Improved Algorithms for Constructive Multi-Phase Test Point Insertion for Scan Based BIST

Nadir Z. Basturkmen * Sudhakar M. Reddy *
Dept. of Electrical and Computer Engineering
University of Iowa
Iowa City, IA 52242

Janusz Rajski
Mentor Graphics Corporation
8005 SW Boeckman Rd.
Wilsonville, OR 97070

Abstract

New test point selection algorithms to improve test point insertion quality and performance of multi-phase test point insertion scheme, while reducing the memory requirement of the analyses are proposed. A new memory efficient probabilistic fault simulation method, which also handles the reconvergences to a limited extent for increased accuracy, is introduced. Syngistic control point insertion is targeted for higher test point insertion quality. Experiments conducted on various large industrial circuits demonstrate the effectiveness of the new algorithms.

1 Introduction

Steady increase in the complexity of integrated circuits requires a significant increase in the testing effort. Automatic Test Pattern Generation (ATPG) process consumes considerable amount of computing resources. Resulting large test data volumes require larger expensive Automatic Test Equipment (ATE) memory and longer test application times. Furthermore, limited increase in pin counts limits the increase in test application bandwidth, worsening the test application problems.

Built in Self Test (BIST) with pseudo-random patterns is an attractive alternative to overcome these difficulties. It offers low hardware overhead and simple test pattern generation and test application. However it suffers from reduced test coverage due to existence of random pattern resistant faults.

One solution to increase the pseudo-random pattern testability of the circuits is to insert test points to enhance controllability and observability of internal nodes. The first systematic test point insertion method was proposed in [5]. In that work, simulations are performed to obtain fault propagation profiles and correlations among signals. Signal correlations are eliminated by analyzing reconvergences and breaking them by the insertion of test points to facilitate propagation of blocked faults. Similarly, [6] uses fault simulation to identify gates that block fault propagation and inserts test points to allow propagation. Path tracing method described in [11] uses test points to ensure that there are sensitized fault activation and propagation paths for every fault for the given test set.

To reduce CPU time requirements of test point insertion process, approximate testability measures such as COP [3] is also used to analyze random pattern testability problems [7]. The work presented in [9] further develops COP based gradient method described in [8]. Cost reduction, i.e. reduction in average expected test length of all faults, as the result of selecting a signal as a test point is calculated for each signal in a circuit. A limited number of promising candidates are then selected for time consuming actual evaluation step and best one is inserted as the next test point. Methods described in [10, 12, 13] use similar analysis methods, but also consider the effects of test point insertion on area and timing.

To achieve a reasonable coverage, most practical circuits require a large number of control and observation points. During test application, typically ,all test points are activated in a single test session and control points are driven by independent, equi-probable signals. There are 2^K possible combinations of values for K equi-probable control points. Many of these combinations may have conflicting control point values. This may result in reduced coverage after a certain number of control points are inserted. The method described in [11] activates the control points only for the test patterns they are needed, through a pattern decoding logic. However, it requires the prior knowledge of test patterns, which may not always be the case.

Multi-Phase Test Point Insertion (MTPI) BIST scheme proposed in [4] utilizes a constructive divide and conquer approach. The test session is partitioned into multiple phases. Each phase adds to the coverage obtained so far, converging towards full coverage. Within each phase, test points (observation points in the first phase, control points in other phases) maximally adding to the fault coverage achieved so far are identified using a probabilistic fault simulation technique. Actual fault simulation is performed at the end of each phase, to eliminate newly detected faults as a result of test point insertion. Control points are held at a constant 0 or 1 throughout the phase they are selected. This approach has several advantages: First, it makes easier to predict the impact of multiple control points, since in each phase a new control point is selected in the presence of

*Research supported in part by NSF Grant No. CCR00-97905 and in part by SRC Grant No. 2001-TJ-949.

Proceedings of the 15th International Conference on VLSI Design (VLSID'02)
0-7695-1441-3/02 $17.00 © 2002 IEEE

control points selected so far in the phase. In this manner, a group of control points operating synergistically is enabled within each phase. Second, power dissipation during test mode is potentially reduced due to the usage of fixed values at control points. Finally, the sharing of the logic driving the control points is inherently determined. Determination of logic sharing among control points is not straightforward in general.

In this paper we propose methods to reduce compute time and memory requirements of procedures to place control and observation points. Since we implemented the proposed method into the existing infrastructure for the Multi-Phase Test Point Insertion method, for the sake of clarity we present our procedure in this context even though the same methods can be used with other procedures for inserting test points, the work presented in this paper preserves the constructive BIST methodology (MTPI) described in [4]. While achieving higher test quality and performance, memory requirements of the test point selection analysis, which can be quite significant for large industrial circuits, is considerably reduced compared to MTPI. A new, memory efficient probabilistic fault simulation methodology is developed for this purpose. To increase the accuracy of calculations without degrading the performance, reconvergent fan outs are handled to a limited extent during the calculations. Analysis to identify synergistic control points, i.e. control points that can be inserted together, is introduced to increase test quality. Experiments demonstrated that test quality is increased, while the use of computing resources are reduced significantly, especially for some large industrial designs.

2 Background and Motivation

Figure 1 illustrates the main components of the MTPI scheme [4]. The outputs of the phase decoder drive the control points and indicate the current phase in progress. The phase decoder block can be synthesized with the number of outputs ranging from $\lceil log_2 N \rceil$ to $N-1$, where N is the number of phases, depending on the routing and area constraints. In Figure 1 the entire test is divided into four phases: $Ph0$-$Ph3$. Non-overlapping phase enable signals $\Phi 0$-$\Phi 3$ and the output functions of the control points $C1$ and $C2$ in all phases are also shown.

An observation point is implemented at a node by means of an additional fan out from that node, which is connected to an output response analyzer. Observation points in MTPI scheme are enabled for the entire duration of the test.

The control point and observation point selection algorithms of MTPI rely on a probabilistic fault simulation technique. In this technique, first, signal probabilities of circuit nodes are determined by a logic simulation of random inputs. Then a fault f is initiated at its location, with a detection probability equal to its excitation probability. Forward propagation of the set of all undetected faults is then performed utilizing the equations derived in [4], in a levelized, event driven fashion. Propagated faults are kept in lists at every circuit node they reach, with the corresponding local probabilities of detection. Although probabilistic fault simulation accurately predicts the detection probabilities of faults at various circuit nodes, the memory requirement of this technique can be quite large. To reduce the memory requirement, MTPI does not include the faults in a list if the detection probability of the fault at that location is less than a certain threshold. However, the memory requirement for the probabilistic fault simulation can still be quite high, especially for very large circuits. Therefore, developing an algorithm with a lesser memory requirement, while keeping the same accuracy level would be useful.

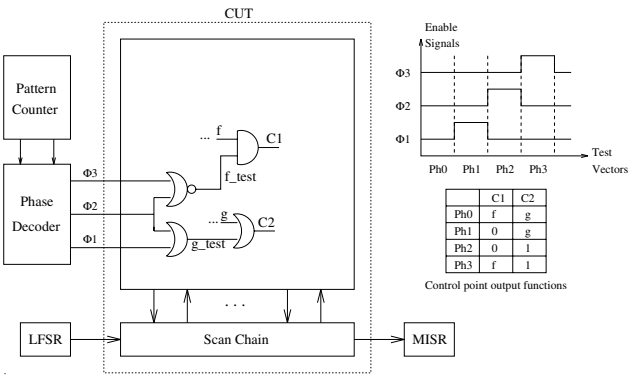

Figure 1. MTPI scheme

Another issue is related to the selection of control points. MTPI analysis determines some initial control point candidates whose signal probability values are extreme (w.r.t. a pre-specified threshold). Then the algorithm inserts those candidates one by one, performs a probabilistic fault simulation in the presence of already selected control points and the current candidate. Finally, the candidate that causes the propagation of most faults to observable points is selected. However, this method may not accurately identify synergistic control points. For example consider the situation shown in Figure 2. Assume that a and b are two points with very low signal probabilities and are allowed to be control points (there may be some restrictions on the circuit lines where a control point can be inserted). The low signal probability at the output of $AND1$ is blocking the propagation of faults in the input cone of the second input of $AND2$. If a is already selected as a control point (providing a high signal probability at a) previously, because it increases detection probability of some other faults, MTPI can recognize that insertion of another control point at b

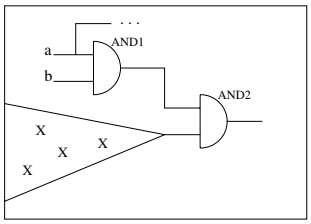

Figure 2. Potential pitfall for MTPI

un blocks the propagation path of the depicted faults. However, if both a and b are not control points at the moment, trying to insert control points at a and b separately will not result in propagation of those faults, and neither of them will be selected as a control point. Therefore, an algorithm that detects such cases, may increase the achieved fault coverage.

As previously mentioned, MTPI selects observation points only in the first phase, and control points in the remaining phases. However, insertion of observation points after the selection of control points can be essential, if propagation of some faults to already observable points is very difficult to achieve, even after control points are inserted. Selecting observation points after control point insertion could increase the fault coverage. However observation points will still be active in all phases, even if they are selected after the first phase. This may require re-simulation of inputs used during phases earlier than when additional observation points are selected.

3 A new probabilistic fault simulation method

As mentioned before, MTPI [4] initiates faults at fault sites and propagates them forward using fault lists. In this approach, fault list at the output of a gate is calculated from the lists at the inputs of that gate. A fault can propagate throughout its fanout cone. However, using a two level method of detection probability calculation becomes more memory efficient. In this technique, faults are first propagated until they reach the outputs of their corresponding fanout free regions (FFRs). Then for each FFR output that has a fault list, a representative of all of the faults in that list, called *surrogate fault*, is initiated. After this point, only the surrogate faults are propagated on behalf of the faults in their respective lists. The complete list of faults at a node can then be obtained by expanding these surrogate faults into original faults, while scaling the detection probabilities of original faults with the detection probabilities of their surrogate faults.

Surrogate fault propagation is equivalent to the calculation of conditional detection probabilities throughout the circuit, given that the output of the FFR is faulty. MTPI propagation procedure does not perform any conditional probability calculation. Hence a new propagation procedure needs to be developed.

3.1 Basic formulation

Assume, a section of a fault propagation path is shown in Figure 3. Assume propagation starts at A and probabilities of having $0, 1, D$ and $\overline{D}$ at A are given.

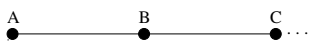

Figure 3. Part of a fault propagation path

Then the probabilities at point B can be found according to the following equation:

$$P_{B_x} = P_{B_x}^{A_0} P_{A_0} + P_{B_x}^{A_1} P_{A_1} + P_{B_x}^{A_D} P_{A_D} + P_{B_x}^{A_{\overline{D}}} P_{A_{\overline{D}}} \quad (1)$$

where $x \in \{0, 1, D, \overline{D}\}$. Here, for example, P_{B_0} denotes total probability of having a 0 at node B, and $P_{B_0}^{A_1}$ denotes probability of having a 0 at node B given the value at node A is 1. These equations can be put into matrix form:

$$\begin{bmatrix} P_{B_0} \\ P_{B_1} \\ P_{B_D} \\ P_{B_{\overline{D}}} \end{bmatrix} = \begin{bmatrix} P_{B_0}^{A_0} & P_{B_0}^{A_1} & P_{B_0}^{A_D} & P_{B_0}^{A_{\overline{D}}} \\ P_{B_1}^{A_0} & P_{B_1}^{A_1} & P_{B_1}^{A_D} & P_{B_1}^{A_{\overline{D}}} \\ P_{B_D}^{A_0} & P_{B_D}^{A_1} & P_{B_D}^{A_D} & P_{B_D}^{A_{\overline{D}}} \\ P_{B_{\overline{D}}}^{A_0} & P_{B_{\overline{D}}}^{A_1} & P_{B_{\overline{D}}}^{A_D} & P_{B_{\overline{D}}}^{A_{\overline{D}}} \end{bmatrix} \begin{bmatrix} P_{A_0} \\ P_{A_1} \\ P_{A_D} \\ P_{A_{\overline{D}}} \end{bmatrix} \quad (2)$$

or

$$\mathbf{P_B} = \mathbf{P_B^A}\,\mathbf{P_A} \quad (3)$$

Here, $\mathbf{P_B}$ and $\mathbf{P_A}$ are the probability vectors at A and B. Matrix $\mathbf{P_B^A}$ gives the conditional probabilities at B given the probabilities at A. If similarly, equations for node C is written in terms of the probabilities at node B, and Equation 1 is substituted in it,

$$\mathbf{P_C} = \mathbf{P_C^B}\,\mathbf{P_B^A}\,\mathbf{P_A} \quad (4)$$

is obtained. This formulation shows that, as we cascade sections of paths from A to B and B to C, conditional probability matrices are cascaded in the reverse order so that we can obtain the probabilities at node C given the probabilities at node A.

Exact computation of conditional signal probabilities in a circuit is proven to be $\#P$-hard [1]. However, a practical approximate surrogate fault propagation method can be developed based on the formulation derived above.

3.2 Propagation rules

Surrogate fault propagation starts with an initial identity probability matrix $\mathbf{I}$ at its fanout stem. Detection probability values of 1.0 through the main diagonal represents the initial conditional detection probabilities of $0, 1, D$ and $\overline{D}$, given the logic value at the stem is $0, 1, D$ and $\overline{D}$ respectively. Then, this matrix is propagated through the gates according to the propagation rules of those gates. During the propagation, off-path signals are assumed to be independent. Figure 4 shows a matrix originated at stem s and

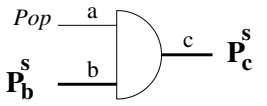

Figure 4. Matrix $\mathbf{P_b^s}$ at the input of an AND gate

reached to input b of an and gate. The signal probability of the off-path input a is P_{op}. Then the entries of $\mathbf{P_c^s}$ are given by:

$$\begin{aligned} P_{c_1}^{s_x} &= P_{b_1}^{s_x} \times P_{op} \\ P_{c_D}^{s_x} &= P_{b_D}^{s_x} \times P_{op} \\ P_{c_{\overline{D}}}^{s_x} &= P_{b_{\overline{D}}}^{s_x} \times P_{op} \\ P_{c_0}^{s_x} &= 1.0 - P_{c_1}^{s_x} - P_{c_D}^{s_x} - P_{c_{\overline{D}}}^{s_x} \end{aligned} \quad (5)$$

where x represents $0, 1, D$ or $\overline{D}$. The equations for other types of gates are derived similarly.

During the course of propagation, two probability matrices can meet at a reconvergence gate. In such a case, output probability matrix is obtained from the input matrices, by appropriately processing the corresponding columns considering the type of the reconvergence gate.

3.3 Handling of the reconvergences

As mentioned earlier, MTPI [4] assumes fault effects are independent to simplify fault propagation. When reconvergence of fault effects occur, this assumption causes inaccurate calculation of detection probabilities.
Example 1:

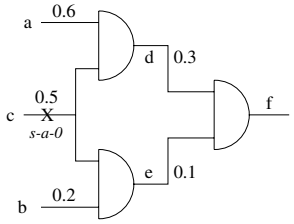

Figure 5. Fault on a reconvergent fanout stem

Assume the signals a, b and c are independent, and the signal probabilities on the lines are as shown in Figure 5. Suppose that there is a s-a-0 fault at stem c. If fault effects are propagated independently, the fault will be detected on c with a probability of 0.5, and on d and e with probabilities of 0.3 and 0.1 respectively. Then the detection probability on f will be:

$$\begin{aligned} P_D^f &= P_D^d \times SigPr^e + P_D^e \times SigPr^d + P_D^d \times P_D^e \\ &= 0.3 \times 0.1 + 0.1 \times 0.3 + 0.3 \times 0.1 = 0.09 \end{aligned}$$

However, given c is faulty, it is not possible to have a 1 value neither on d nor on e. Hence the first two terms in the above equation must be zero. The third term, $P_D^d \times P_D^e$ includes the fault probability twice, that is $(0.5 \times 0.6) \times (0.5 \times 0.2)$. If the above calculation is carried out conditional on the probability of having a fault on c the result will be $(1.0 \times 0.6) \times (1.0 \times 0.2) \times 0.5 = 0.06$. This can also be verified, by noting that the given circuit is in fact equivalent to a 3-input AND gate, with inputs a, b and c.

The same answer can also be obtained using the matrix formulation. Calculation starts with an identity matrix $\mathbf{I}$ at c, and it is propagated to d and e:

$$\mathbf{P_d^c} = \begin{bmatrix} 1.0 & 0.4 & 0.4 & 0.4 \\ 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0.6 \end{bmatrix} \quad \mathbf{P_e^c} = \begin{bmatrix} 1.0 & 0.2 & 0.2 & 0.2 \\ 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.8 \end{bmatrix}$$

$\mathbf{P_f^c}$ is obtained from $\mathbf{P_d^c}$ and $\mathbf{P_e^c}$. Then, given the input probability vector $\mathbf{P_c}$ ($Pr(0) = 0.5$ and $Pr(D) = 0.5$), $\mathbf{P_f}$ is calculated as:

$$\mathbf{P_f} = \mathbf{P_f^c}\,\mathbf{P_c} = \begin{bmatrix} 1.0 & 0.88 & 0.88 & 0.88 \\ 0 & 0.12 & 0 & 0 \\ 0 & 0 & 0.12 & 0 \\ 0 & 0 & 0 & 0.12 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \\ 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.94 \\ 0 \\ 0.06 \\ 0 \end{bmatrix}$$

□

3.4 Fault propagation through paths

Surrogate fault propagation can be carried out, by first partitioning the propagation paths into sections that can be cascaded, then calculating conditional probability matrices between the end points of those sections, and finally cascading the sections to form propagation paths, and hence cascading the corresponding matrices and finding resultant propagation probabilities over the paths. During the partitioning process the sections must be selected as large as possible. However, it must allow the handling of reconvergences too. To achieve these goals, propagation paths are divided into segments as defined below:
Definition 1: A *segment* is a section of a fault propagation path, which starts at a reconvergent fanout stem or at the output of a reconvergence gate, and ends at a reconvergent fanout stem or reconvergence gate.

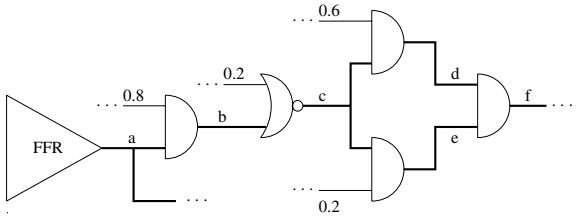

Figure 6. Surrogate fault propagation example

Following example demonstrates how segments are cascaded to obtain propagation probabilities.
Example 2: Consider the circuit in Figure 6. Suppose, a surrogate fault is initiated at stem a. Using Definition 1, we can identify three segments of fault propagation paths. Segment (a, b, c) starts and ends at reconvergent fanout stems a and c, segments (c, d) and (c, e) starts at reconvergent fanout stem c and ends at a reconvergence gate. Assume the off-path signals have signal probabilities as shown in the figure. The surrogate fault of stem a will be propagated to f. An identity matrix $\mathbf{I}$ is initiated at a, and it is propagated to b and c first. At this point it is necessary to handle conditional propagation from c to f first, which was done in Example 1. Then the probabilities at f given the probabilities at a are obtained as $\mathbf{P_f^a} = \mathbf{P_f^c}\,\mathbf{P_c^a}$, and the probability vector at f is obtained as $\mathbf{P_f} = \mathbf{P_f^a}\,\mathbf{P_a}$:

$$\mathbf{P_f} = \begin{bmatrix} 0.904 & 0.981 & 0.904 & 0.904 \\ 0.096 & 0.019 & 0.019 & 0.019 \\ 0 & 0 & 0 & 0.077 \\ 0 & 0 & 0.077 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \\ 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.904 \\ 0.058 \\ 0 \\ 0.038 \end{bmatrix}$$

□

As mentioned above, the propagation procedure requires knowledge of reconvergent fan out stems. This information is extracted and stored during a preprocessing step which extracts stem regions of the circuit. Stem region concept is explained in Section 3.5.

Previous examples demonstrated cases where matrices initiated at the same stem reconverge. When there are more than one reconvergent fanout stem, new identity matrices are initiated at each one of them and propagated forward.

Hence it will be necessary to keep track of the source of propagated matrices, by attaching appropriate labels indicating the source of the matrix. If two matrices initiated at different stems reconverge at a gate, they can not be immediately used for calculating gate output probability matrix. These calculations must be made w.r.t. the same source. This is done by appropriately cascading the paths, until two inputs has the same source.

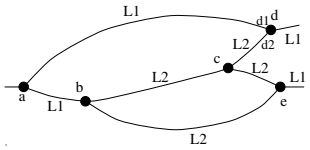

Figure 7. Propagation path labeling

Figure 7 shows fault propagation paths for the surrogate fault of stem a. Nodes a, b and c are fanout stems, nodes d and e are reconvergence gates. The matrix initiated at a is labeled as L_1. This label is attached to the matrices during propagation as long as no new identity matrix is created along the way. In the figure, label L_1 is carried along the segments (a, b) and (a, d). When propagation reaches stem b, a new identity matrix is created, hence the matrices propagated after this point are labeled as L_2. Note that, stem c is not reconvergent, hence no new matrix is initiated at that point and there is no need for a new label. When matrices reconverge at d, the output matrix of the reconvergence gate, $\mathbf{P}_{\mathbf{d}}^{\mathbf{b}}$, can not be calculated immediately. Because, probability matrix at d_1 input of the reconvergence gate is expressed w.r.t. a and matrix at d_2 input is expressed w.r.t. b. The probability matrix at d_2 must also be expressed w.r.t. a. This is done by cascading the segments (b, d_2) and (a, b) to obtain the matrix $\mathbf{P}_{\mathbf{d_2}}^{\mathbf{a}} = \mathbf{P}_{\mathbf{d_2}}^{\mathbf{b}}\,\mathbf{P}_{\mathbf{b}}^{\mathbf{a}}$. Now, $\mathbf{P}_{\mathbf{d}}^{\mathbf{a}}$ can be calculated using $\mathbf{P}_{\mathbf{d_1}}^{\mathbf{a}}$ and $\mathbf{P}_{\mathbf{d_2}}^{\mathbf{a}}$. This means that, output matrix of d is now expressed w.r.t. stem a, therefore after that point, propagated matrices are labeled as L_1.

The matrices reconverging at e are both labeled as L_2, hence output at e can directly be calculated from those two matrices. However, labeling can be returned to the point where reconvergence originated by calculating $\mathbf{P}_{\mathbf{e}}^{\mathbf{a}} = \mathbf{P}_{\mathbf{e}}^{\mathbf{b}}\,\mathbf{P}_{\mathbf{b}}^{\mathbf{a}}$. By doing so, labeling is kept as close to the original stem as possible.

During the course of propagation, matrices are never saved at a node unless it is a reconvergent fanout stem. It is necessary to save those matrices to be able to do segment cascade operations. On the other hand, as matrices are propagated forward, local probability of detection of the original surrogate fault is calculated and kept in surrogate fault lists at every node. As a matter of fact, there is no need to keep a fault list if the node is not observable or allowed to be a an observation point, because in the test point evaluation step, only such lists will be considered. This can help to reduce the fault list size, since flattened circuit descriptions are used during the analyses and many nodes are not allowed to be observation points.

The method described until now, captures correlation among the reconvergent signals correctly, if there is a single source of correlation, by calculating probabilities at the point of reconvergence conditional on the source. However, if reconverging signals are correlated by more than one source, this method can not calculate probabilities accurately.

3.5 Region of fault propagation

MTPI [4] propagates the faults throughout their fanout cones. The method presented here tries to capture, at least partially, the effects of reconvergences. Carrying out such calculation throughout the fanout cone of each surrogate fault can be costly. Limiting the calculation to a region smaller than the fanout cone of a surrogate fault will be useful. In [2] Maamari and Rajski define *stem region* of a fanout stem, which is at most as big as fanout cone of the stem. A deterministic fault simulation framework in which there is no need to explicitly simulate stem faults beyond their respective stem regions is also given in [2]. This concept can be adopted for a probabilistic fault simulation framework.

Informally, stem region of a stem can be described as follows: It is bounded by the lines called *exit lines*. An exit line is the immediate dominator of the stem for a subset of outputs of the circuit, that is, every path between the stem and any of the outputs in that subset goes through that particular exit line, and it is the closest such line to the stem itself. The subsets of outputs that can be reached from different exit lines are disjoint. This means that different exit lines never reconverge.

The deterministic fault simulation method described in [2] states that, a stem fault will be detected, if it is detected on an exit line, and the line is critical, i.e. a change in its logic value changes the logic value of a primary output. Therefore there is no need to explicitly simulate the fault any further. The detection is guaranteed in such a case, because exit lines never reconverge, and fault effects reaching different exit lines can not mask each other.

The same property of exit lines can be used to stop probabilistic fault simulation close to those points. In fact, after reaching an exit line, probabilistic simulation of a fault is continued until it reaches the output of the FFR it is currently in. There is no need to propagate that surrogate fault any further, because it can be represented by another surrogate fault which will be initiated at the stem it now reached. The new surrogate fault will now represent two kinds of faults: Surrogate fault(s) already propagated up to that point and local faults (if there are any) of its own FFR. This results in a multi-level surrogate representation. Hence, when fault lists are evaluated during observation point or control point selection, a surrogate fault must be expanded into the local faults it represents, as well as the local faults of the surrogate faults it represents. Note that, any previous level surrogate fault may also be representing some other surrogate faults. Therefore, this backward expansion should continue as long as there are surrogate faults to be expanded.

Since exit lines of a stem region do not reconverge, surrogate faults corresponding to FFR of those exit lines do

not reconverge either. Therefore, surrogate faults reconverging at some point in the circuit can not represent some common faults. This implies that it is enough to explicitly propagate a surrogate fault within its stem region to handle reconvergences, and implicitly propagate thereafter, using multi-level surrogate representation.

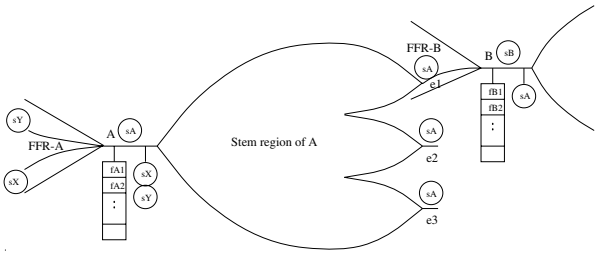

Figure 8. Multi-level surrogate fault representation

Figure 8 demonstrates multi-level surrogate representation. Local faults of FFR-A, $f_{A_1}, f_{A_2}, ...$ have been propagated to stem A, as well as some surrogate faults s_X and s_Y. A new surrogate fault s_A, which is initiated at stem A, represents all of those faults. Then s_A is propagated throughout its stem region. When it reaches an exit line, propagation continues until it also reaches to the output of the FFR of that exit line. In the figure above, this case is shown for the exit line e_1. After reaching e_1, propagation of s_A continued until it reaches output of FFR-B. When s_A reaches stem B, it is placed into the stem faults list of B and a new surrogate fault s_B is created at that point. Now s_B represents its own local faults $f_{B_1}, f_{B_2}, ...$, and the faults that s_A represents. When s_B is expanded during the evaluation process, it must be expanded into the local faults of B: $f_{B_1}, f_{B_2}...$, local faults of A: $f_{A_1}, f_{A_2}, ...$, as well as the local faults of stems X and Y (not shown in the figure): $f_{X_1}, f_{X_2}, ...$ and $f_{Y_1}, f_{Y_2}, ...$

4 Test point selection

Selection of observation points is is based on the probabilistic fault simulation method described in the previous section. The selection algorithm is essentially the same as the the one used in [4], except in the evaluation step, the surrogate faults must be expanded properly. Hence, it is not discussed here in detail.

Selection of control points however, depends on the use of COP testability measures and is discussed next.

4.1 Control point selection

Signals with extreme signal probabilities in the circuit are good candidates for control points, because they cause fault inactivation and block fault propagation paths. However, trying every such point as a control point during the analysis would be very time consuming. Hence, it is necessary to determine an initial set of control point candidates, which are likely to be effective, before the actual evaluation step.

4.1.1 Selection of control point candidates

Control point candidates are selected among failing signals, which are determined using the following definition:

Definition 2: Let LTh and HTh be two pre-specified threshold values for low and high signal probabilities respectively. Then, a signal with signal probability sp is said to be a *failing signal* if $sp > HTh$ or $sp < LTh$.

To determine control point candidates, fault propagation paths within a stem region are analyzed first, and failing signals that blocks fault propagation paths are determined. As a heuristic method, two paths from a stem to each of its exit lines are chosen: A path with a minimum number of blocking signals on it, because it is the easiest to unblock. Another one with a maximum number of blocking signals on it is also chosen. Targeting blocking signals on such a path can unblock some untargeted paths as well, because the origin of blockages on different paths can be the same. This point will be more clear in a moment. All such blocking signals within a stem region are marked. Blocking signals from exit lines through output of their corresponding FFRs are also marked. Next, propagation paths from fault locations to FFR outputs for individual faults are examined. The signals hindering fault propagation through FFR outputs are marked. Finally, failing signals that cause inactivation at fault sites are added to this list.

The potential candidates are now reduced to a subset of all failing signals. However, since the effect of a failing signal can propagate throughout circuit, it is possible that some of the signal probability failures at blocking signals have some common origins, called *source* of the failure. To find such sources, blocking signals are traced back in the circuit as long as the signal probability failure continues w.r.t. threshold values LTh and HTh. When all of the input signal probabilities of a gate are no longer failing, back trace is stopped and the output of the gate is considered as the source of the signal probability failure.

4.1.2 Evaluation of control point candidates

Source signals are the candidates for the insertion of control points. Effectiveness of each of these points as a control point is evaluated by calculating the detection probabilities of undetected faults before and after inserting them as control points.

First step of the evaluation process is to calculate initial fault detection probabilities, at the presence of already selected control points (if there are any), but without any candidate control point. Signal controllability values are obtained by logic simulation of random vectors. Observability values are then calculated from primary outputs through primary inputs using COP observability calculation rules.

To evaluate the effect of control point candidates, they are inserted into the circuit one candidate at a time, at the presence of already selected control points, if there are any. Appropriate constant values are injected at control points, and their effects are propagated throughout the circuit, with the method described in [4]. Then the new observability values within the fan in cone of the region effected by controllability changes are obtained. Both signal probability and observability calculations are not carried out beyond points where changes are less than a pre-specified delta threshold $DeltaTh$.

Once the new observability and controllability values are calculated, the change in detection probabilities of the faults in the effected region is calculated, and the number of new faults to be detected by the candidate is estimated. Formally speaking, let $C_{sel} = \{C_{sel_1}, C_{sel_2}, ..., C_{sel_k}\}$ denote the set of k control points selected so far, and $C_{can} = \{C_{can_1}, C_{can_2}, ..., C_{can_l}\}$ denote the set of l control point candidates. The i^{th} candidate $C_{can_i} \in C_{can}$ is evaluated by inserting the set $C_{ins_i} = C_{sel} \cup C_{can_i}$ into the circuit. Let Pd_{j0} be the detection probability of fault j when only selected control points are inserted, and Pd_{ji} be the detection probability of fault j when the set C_{ins_i} is inserted in to the circuit. Fault j is covered by candidate i if $Pd_{j0} < DTh$ and $Pd_{ji} > DTh$, where DTh is the detection threshold, at which point a fault is assumed to be detected.

To limit the runtime of the evaluation process described above, the number of control points inserted in to circuit is limited with a window size w, and at most last $w-1$ selected control points and the next control point candidate are inserted into circuit for analysis. This confines the control point analysis to a smaller region and reduces runtime.

The analysis based on the detection probabilities alone may not be enough, due to the existence of synergistic control point candidates. Consider the case in Figure 2. a and b are two signals with very low signal probabilities. Trying either a or b alone as a control point will not increase the detection probabilities of depicted faults considerably, because the signal probability on line c will not change significantly. However, this difficulty may be overcome by using logic implications rather than relying on probabilistic analysis. If $a = 0$ is implied with all other signals are initially unknown, it possible to observe that, propagation of depicted faults are blocked regardless of the value of b, hence a must be selected as a control point in order to propagate those faults.

Such blockage analysis is carried out in two steps like probabilistic analysis. First step is the forward implication of logic values, which is carried out in an event driven manner. The gates effected by the implications are marked during the implication process. Then, a backward tracing is done in the reverse direction, by determining the blocked inputs of the gates, due to implied logic values. This blockages are pushed backwards, to the gates connected to those blocked inputs. A gate with multiple fanouts is marked as blocked if all of its fanout branches are blocked. The backward trace stops either at primary inputs, or there are no more blockages to push backward. Meanwhile, number of undetected faults within the blocked region are also determined and saved for each candidate C_{can_i}.

The candidate that provides the maximum benefit is then selected as the next control point. Benefit provided by a candidate is given by the number of faults it covers (given by the probabilistic analysis) or by the number of faults it blocks (given by the implication analysis) whichever is larger. The control points are selected as long as the number of control points do not exceed the number of maximum control points for that phase, and the benefit provided by the control point meets a benefit-per-cost (BPC) criterion.

5 Results

The proposed algorithms for test point selection are implemented to run in conjunction with an MTPI [4] implementation. Experiments were then conducted on full scan versions of various industrial circuits, using MTPI and the proposed procedure.

The test point insertion is performed with a pre-specified number of control and observation points and fault coverages are obtained after application of a pre-specified number of pseudo-random vectors. Same patterns are used for both sets of experiments. All experiments are conducted using 4 phases. Easy to detect faults are eliminated in a *pre-phase* fault simulation, whose vectors are repeated in the first phase, to simplify the analysis. Detection probability threshold DTh is set to $4/D_{Ph_i}$, where D_{Ph_i} is the duration of phase Ph_i. Minimum BPC value is set to 4 for circuits A, B and D and it is set to 1 for circuits C and E. This is because MTPI was rejecting control point candidates in the runs for the latter circuits, even though maximum allowed number of control points is not reached yet. A value of 0.1 for the low signal probability threshold LTh is used in MTPI experiments. HTh is given by 1.0 - LTh. However, for circuit E, LTh is set to 0.01 to reduce the number of control point candidates and hence the run time of MTPI. For the proposed procedure, LTh value is set to 0.1 and the window size for control points injected into circuit together for evaluation is set to 4 for circuit A, and to 3 for other circuits.

The proposed procedure also inserts observation points in phases other than Ph_0. A pre-specified percentage of observation points are placed in Ph_0. For other phases, the number of observation points are determined by dividing the remaining number of observation points by the number of remaining phases. A limited number of control points are also selected in Ph_0 depending only on the blockage analysis. If a failing signal causes blockage of more than 1% of the remaining faults, it is inserted as a control point. The maximum number of such control points is also pre-specified. A final simulation of all test vectors is carried out to capture the effects of those control points and the observation points inserted after Ph_0.

Table 1 lists the number of test points and the final coverage results for both procedures. For the proposed procedure, percentage of observation points inserted in Ph_0 is listed in OP_0 column. These results show that the proposed procedure achieves better coverage than MTPI in all of the circuits except circuit D. At the same time, the number of control points are also improved for circuits B, C and E. Improvement is especially significant for the large circuit E. All of the available observation points are placed by both procedures. This is because the relative cost of an observation point is 1/4th of a control point. Also most of them are placed when there are many undetected faults as indicated by OP_0 column (for MTPI these values are all 100%).

Run times and memory usages of both procedures are also listed in Table 1. These results are obtained on a Sun Ultra Sparc 60 machine with 4GB of memory.

Run times of the proposed procedure shows significant

			MTPI				Proposed				
Circuit	#Gates	#Pat.	OP/CP	Cov.	Mem. (MB)	Time (sec.)	OP/CP	OP_0 (%)	Cov.	Mem. (MB)	Time (sec.)
A	31K	64K	16/15	96.09	6	46	16/15	50	96.23	16	56
B	203K	64K	200/78	98.64	73	1210	200/75	90	98.67	32	1400
C	550K	128K	400/37	93.92	196	42456	400/27	90	94.08	72	7488
D	893K	128K	750/174	94.50	203	7603	750/175	98	94.39	98	7328
E	2531K	512K	2500/702	87.32	1600	463885	2500/672	94	87.45	950	224321

Table 1. Test point insertion results, memory usages and run times

reduction compared to MTPI as the circuit sizes increase. For small circuits A and B MTPI runs faster, where as for large circuits C, D and E the proposed procedure is faster than MTPI. This is because the total run times for larger circuits are dominated by the control point selection step and the proposed procedure selects control points much faster than MTPI. For smaller circuits, the preprocessing time required to extract stem regions becomes significant. Also, the observation point selection in the proposed procedure requires more time than MTPI since they both use the same selection algorithm except that the new procedure uses surrogate fault representation. The backward expansion of surrogate fault lists is slower than directly using them. Nevertheless, these steps are not the dominant steps for large circuits and speed up ratios of 52% for circuit E and 82% for circuit C are significant considering the total run times for these circuits.

Similarly, memory requirement of the proposed procedure is significantly less than MTPI for large circuits. However, for small circuit A, MTPI uses less memory than the proposed procedure. This is because for small circuits, fault list sizes are not so significant, and total memory usage is dominated by other data structures. Since our implementation is done on top of an existing MTPI implementation and uses extra data structures, memory usage of the proposed procedure is more than MTPI when the fault list sizes are not large. For example, for circuit A, there are 2357 remaining faults after pre-phase fault simulation, whereas this number is about 953K for circuit E. Hence the proposed procedure provides a memory gain of 40%, when the fault list size is very large and consumes a significant portion of the memory.

6 Conclusion

In this work, new test point selection algorithms for MTPI [4] have been presented. The new algorithms were able to improve test coverage, without degrading or even improving the number of selected test points. The memory usage showed significant improvement for large test cases, which was a primary target of the new algorithms. The run times were also improved. Improvements were again significant for large test cases, that has very long run times. This shows that, use of surrogate fault propagation was effective in reducing the memory usage, without degrading the level of accuracy and performance. The control point selection algorithms were also effective, especially for reducing run times, since MTPI spent most of its time in control point selection step for the largest test cases.

Acknowledgement

We would like to thank Nagesh Tamarapalli and Mark Kassab from Mentor Graphics, for their help and support during the course of this work.

References

[1] P. Goel, "Test generation costs and projections", *Proc. 17th Des. Auto. Conf.*, pp. 77-84, 1980.

[2] F. Maamari, and J. Rajski, "A Method of Fault Simulation Based on Stem Regions", *IEEE Trans. on Comp. Aided Des.*, vol. 9 no. 2, pp. 212-220, 1990.

[3] F. Brglez, P. Pownall, and R. Hum, "Applications of Testability Analysis: From ATPG to Critical Delay Path Tracing", *Proc. of ITC*, pp. 705-712, 1984.

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

[5] A.J. Briers, and K.A.E. Totton, "Random Pattern Testability by Fast Fault Simulation", *Proc. of ITC*, pp. 274-281, 1986.

[6] V.S. Iyengar, and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs", *Proc. of ITC*, pp. 501-508, 1989.

[7] Y. Savaria et al., "Automatic Test point Insertion for Pseudo-Random Testing" *Proc. Int'l Symp. on Circ. and Systems* pp. 1960-1963, 1991.

[8] R. Lisanke et al., "Testability Driven Random Test-Pattern Generation", *IEEE Trans. on Comp. Aided Des.*, vol. CAD-6, No. 6, pp 1082-1087 Nov. 1987.

[9] B.H. Seiss et al., "Test point Insertion for Scan-Based BIST", *Proc. of Euro. Test Conf.*, pp. 253-262, 1991.

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

[11] N.A. Touba, and E.J. McCluskey, "Test Point Insertion Based on Path Tracing", *Proc. of 14th VLSI Test Symp.*, pp. 2-8, 1996.

[12] H.C. Tsai et al., "A Hybrid Algorithm for Test Point Selection for Scan-Based BIST", *Proc. of Des. Auto. Conf.*, pp. 478-483, 1997.

[13] M. Nakao et al., "Low Overhead Test Point Insertion for Scan-Based BIST" *Proc. of ITC*, pp. 348-357, 1999.