Random Pattern Testability Enhancement by Circuit Rewiring

Shih-Chieh Chang **Kwen-Yo Chen** **Wen-Ben Jone** *__Sunil R. Das__

Department of Computer Science and Information Engineering National Chung Cheng University, Chiayi, Taiwan 621, Republic of China.

*Department of Electrical Engineering University of Ottawa, Ottaws, Ontario K1N 6N5

Abstract

Generally, there exist random-pattern resistant faults that result in the poor fault coverage in Build-In Self-Testing (BIST) scheme. In this paper, we propose a method to enhance the random pattern testability by a circuit restructuring technique, called *circuit rewiring*. The basic idea of rewiring is to replace a wire by another wire with the circuit functionality remaining unchanged. For two types of rewiring, fanin rewiring and fanout rewiring, we first analyze the testability change for each type of wire replacement. Based on the analysis, an efficient algorithm is given to enhance circuit testability. For a poor observability node, we try to increase its observability by adding an additional fanout to the node and by removing an alternative wire whose source node has relatively good observability. The technique introduces no hardware overhead since each wire addition is followed immediately by another wire removal. Thus, it is basically cost-free when compared to other testability enhancement techniques.

1 Introduction

Testing a digital circuit is an experiment in which the circuit under test (CUT) is exercised by applying a sequence of input test patterns and then its output responses are analyzed to ascertain whether there exist faults in the circuit. *Built-In Self-Testing* (BIST) has been widely used in testing modern circuits or chips, and the basic idea is to have the circuit test itself [1] [2]. Basically, BIST of logic circuits can be divided into two major categories: *random* testing and *exhaustive* testing. By embedding a built-in device such as *linear feedback shift register* (LFSR) inside the CUT, we can have a sequence of random test patterns generated if the LFSR is initiated with a proper seed value. The defects which occur can be sensitized and detected if the number of random test patterns is large. Of course, there is no really random pattern generator as long as the machine is built by humans, since bit dependency occurs. However, the test vectors, called *pseudo-random* test patterns, generated still have many properties of random patterns [3]. The problem with pseudo-random testing lies in the difficulty of enhancing the random pattern testability for circuits containing many random-pattern resistant faults.

Many random-pattern resistance faults can be solved by applying *weighted* random patterns [4] [14] [19] or by inserting *test points* [13] [15] [16] [17] [18]. The basic idea of weighted random pattern testing is to test the logic with biased random patterns, i.e., the numbers of logic *1* and logic *0* applied to each input are not equal. This can be achieved by building a weighted random pattern generator from an LFSR with AND/OR gates. Substantial improvement can be obtained if the test patterns are properly weighted. Instead of changing the distribution of random patterns, another major approach is to modify the logic by inserting test points such that the detection probabilities of all random-pattern resistant faults can be increased [15]. Both approaches incur hardware overhead and performance degradation.

In this work, we propose a method which is able to enhance the random pattern testability of the CUT by a circuit restructuring technique called circuit rewiring. The rewiring technique for testability enhancement has the potential of no hardware overhead and no performance degradation. The basic idea of circuit rewiring is to find a wire to replace another one without changing the circuit function. Thus, it is beneficial to supersede a hard-to-test line using another easy-to-test line. Given a set of random test patterns, all undetected faults can be identified by fault simulation. The fault coverage accomplished by the set of random patterns can be further increased by circuit rewiring. In this experiment, we try to rewire the circuit such that the observabilities of undetected lines can be enhanced. Simulation results demonstrate that the circuit rewiring technique can successfully break the random pattern barrier for many benchmark circuits. Without rewiring, the fault coverage of a CUT might saturate and remain constant even with the application of additional test patterns. However, the fault coverage can be increased if the circuit lines are properly rewired. We have observed that circuit rewiring increases random pattern fault coverage in most cases.

The background of circuit rewiring is described in Section 2. Section 3 investigates the relationship between circuit rewiring and random pattern testability, and proposes a heuristic rewiring method. Simulation results are provided in Section 4, and Section 5 concludes the work.

2 Background

This paper applies the idea of rewiring to improve the testability of a circuit. There have been several studies [5] [6] [7] [10] [12] on how to logically exchange wire connections under the constraint that the circuit function remains unchanged. In this section, we will review the basic concept of rewiring which has the ability of modifying a circuit. In Section 3, an algorithm using circuit rewiring for testability enhancement will be presented.

By definition, wire A is an *alternative* wire of wire B if we can replace wire B by wire A. Consider the circuit shown in Fig. 1, wire g_1->g_4 can be replaced by wire g_5->g_9 while the circuit function remains unchanged, i.e., both circuits in Fig. 1 are functionally equivalent. In this case, we say that wire g_5->g_9 is an alternative wire of wire g_1->g_4. Additionally, the alternative wires are mutually alternative; i.e., if wire A is an

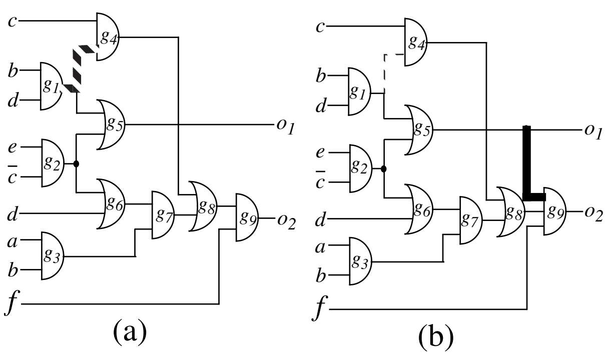

Fig. 1 : An example for single alternative wire replacement.

alternative wire of wire B, then wire B is also an alternative wire of wire A. Thus, we can also replace g_5->g_9 by g_1->g_4 in Fig. 1.

Let us further examine the same example. First, a *redundant* wire is a wire which can be added (removed) to (from) a circuit without changing the circuit function. It can be found that the nonexistent wire g_5->g_9 is redundant in Fig. 1(a) and can therefore be added to the circuit. On the other hand, g_1->g_4 is ir-redundant in Fig. 1(a). However, when wire g_5->g_9 is added to the circuit, wire g_1->g_4 becomes redundant and can thus be removed from the circuit (Fig. 1(b)). From this example, we can observe that an originally irredundant wire may become redundant after adding some redundant wire to a circuit. The way of redundancy addition and redundancy removal forms the basic idea for the circuit rewiring technique.

The procedure which finds an alternative wire for the target wire first searches a nonexistent wire, called the *candidate* connection that, once added, makes the target wire redundant. Then, the rewiring technique checks whether the candidate connection is redundant, i.e., it checks whether adding the non-existing wire preserves the circuit's functionality. It is only when the candidate connection is verified as redundant, that we can then add the candidate connection and thereby further remove the target wire. All redundancy checks can be done through reasoning of automatic test pattern generation (ATPG). More details for finding alternative wires can be found in [7].

3 Rewiring for testability

In the section, the effect of rewiring to circuit testability is first analyzed, and then a heuristic method for testability enhancement will be presented. In order to test a stuck-at fault, it is required to activate the fault and then propagate the fault effect to at least one primary output. By definition, the probability of a primary input vector to activate a fault *f* is called the *controllability* of *f,* while that to propagate the fault effect of *f* to a primary output is called the *observability* of *f.* Both controllability and observability of a fault adequately indicate the degree of difficulty to activate the fault and to observe the fault effect. Thus, the testability of a fault can be measured by these two values.

For observability, in general, if a node is close to a primary output, then the observability of the node is high. A primary output node is considered to be fully observable. Further, if a node has many fanouts, the node has good opportunity to be highly observable since the fault effect can propagate to primary outputs from one of its fanouts. (We neglect the fault-masking problem.) In a circuit, some nodes may have very good observability (such as a primary output node or a node with many fanouts) while some nodes may have very poor observability. Since each iteration of the rewiring process allows one to add/remove a wire at the fanout of a certain node, the basic idea of our approach is to improve the low observability of a node by adding new fanouts to this node. In the following, two special types of rewiring extended from [6] are analyzed, and the effects to circuit testability caused by such rewirings are then investigated. Before further discussion, several definitions are provided. Node B is a transitive fanout of node A, if there exists a path from node A to node B. Node A is a *dominator* of node B (wire C) if every path from node B (wire C) to any primary output must pass through node A. For example, in Fig. 1(a), nodes g_8 and g_9 are dominators of both node g_4 and wire g_1->g_4.

3.1 Two special types of rewiring

Here, we will describe two special types of rewiring. The first one is named the *fanout rewiring*. In this case, a fanout wire g_a->g_b of gate g_a dotted in Fig. 2(a) is replaced by another fanout wire g_a->g_d, bold in Fig. 2(b); additionally, node g_d is restricted to be a dominator of g_b and both gates g_d and g_b must have the *same* type of function, e.g., they are both OR gates in this example. In Fig. 2, let Cone II contain all the wires and gates that are the transitive fanouts of g_a and are dominated by g_d, while let Cone I contain all the wires dominated by g_a. We have the following theorem:

Theorem 1: Suppose wire g_a->g_b can be replaced by g_a->g_d where g_d is a dominator of g_b. A test pattern which can detect a stuck-at fault in Cone II after rewiring (Fig. 2(b)), can also detect the same fault before rewiring (Fig. 2(a)). Further, if a test vector can detect a fault in Cone I before rewiring (Fig. 2(a)), the same vector can also detect the fault after rewiring (Fig. 2(b)).

Proof: Consider testing line *w* stuck-at fault in the rewired circuit shown in Fig. 2(b). Because g_d is a dominator of *w*, the side-input g_a must be assigned the noncontrolling value *0*. Assigning *0* to g_a will cause both wires g_a->g_d in Fig. 2(b) and g_a->g_b in Fig. 2(a) to be "transparent", i.e., their presence will not affect the result. Therefore, any test pattern which can detect *w* stuck-at fault for the circuit after line substitution is also a test pattern for the same fault in the circuit before line substitution. The case for Cone I and other gate types can be proved similarly. Q.E.D.

Lemma 1: If wire g_a->g_b is replaced by g_a->g_d, the testability of a fault in Cone II decreases while that in Cone I increases.

Proof: This can be directly proved from Theorem 1.

The intuition behind the above theorem and lemma is that, after replacing g_a->g_b by g_a->g_d, node g_a has a fanout wire which is closer to the primary outputs; as a result, the observability of g_a increases and so do the wires dominated by

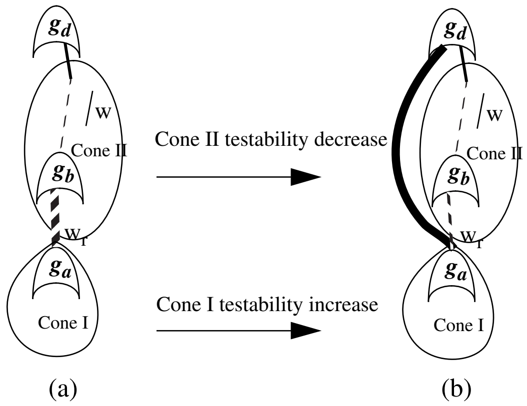

Fig. 2 : Fanout rewiring.

g_a. On the other hand, the wires in Cone II has the additional constraint: g_a must be *0* during testing. Therefore, the testability for wires in Cone II will decrease after wire replacement.

A second type of rewiring is referred as the *fanin rewiring*. Let the fanin wire, g_a->g_d of gate g_d dotted in Fig. 3(a), be replaced by another fanin wire g_b->g_d, bold in Fig. 3(b). Further, wires dominated by g_a are referred as Cone III and wires dominated by g_b are referred as Cone IV. Suppose there is no multiple fault cancellation problem, the observability of the wires in Cone III will decrease after rewiring because one fanout wire of the gate g_a has been deleted. Since an additional fanout wire is added to gate g_b, the observability of wires in Cone IV will potentially increase.

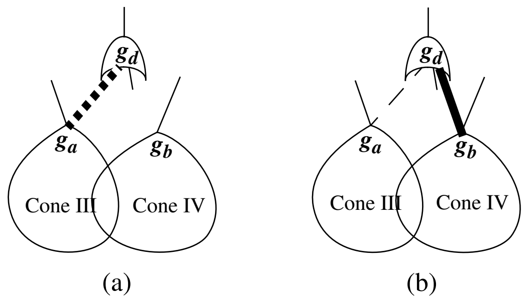

Fig. 3 : Fanin rewiring.

The change of testability after rewiring can be used to guide wire replacement to improve the circuit testability. For example, since the circuit's primary outputs are fully observable, there is no loss of observability by removing any of its fanouts. Consider wire g_5->g_9 in Fig. 1(b), since g_5 is directly connected to primary output O_1, node g_5 is fully observable. Removing another fanout from g_5 will not decrease the observability of g_5 at all. Therefore, replacing g_5->g_9 by g_1->g_4 will not decease the observability of g_5 but may increase the observability of g_1 because the number of g_1's fanouts is increased by one.

3.2 A heuristic for improving testability

There are two possibilities in using the circuit rewiring technique for testability enhancement. The first one tries to improve the testability of a CUT without considering what kind of patterns are applied, while the second approach tries to improve the fault coverage for a given set of patterns.

In the first approach, we estimate the testability of each node in the CUT. The estimation can be done by applying testability analysis methods [2] [8] [9] [11] . From the estimation, a set of low-testability nodes can be identified, to which new fanouts can be added (if possible). In the rewiring technique, whenever a wire is added to improve the observability of node *a*, the fanout wire of another node *b* must be removed and the observability of node *b* will be reduced. In order to limit the impact of testability loss, we choose to remove a wire whose source node either has many fanouts or has another fanout close to primary outputs. In this way, we are able to "shift" some testability from high testability nodes to low testability nodes. Although the first approach sounds promising, initial results obtained based on computer simulation demonstrate that this approach does not improve the fault coverage too much. The reasons are: (1) the testability analysis tools are not accurate enough to guide the rewiring process, and (2) there is no guarantee for each rewiring to improve the fault coverage. Thus, we concentrate on developing the second heuristic stated below.

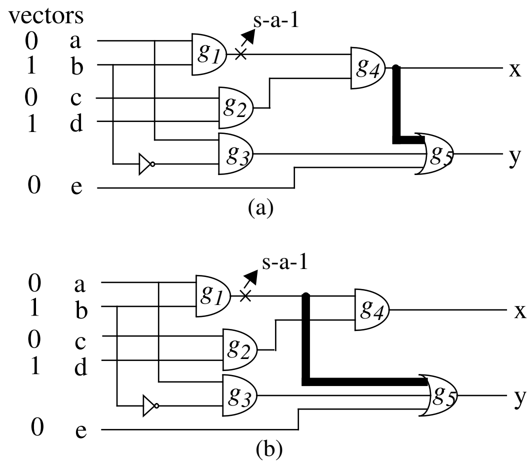

Fig. 4 : An example for testability enhancement.

In the second approach, we assume that a set of random or deterministic test patterns has been given and try to improve the fault coverage for these patterns. First, the patterns are simulated and faults which are not covered by these patterns are identified. Then, we index nodes in such a way that a node which is the transitive fanout of many undetected faults is given a large number. The objective in this approach is to improve the observabilities for nodes with large indices because increasing the observabilities for these nodes can greatly improve the observabilities of many undetected faults. We will use the same technique as the one used in the first approach to add new fanouts for nodes with large indices. After each rewiring, the circuit must be re-simulated because the fault detection behavior might be changed after each iteration of the rewiring step has been taken. That is, the rewiring process can not guarantee the detection of undetected faults; even worse, some detected faults might turn out to be undetected. Thus, fault simulation must be performed for each iteration to guarantee that the rewiring process is beneficial.

Consider the example shown in Fig. 4(a) with test vector (a, b, c, d, e) = (0, 1, 0, 1, 0). After fault simulation using the test vector, it can be found that stem line g_1 *stuck-at 1* cannot be detected, and an index is assigned to it. By assumption, node g_1 has the largest index, so one more fanout is added to node g_1 for improving its observability. Since g_1->g_5 (bold in Fig. 4(b)) can replace g_4->g_5 (bold in Fig. 4(a)) whose source node g_4 has high observability, we perform this wire replacement in Fig. 4(b). After rewiring, the fault simulation process is performed again, and stem line g_1 *s-a-1* is successfully detected by test vector (0, 1, 0, 1, 0). The process will be repeated until either there is no fault remaining undetected or no further rewiring is possible.

The problem with the second approach is that, for each rewiring, the fault simulation process must be executed to ensure that fault coverage is not devastated. It will be more efficient if there are rules to guide the wire selection and the rewiring processes such that the fault simulation process can be avoided for each rewiring. However, this is very difficult (if not impossible) since the rewiring process results in global effects to the entire circuit so no simple rules can be derived. Thus, to ensure a better solution, the fault simulation needs to be performed for each rewiring process. However, we feel that this is still tolerable because the process of rewiring for testability is a one-time process and is therefore a worthwhile effort.

4 Simulation Results

In order to evaluate the performance of the rewiring technique, the simulation software implementing the second approach has been developed based on the SIS environment. The pseudocode is given in Fig. 5. The results of computer simulation are shown in Table 1, and it can be found that the rewiring process improves random pattern fault coverage in most cases. The fourth column of the table gives the number of faults detected for each original circuit, while the sixth column gives that for each rewired circuit. Notice that different random test pattern lengths are used depending on the random pattern resistances of these circuits. It appears that the rewiring process is able to break the random test pattern barriers for many circuits. For example, the fault coverage of circuit ALU2 remains 84.05% after 3,000 random test patterns have been applied. In fact, we tried 50,000 random test patterns for ALU2 and failed to improve the fault coverage. However, after rewiring, the fault coverage can be easily improved to 86.21% by 1,000 random test patterns and to 87.06% by 3,000 random patterns. Each number *n* following the asterisk (**n*) in Table 1 indicates that the coverage can no longer be improved even by applying *n* random test patterns. Thus, it has been observed that circuit rewiring has successfully broken the random pattern barriers for many circuits. Of course, there are cases where rewiring fails to improve the fault coverage, so other design for testability methods which really add extra circuit must be used. All experiments were performed on the platform of an UltraSparc workstation.

```
rewire_for_testability (network, n) {
    PN = generate_random_patterns (n);
    FC_o = fault_simulation (network, PN);
    index_all_node (network);            (Reference Section 3.2)
    for each node i {
        for each fanin node k {
            if (there exists alternative wire) {
                find the node m with the maximum index;
                rewire k->i to wire m->i;
                FC_m = fault_simulation (network, PN);
                if (FC_m <= FC_o)
                    undo rewiring;            (rewire m->i to wire k->i)
                else
                    FC_o = FC_m;
            } } }
    return FC_o;
}
```

Fig. 5 : Pseudocode for Approach 2

Table 1: Simulation Results.

Circuit	# of Faults	# of Random Patterns	# of DF of ckt.org	FC of ckt.org	# of DF of ckt.rm	FC of ckt.rm	Time (sec)
9symml	774	500	709	91.60%	755	97.55%	556.68
		100	760	98.19%	773	99.87%	435.56
		2000	771	99.61%	774	100.00%	233.07
C1355	1984	500	1876	94.56%	1876	94.56%	54.07
C1908	2116	500	1804	85.26%	1804	85.26%	6487.14
		1000	1974	93.29%	1974	93.29%	4692.04
C2670	3090	1000	2498	81.68%	2550	82.52%	11826.18
		5000	2537	82.10%	2555	82.69%	21795.32
C3540	4440	2000	4222	95.09%	4222	95.09%	18271.36
C432	736	500	690	93.75%	692	94.02%	245.41
		1000	690	93.75%	692	94.02%	266.08
		*50000	690	93.75%	692	94.02%	1852.26

Table 1: Simulation Results.

Circuit	# of Faults	# of Random Patterns	# of DF of ckt.org	FC of ckt.org	# of DF of ckt.rm	FC of ckt.rm	Time (sec)
C499	1568	500	1494	95.28%	1502	95.79%	666.08
		1000	1535	97.90%	1543	98.41%	735.17
		2000	1558	99.36%	1566	99.87%	904.12
		5000	1560	99.49%	1568	100.00%	1403.15
		*100000	1560	99.49%	1568	100.00%	18953.08
C6288	9408	500	9404	99.96%	9404	99.96%	1262.23
C880	1280	1000	1221	95.39%	1225	95.70%	723.66
		5000	1259	98.36%	1260	98.44%	451.40
		10000	1268	99.06%	1268	99.06%	71.10
alu2	1762	1000	1464	83.09%	1519	86.21%	3766.57
		3000	1481	84.05%	1534	87.06%	5351.31
		*50000	1481	84.05%	1534	87.06%	83394.63
alu4	3164	5000	2775	87.71%	2843	89.85%	53492.16
apex6	2432	5000	2356	96.88%	2372	97.53%	3156.88
		10000	2383	97.99%	2398	98.60%	3239.53
apex7	922	500	831	90.13%	840	91.11%	314.05
		1000	837	90.78%	845	91.65%	363.05
		5000	849	92.08%	857	92.95%	706.08
		10000	859	93.17%	866	93.93%	933.03
		20000	863	93.60%	870	94.36%	1692.35
		*100000	863	93.60%	870	94.36%	8300.22
b1	48	500	30	62.50%	34	70.83%	0.01
		*100000	30	62.50%	34	70.83%	3.02
b9	480	500	465	96.88%	472	98.33%	56.10
		1000	471	98.12%	474	98.75%	61.09
		*50000	471	98.12%	474	98.75%	442.12
c8	930	1000	758	81.51%	763	82.04%	634.91
		5000	787	84.62%	787	84.62%	657.02
cc	262	100	228	87.02%	233	88.93%	11.03
		200	258	98.47%	259	98.85%	2.01
		300	262	100.00%	262	100.00%	0.00
cm85a	176	100	156	88.64%	163	92.61%	4.02
		200	174	98.86%	175	99.43%	1.01
		300	176	100.00%	176	100.00%	0.00
cmb	164	1000	103	62.80%	105	64.02%	4.02
		5000	138	84.15%	138	84.15%	9.02
		10000	150	91.46%	150	91.46%	13.03
comp	510	5000	304	59.61%	310	60.78%	38.06
		10000	326	63.92%	332	65.10%	69.09
		50000	422	82.75%	424	83.14%	289.52
cordic	328	500	256	78.05%	257	78.35%	81.20
		1000	315	96.04%	315	96.04	21.01
cu	230	500	204	88.70%	209	90.87%	19.06
		1000	205	89.13%	210	91.30%	21.05
		5000	211	91.74%	216	93.91%	42.09
		10000	212	92.17%	217	94.35%	37.02
		*50000	212	92.17%	217	94.35%	162.08

5 Conclusions

In this paper, we have presented a method to enhance the random pattern testability by circuit rewiring. The basic idea is to remove fanouts from gates whose observabilities have been determined to be relatively high; meanwhile, alternative wires are added to increase the observabilities for gates whose observabilities are relatively low. The wire addition and removal process does not change the circuit function. Compared with other random testability enhancement approaches, the proposed circuit rewiring technique does not introduce any hardware overhead, thus it is basically a cost-free method. Simulation results demonstrate that the rewiring technique is able to successfully break the random pattern barrier of the CUT and is thereby able to improve the random test fault coverage efficiently for most circuits. The approach can also be incorporated into other random testability enhancement methods (needs hardware overhead) as a preprocess, i.e., the random testability can first be improved using the proposed method, then other methods are used to gain higher fault coverage with tolerable hardware overhead. So far, the majority of computing time is used in fault simulation for each rewiring process. Future research includes: (a) finding rules to guide circuit rewiring to avoid excessive fault simulation, or (b) using a smarter fault simulation method if the rules can not be obtained.

6 References

[1] M. Abramovici, M.A. Breuer, A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] V. D. Agrawal and S. C. Seth, “Probabilistic testability,” *Proc. of Int. Conf. Computer. Design: VLSI Computers*, 1985, pp.562-565.

[3] P. H. Bardell, W. H. McAnney, J. Savir, "Built-In Test for VLSI Pseudorandom Techniques,” New York, Wiley, 1987.

[4] S. Chakravarty and H. B. Hunt III, “On computing Signal Probability and Detection Probability of Stuck-at Faults,” IEEE Trans. on Computer, Vol 39, No, 11, Nov 1990, pp. 1369-1377.

[5] S. C. Chang, K. T. Cheng, N.S. Woo and M. Marek-Sadowska “Layout Driven Logic Synthesis for FPGA,” *Proc. Design Automation Conf.* pp. 308-313, June 1994

[6] S.C. Chang, M. Marek-Sadowska, and K.T. Cheng, “Perturb and Simplify: Multi-level Boolean Network Optimizer”, IEEE Transaction on Computer Aided Design, Vol. 15, pp. 1494-1504, Nov, 1996.

[7] K. T. Cheng and L. A. Entrena, “Multi-Level Logic Optimization by Redundancy Addition and Removal,” in *Proc. European Conference On Design Automation*, pp. 373-377, Feb. 1993.

[8] S. Chakravarty and H. B. Hunt III, “On computing Signal Probability and Detection Probability of Stuck-at Faults,” *IEEE Trans. on Computer, Vol 39, No, 11,* Nov 1990, pp. 1369-1377.

[9] S. K. Jain and V. D. Agrawal, “Statistical Fault Analysis”, *IEEE Design & Test Computers*, Vol 2, No 2, 1985, pp. 38-44.

[10] W. Kunz and D.K. Pradhan, “Multi-Level Logic Optimization by Implication Analysis”, *Digest Int. Conf. on Computer Aided Design,* pp. 6-13, Nov.1994.

[11] L.H. Goldstein, “Controllability/Observability Analysis of Digital Circuits,” *IEEE Trans. on Circuits and Systems, vol. CAS-26*, Sep, 1979, pp. 685-693.

[12] B. Rohfleisch, B. Wurth, K. Antreich “Logic Clause Analysis for Delay Optimization”, Proc. DAC, 1995, pp. 668-672.

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

[14] H. D. Schnurmann, E. Lindbloom, and R. F. Carpenter, “The Weighted Random Test-Pattern Genertator,” *IEEE Trans. on Computers*, Vol. C-24, No. 7, pp. 695-700, July, 1975.

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

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

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

[18] H. C. Tsai, K.T. Cheng, C. J. Lin, S. Bhawmik, “A Hybrid Algorithm for Test Point Selection for Scan-Based BIST,” *Design Automation Conf.*, 1997, pp. 478-483.

[19] H. J. Wunderlich, “Self Test Using Unequiprobable Random Patterns,” *Digest of Papers 17th Intn’l. Symp. on Fault-Tolerant Computing*, pp. 258-263, July, 1987.