation

In this section, we summarize the well-known probabilistic random pattern testability measure Controllability/Observability Procedure (COP) [9], the Cost Reduction Factors (CRF) [4] used to estimate the improvement of testability due to insertion of a test point, and how test points are sequentially selected to achieve a high fault coverage with reasonable computation cost.

As detailed in [9], COP provides estimates of 1-controllability C_s and observability O_s of every node s in a combinational circuit. C_s represents the probability that node s has a logic value 1; O_s represents the probability that a logic value at s can be observed in at least one of the primary outputs. The 1-controllability of circuit nodes can be evaluated, in linear time, from primary inputs toward primary outputs level by level. After that, the observability can be evaluated from opposite direction again level by level. Figure 4 shows how 1-controllability and observability of an AND gate, an OR gate, a NOT gate, and a fanout stem are calculated.

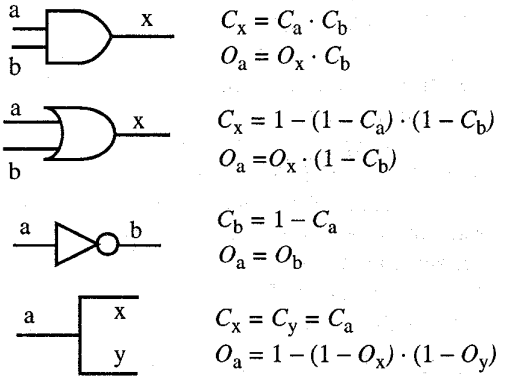

Figure 4: Examples of calculating controllabilities and observabilities

Controllabilities and observabilities by themselves are not sufficient to guide the selection of test points because they only estimate the local testability rather than the global testability impact when a test point is inserted. A good cost-function U used in [12] is defined as

$$U = \frac{1}{|F|}\left(\sum_{i \in F} \frac{1}{pd_i}\right) \qquad \text{(EQ 1)}$$

where F represents the fault set and $|F|$ is the cardinality of F, and

$pd_i = C_s \cdot O_s$, if i represents the stuck-at-0 fault at s, and
$pd_i = (1\text{-}C_s) \cdot O_s$, if i represents the stuck-at-1 fault at s.

While Equation 1 was derived in [12] from the viewpoint of the fault simulation cost, it can also be viewed as the average expected test length of all the faults in the circuit under random pattern tests because for any fault i, $1/pd_i$ is the expected test length of the fault. As will be explained later, to better use the concept of cost function U, two useful gradient values of U are also defined in [12]:

$$G_{C_S} = \frac{dU}{dC_s} \quad \text{(EQ 2)}$$

$$G_{O_S} = \frac{dU}{dO_s} \quad \text{(EQ 3)}$$

These two are called controllability gradient and observability gradient respectively. The gradients of all nodes can be computed in linear time using the algorithm in [12]. Both gradient values are still not very good indicators of the global testability impact for inserting a test point because they represent the change rate of U to C_s and O_s respectively due to an infinitely small change of C_s and O_s, whereas the insertion of a test point always cause the O_s or C_s to change drastically. Therefore, the Cost Reduction Factor (CRF) was introduced in [4] to approximate the Actual Cost Reduction (ACR), which is the reduction of U due to insertion of a test point. The CRFs for inserting an OR gate at s (CRF_s^{OR}), an AND gate at s (CRF_s^{AND}) and an observation point (CRF_s^{OBS}) are defined below:

$$CRF_s^{OR} = \frac{C_s - 1}{C_s + 1} \cdot C_s \cdot \frac{dU}{dC_s} - \frac{2}{C_s \cdot (1 - C_s) \cdot O_s} \quad \text{(EQ 4)}$$

$$CRF_s^{AND} = \frac{1 - C_s}{2 - C_s} \cdot C_s \cdot \frac{dU}{dC_s} - \frac{2}{C_s \cdot (1 - C_s) \cdot O_s} \quad \text{(EQ 5)}$$

$$CRF_s^{OBS} = (O_s - 1) \cdot O_s \cdot \frac{dU}{dO_s} \quad ,$$

if s represents a fanout stem in the original circuit. (EQ 6)

$$CRF_s^{OBS} = (O_s - 1) \cdot O_s \cdot \frac{dU}{dO_s} - \left(\frac{1}{C_s} + \frac{1}{1 - C_s}\right) \quad ,$$

otherwise. (EQ 7)

Figure 5 shows the test point selection algorithm which is modified from the test point selection algorithm proposed in [4] by considering the performance impact of inserting test points. We use a timing analyzer to compute the slack of each signal. As pointed out in Section 2, the candidate test points can only be at the cell boundaries but not inside the cells. An added control point will be mapped into a 2-input AND or OR gate.

2.2 Implementation and Experimental Results

We have implemented the algorithm in C. The implementation has been closely integrated with a timing analyzer for timing related operations. We have tried several MCNC

```
Timing_Driven_Test_Point_Selection {

 while (fault_coverage < desired_fc) and (#_of_test_point <
  maximum_number) {
    Compute slacks for all the nodes in the circuit;
    For each node s
        Compute the controllability Cs and observability Os;
        Compute the gradients GCs and GOs;
        Compute the CRFs^OBS;
        Compute the CRFs^AND or CRFs^OR, depending on the
          sign of the gradients GCs and GOs;
    Determine the max. value of CRF, called CRFmax, among all
     the CRFs computed;
    Determine the set of candidate test points NT, where
     NT= { i | CRFi^and/or/obs > threshold · CRFmax and Slacki >
     Slackthreshold};
    For each i in NT, insert the test point, then compute the Ui;
    Select test point j if Uj < Ui, for any i belong to NT and i ≠ j;
    Insert the test point j;
  } /* while */
}
```

Figure 5: Timing-driven test point insertion algorithm

benchmark circuits [9]. The circuits were first optimized for performance using Berkeley SIS [10] synthesis system and mapped into AT&T 0.9 micron CMOS cell library. Table 1 lists the results of test point insertion with and without the timing-driven option. The performance degradation using the algorithm without the timing-driven option is 2.7%, 33.4% and 12.8% respectively. In contrast, zero performance degradation with comparable fault coverages can be achieved by the timing-driven technique.

Table 1: Results of full-scan BIST

Ckt	Delay of the longest path			fault covg. (32k random patterns)			# of test pt.
	orig. ckt. (ns)	after TPI (ns)		orig. ckt (%)	after TPI (%)		
		non-TD	TD		non-TD	TD	
C2670	40.7	41.8	40.7	83.2	98.3	98.3	10
C7552	62.8	83.8	62.8	85.1	98.0	98.1	20
apex6	14.1	15.9	14.1	97.6	100.0	99.0	5

It is quite possible that the signals on critical paths may have a testability problem. Critical nets may be topologically either distant from primary inputs or from primary outputs or both and thus tend to have a lower testability than other nets. However, it is not really necessary to add test points directly at critical nets. Adding control points at the transitive fanins of a critical net may also help to boost the testabilities at the critical nets. Similarly, carefully selecting an observation point at a non-critical net will also help to improve the observability at the critical nets.

3. Timing-Driven Test Point Insertion for Partial-Scan BIST

3.1 The Basic Scheme

The basic idea of PSBIST is illustrated in Figure 6 and will be described in this subsection. Consider a directed graph $G=(V, E)$ of a synchronous sequential circuit, where a vertex v_i represents a flip-flop i, and a directed edge from v_i to v_j implies that there exists a combinational path from flip-flop i to flip-flop j.

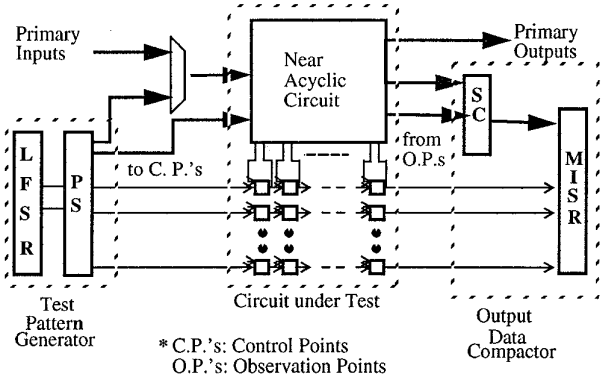

Figure 6: Structure of PSBIST

Definition 1: A Near Acyclic Circuit (NAC) is a synchronous sequential circuit whose corresponding directed graph does not contain any cycle whose length is greater than one. In a NAC, those flip-flops whose corresponding vertices involve in cycles of length one are called self-loop flip-flops. Others are called non-self-loop flip-flops.

Compared to full-scan based BIST structure shown in Figure 1, there are three major differences: First, only some but not all of flip-flops are scanned. Scan flip-flops are selected using loop-cutting algorithm [11], which renders the remaining part an NAC. Second, all the remaining self-loop flip-flops (after the selection of scan flip-flops in the first step) are replaced with equivalent initializable flip-flops, called INIT flip-flops, to guarantee the initializability of the circuit for the BIST application (All the INIT flip-flops can be set to a known state in one clock cycle by setting the preset/reset lines to active values and clock these flip-flops once). In addition, two control lines MODE_SW and INIT_SW are used to set the mode of the scanned flip-flops and the mode of INIT flip-flops. Also, the scanned flip-flops and the non-scanned flip-flops are driven by scan clock, SCAN_CLK, and system clock, SYS_CLK, respectively. SCAN_CLK is also used to drive both TPG and ODC during the BIST operation. The above description, however, is only a simplistic view of control and clock signals. Actual implementation would require a careful design to properly generate both clock and control signals avoiding skews or signal glitches.

The BIST application procedure, which is slightly different from that of a full-scan based BIST because some of the flip-flops are not scanned, is as follows.

Procedure 1:

1. Initialize the machine to a known state.
2. Set the MODE_SW to the value of the scan mode.
3. Clock the SCAN_CLK k times, where k is the length of the longest scan chain. Each clocking action scans one bit of a random pattern into the input of each scan chain and, at the same time, compacts one bit from the output of each scan chain as well as all the primary outputs into MISR. The state of the NAC remains the same during this period.
4. Set the MODE_SW to the value of the normal mode.
5. Clock both SYS_CLK and SCAN_CLK, and thereby latching the data into both scanned and non-scanned flip-flops.
6. Repeat steps 2 to 5 for a predetermined number of times.

3.2 Testability Computation

We extended the COP and CRF computation for NACs that contain self-loops. We decompose the NAC into blocks, similar to that in [5]. A block is defined as a set of logic gates and non-self-loop flip-flops feeding a sell- loop flip-flop or a primary output. Furthermore, these blocks are levelized according to the following definition of "Macro-Level". Before defining Macro-level, we have to define the directed graph G'. Graph G' is derived from the graph G, defined in the previous section, by removing all the edges that originate from and end at the same nodes and by attaching primary input nodes and primary output nodes to the graph.

Definition 3: The **Macro-Level** of a self-loop flip-flop or a primary output f in an NAC, denoted by ML(f), is the "node level" of the corresponding node in G', where the node level of a node f is defined by: ML(f) = 0 if f is a primary input. Otherwise, ML(f) =1 + max(ML(k)), for all nodes k's that feed node f directly.

An example of a decomposed circuit is shown in Figure 7. The sequential graph is used to define the block levels. The controllability and observability of each signal in a block is calculated using COP and the formulae in Section 2.1. Controllabilities are computed by traversing blocks from primary inputs and going toward primary outputs, block by block. On the contrary, observabilities are computed by

traversing the circuit from primary outputs toward primary inputs. There is a feedback line for each block.

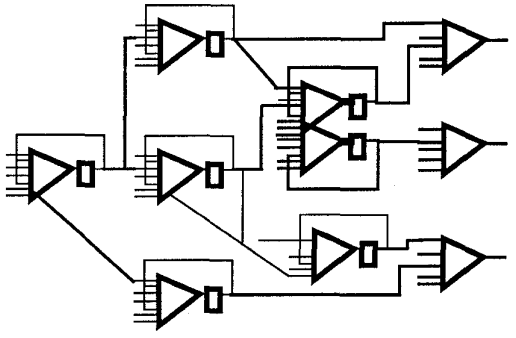

Figure 7: An example NAC after decomposition

Due to the existence of a feedback line for each self-loop block as shown in Figure 8, the methods for computing the various testability measures described in Section 2.1 cannot be directly used. In the following, we describe the extensions of those methods for an NAC.

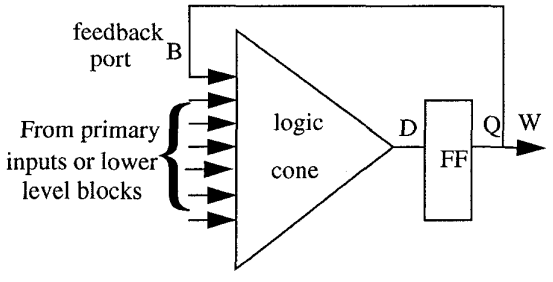

Figure 8: A self-loop flip-flop block

Computation of controllabilities. In computing the controllabilities for signals in a self-loop block as shown in Figure 8, we use a symbolic technique to avoid computing using excessive iteration. Before starting the controllability computation, the controllabilities at the inputs of the target block (could be either primary inputs or outputs of lower macro-level self-loop blocks) must have been computed. A symbolic variable, say x, is assigned to the feedback port B. COP is then used to compute the controllabilities of internal signals. The controllabilities of the internal signals which are in the fanout cone of the feedback port B will be a polynomial of x. According to [13], any high order exponents of x should be suppressed to eliminate the inaccuracy caused by the statistical dependence among the fanout branches from x. For example, consider the case shown in Figure 9. The controllabilities at the inputs of the AND gate are $C_1=ax+b$ and $C_2=cx+d$ respectively. According to COP, the output controllability $C_0=C_1 \bullet C_2=acx^2+(ad+bc)x+bd$. According to Parker & McCluskey's theorem [13], the term of x^2 should be simplified as x to correct the inaccuracy caused by the dependency between these two input signals originating from the feedback port B. This simplification not only increases the accuracy of the controllability measures, but also simplifies the expression of the controllability function of any internal signal as a *linear function* of x. We also assume that a flip-flop is modelled as a straight line with a data input and a data output and the controllabilities of the data input and the data output are identical. Therefore, if the controllability at the input of the self-loop flip-flop D is $ax+b$ where a and b are some constants, we can easily derive the controllability at B and in turn those at internal signals in the fanout cone of B by solving the equation $x=ax+b$.

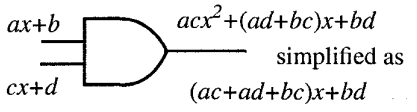

Figure 9: Computing the controllabilities of signals within a self-loop block

Computation of observabilities. To compute the observabilities of signals within a self-loop block, we assign the observability of the feedback port B, O_B, a variable x (See Figure 8). Since the observability at W should have been computed, we can express the observability at Q (or D) as a linear function of x ($O_Q = 1 - (1-O_W)(1-O_B) = ax + b$). Since the controllabilities of all signals should have been computed before the computation of observabilities, according to COP, the observabilities of the internal signals can be expressed as a polynomial of x. During the computation, when a fanout stem is reached as shown in Figure 10, the observability at the fanout stem with two fanout branches may become a quadratic function of x if both a and c are non-zero numbers. In general, the observability of an internal signal of a block will be a polynomial of x. Solving the equation of $x=f(x)$ at the feedback port of the block where $f(x)$ is a polynomial of x is a non-trivial task. To maintain the observability of every signal in the block to be a *linear* function of x, we suppress the second order exponents as shown in Figure 10. Note that the second order term arises from the reconvergence of the two fanout branches. When both coefficients a and c in Figure 10 are non-zero, these two fanout branches must converge and the value derived by COP is not exact. It is our conjecture that suppressing the high-order exponents will result in more accurate observability values. If we suppress the high order exponents at the fanout stem during the computation, the observability function of any signal in the block will be a linear function of x or a constant and is, in general, in the form of $ax+b$. We can prove that, for every signal, (1) both a and b are between 0 and 1, (2) $a+b$ is also between 0 and 1 and, therefore, (3) $ax+b$ is between 0 and 1 for any value of x between 0 and 1. Furthermore, in

Figure 10, because the coefficient of the quadratic term, $a'=(-ac)$, is a non-positive number and x is a number between 0 and 1, O'_r must be smaller than or equal to O_r. Since COP tends to produce an observability value higher than the actual value at the fanout stem, the suppression technique should produce a more accurate observability value.

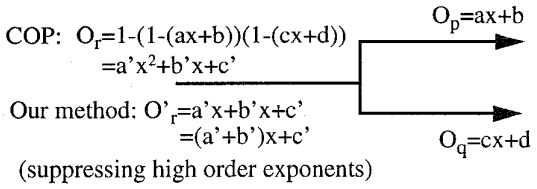

Figure 10: Computing the observability at a fanout stem

Computation of controllability gradients and observability gradients. The calculation of observability gradients, dU/dO, is performed block by block from lower macro-level blocks toward higher macro-level blocks. Computation begins by initializing dU/dO values at primary input nodes based on the node's controllability and observability values. Before the computation of a block, the dU/dO values at the inputs of the block except the feedback port have been computed.The dU/dO values for all other nodes are derived by recursively applying the chain rule for all gates in the forward path [12]. The dU/dO value at the primary input i is [12]:

$$\frac{dU}{dO_i} = \frac{-C_i}{Pd_{i-0}^2} + \frac{(C_i - 1)}{Pd_{i-1}^2} \qquad \text{(EQ 8)}$$

where C_i is the controllability of primary input i and Pd_{i-0} (Pd_{i-1}) is the detection probability of the stuck-at-0 (stuck-at-1) fault at signal i. The chain rule propagates derivatives from gate inputs to gate outputs using the formula [12]:

$$\frac{dU}{dO_k} = \frac{-C_k}{Pd_{k-0}^2} + \frac{(C_k - 1)}{Pd_{k-1}^2} + \sum_{i}^{inputs} \frac{dU}{dO_i}\frac{dO_i}{dO_k} \qquad \text{(EQ 9)}$$

where the first two terms are due to the change in the output node k itself, and the summation is due to all gate inputs. The quantity dO_i/dO_k depends on the gate type, and represents a transfer function that dictates how a gate's input observability will change, given a change in output observability. The detail of the chain rule is described in [12].

The dU/dO value at the feedback port is assigned a symbolic variable x. Since the first two terms of Equation 9 are constants and the dO_i/dO_k values are constants because the controllabilities and observabilities are known, the dU/dO_k value for any signal k in the block will be a linear function of x. If the dU/dO_k value at the input of the self-loop flip-flop D is $ax+b$ for some constants a and b, the dU/dO_k value at the feedback port B and in turn those at internal signals in the fanout cone of B can be derived by simply solving the equation $x=ax+b$.

Once the observability gradients for all signals are derived, we can start the calculation of the controllability gradients. The controllability gradients are computed block by block from the higher macro-level blocks toward the lower macro-level blocks. For each block, the controllability gradient, dU/dC, of the feedback port B, is assigned as a symbolic variable x. A chain rule formula for controllability gradient given in [12] is the used to compute backwards the controllability gradients at internal signals of the block. Again, the dU/dC values for all signals in the block can be expressed as a linear function of x. Therefore, the controllability gradients can be computed without iteration for an NAC.

Once all the controllabilities, observabilities and controllability and observability gradients are derived, we can use Equations 4-7 to calculate the cost reduction factors.

3.3 Implementation & Experimental Results

We have implemented a prototype program for selecting the test points for NACs. Again the program has been fully integrated with a timing analyzer for timing related operations. We have conducted experiments for several ISCAS89 sequential benchmark circuits [14] and obtained encouraging results as shown in Table 2. Because circuit s3330 has a large number of redundant faults, we ran a sequential redundancy removal program [15] first. The resulting circuit is named s3330sir. For each benchmark circuit, we first use the cycle-breaking program to select a set of scan flip-flops to convert the circuit into an NAC. The second column of Table 2 shows the total number of flip-flops in the benchmark circuit and the number of flip-flops selected for scan. Each circuit was first optimized for area and timing by the Berkeley synthesis system SIS [10] and was mapped into AT&T 0.9 micron CMOS cell library. The longest propagation delay of the circuit is shown in the third column. Each row in Table 2 shows the results of test point selection for a specified number of test points as given in the last column. We ran the program twice: one without timing driven option and the other with timing driven option. The propagation delays of the circuits after inserting the selected test points are given in Columns 4 and 5. For each of the original circuits and the circuits with test points, we ran fault simulation with 32k random patterns. Except s3330sir, the test-point-inserted circuits generated using timing driven option achieves a similar fault coverage as the ones generated without using the timing driven option. However, the former incurs zero performance degradation while the latter incurs some performance degradation

Circuit s3330sir needs more test points for the timing-driven option to achieve a comparable fault coverage. Figure 11 shows the curves of the fault coverage vs. the number of test points for both timing-driven and non-timing-driven cases for s3330sir. Figure 12 shows an example to illustrate a situation that more test points may be needed to avoid the selection of test points on critical paths. Suppose the signal controllability at signal *s* must be increased to improve the circuit testability as indicated by an up arrow. If the signals p, q, r, t and s are all in critical paths, we can still increase the controllability at s by decreasing the controllabilities at *x* and *y* (indicated by down arrows) and increasing the controllability at *z* that are not in critical paths.

Table 2: Results of partial-scan BIST

ckt	#.of total ff/ scan ff	Delay of the longest path (ns)			fault covg (32k random patterns) (%)			# of test pt.
		orig. ckt	after TPI non-TD	after TPI TD	orig. ckt	after TPI non-TD	after TPI TD	
s1423	74/22	20.7	21.2	20.7	77.6	97.0	96.9	10
s1423	74/22	20.7	21.5	20.7	77.6	98.5	98.6	20
s1423	74/22	20.7	21.5	20.7	77.6	99.0	98.8	25
s3330sir	127/8	15.4	16.3	15.4	74.2	93.8	90.6	15
s3330sir	127/8	15.4	16.3	15.4	74.2	97.1	95.0	25
s3330sir	127/8	15.4	16.3	15.4	74.2		97.1	35
s3384	183/8	21.4	21.8	21.4	92.6	98.6	98.6	5
s3384	183/8	21.4	22.1	21.4	92.6	98.7	98.7	15
s5378	179/30	12.4	13.3	12.4	90.7	97.2	97.3	20
s5378	179/30	12.4	13.3	12.4	90.7	98.1	98.3	30
s5378	179/30	12.4	13.3	12.4	90.7	97.2	97.3	35
s15850.1	394/23	22.3	22.3	22.3	93.9	96.2	96.2	10
s15850.1	394/23	22.3	23.6	22.3	93.9	98.0	98.0	30
s15850.1	394/23	22.3	23.6	22.3	93.9	98.8	98.8	40

In Table 3, we further compare the fault coverages and the number of test points with the results reported in [5]. In Columns 2-4 we list the number of scan flip-flops and the number of test points for designs achieving similar fault coverages. The comparison between the non-timing-driven results and the results of [5] indicates that for most cases our program selects fewer test points to achieve a comparable fault coverage. The technique given in [5] does not consider delay penalty of inserting test points and, therefore, some performance degradation is expected. We like to point out that the networks we used for experiments have been optimized by SIS. Therefore, the logic is slightly different from that used by [5] and, thus, the comparison made in Table 3 may not be accurate.

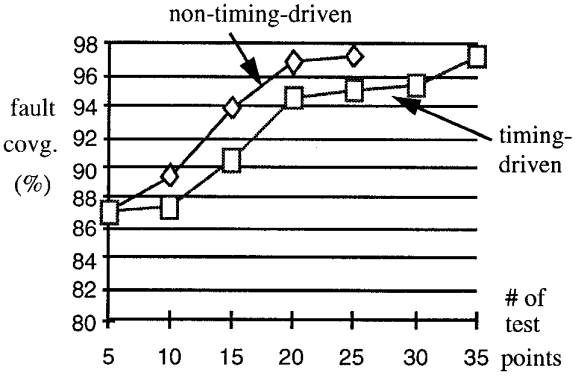

Figure 11: Fault coverages vs. number of test points for s3330sir

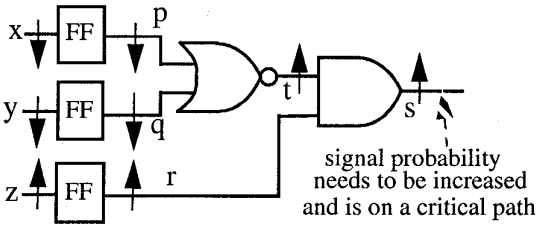

Figure 12: More test points may be needed to avoid selecting test points in critical paths.

Table 3: Comparison with results given in [5]

ckt	# of scan FFs + # of test points			fault coverage (32K random patterns)		
	results of [5]	our algorithm non-TD	our algorithm TD	results of [5]	our algorithm non-TD	our algorithm TD
s1423	51	47	47	98.9	99.0	98.8
s3330sir	28	28	28	95.7	96.9	93.5
s3330sir	28	33	33	95.7	97.1	95.1
s3384	42	23	23	99.6	98.7	98.7
s5378	70	60	60	98.3	98.1	98.3
s5378	70	65	65	98.3	98.3	98.5
s15850.1	128	63	63	98.9	98.8	98.8

Table 4 shows the run time of our program on SUN Sparc-Station 5. For all circuits, 10 test points are selected for both non-timing-driven and timing-driven cases. Even though the slacks for all signals need to be computed for the timing-driven case, the run time is not necessarily longer than that of the non-timing-driven case. This is because the number of test point candidates for simulation for the former is smaller than that for the latter.

Table 4: CPU time on SUN SparcStation 5

Circuit	non-timing driven	timing driven
s1423	190 sec.	182 sec.
s3330sir	276 sec.	245 sec.
s3384	820 sec.	784 sec.
s5378	456 sec.	385 sec.
s15850.1	61 min.	58 min.

4. Conclusion

In this paper, we proposed a timing-driven test point insertion algorithm for scan-based BIST schemes. The test points are selected to maximize the overall improvement of the random pattern testability and minimize the performance degradation. The testability improvement factors are computed based on the gradient methods given in [4][12]. We generalize their methods for partial-scan circuits that contain feedbacks. The testability measures are efficiently computed without iteration by a symbolic technique. Our algorithm selects the test points which result in the greatest testability improvement without violating the timing constraints. Our experimental results indicate that: (1) test point insertion without taking timing into account usually results in some performance degradation and (2) zero performance degradation and a high fault coverage of random patterns can be achieved by our timing-driven test point insertion algorithms. For some circuits, it may be necessary to select more test points in order to achieve a high fault coverage without inserting test points at critical nets. The proposed technique combined with the timing-driven partial-scan technique [6] could potentially offer a zero-performance-penalty BIST solution to timing critical sequential circuits.

Reference.

[1] P. Bardell and W. McAnney, "Self-Testing of Multiple Chip Modules," *Proc. Int'l. Test Conf.*, Nov. 1982, pp. 200-204.

[2] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," *Proc. of 11th IEEE VLSI Test Symp.*, April 1993, pp. 4-9.

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

[4] B. Seiss, P. Trouborst and M. Schulz, "Test Points Insertion for Scan-Based BIST", *Proc. of 2nd European Test Conf.*, April 1991, pp. 253-262.

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

[6] J. Jou and K.T. Cheng, "Timing-Driving Partial Scan for High Speed Circuits," *Proc. Int'l. Conf. on Computer Aided Design*, Nov. 1991, pp. 404-407.

[7] P. H. Bardell, "Design Considerations for Parallel Pseudo-Random Pattern Generators," *J. of Electronic Testing: Theory and Applications,* Vol. 1, pp.73-87, Feb. 1990.

[8] Y. Zorian and A. Ivanov, "Programmable Space Compaction for BIST," *Proc. 23rd Fault Tolerant Computing Symp.,* June 1993, pp. 340-349.

[9] F. Brglez, "On Testability of Combinational networks," in *Proc. 1984 IEEE Int. Symp. Circuits and Systems,* May 1984, pp. 221-225.

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

[11] K.T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedbacks," *IEEE Trans. on Comput.*, Vol. 39, No. 4, April 1990, pp. 544-548l.

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

[13] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, June 1975, pp. 668-670.

[14] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Int'l. Symposium on Circuits and Systems*, May 1989, pp. 1929-1934.

[15] K. T. Cheng, "Redundancy Removal for Sequential Circuits Without Reset States," *IEEE Trans. Computer-Aided Design*, Vol. 12, Jan. 1993, pp. 13-24.