I_{DDQ} Measurement Based Diagnosis of Bridging Faults in Full Scan Circuits

Sreejit Chakravarty[†] and Suresh Sivaprakasam
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260

Technical Report No: 93-04

sreejit@cs.buffalo.edu

February 4, 1993

Abstract

An algorithm for diagnosing two node bridging faults in static CMOS combinational circuits(full scan circuits) is presented. This algorithm uses results from I_{DDQ} testing. The bridging faults considered can be between nodes that are outputs of a gate or internal nodes of a gates. Experimental results on all the ISCAS89 circuits show that I_{DDQ} measurement based diagnosis using a small number of randomly generated vectors, is very effective. Moreover, it is computationally feasible to diagnose such a large number of bridging faults when I_{DDQ} measurement is used.

1 Introduction

A *bridging fault* is said to occur when two or more normally disconnected nodes of the circuit are connected. Only static CMOS combinational(or full scan) circuits are considered. Consequently, every gate has an *output node* and a number of *internal nodes*. In this study, the two nodes involved in the bridging fault could be either an *output node* or *internal node*. Furthermore, we consider all such bridging faults. Bridging faults in static CMOS circuits, can be detected by monitoring the steady state current(I_{DDQ}) and using I_{DDQ} *Tests*. This is discussed in more detail in section 2.

The motivation for the studying such faults is based on recent findings that bridging faults occur very frequently [10, 11]. However, because of the large number of such faults many researchers restrict the set of bridging faults to a relatively smaller number [12]. For this, the layout of the circuit is analyzed assuming the point defects. We use a different approach in that we consider all bridging faults for the following reasons. The layout of the circuit is not always available. Even if they are available they are often too large so that fault extraction will take a long time. In addition, this extraction process has to be executed every time the circuit is layed out using a different tool or different run of the same tool.

The diagnosis problem is defined as follows. Given a combinational circuit C, detected by an I_{DDQ} test set to be faulty, identify the fault in C. We assume that the gate level description of the circuit is available, along with the test set; furthermore we assume that there is only *one* bridging fault of the type described above; and an indication as to which input vector gave a *faulty response*.

[1]This research was partially supported by NSF grant No. MIP9102509

In this paper we present an I_{DDQ} measurement based diagnosis algorithm for bridging faults. This algorithm is a modification and improvement of the I_{DDQ} measurement based diagnosis algorithm for bridging faults between *output nodes* of a circuit from [5]. This algorithm in [5] unlike the one we present here, cannot correctly diagnose bridging faults in which one or both the nodes is an *internal node.*

The diagnosis algorithm we present has some important characteristics that makes it very efficient. Traditional diagnosis algorithms use fault dictionaries. These fault dictionaries are then used to drop the faults from the fault list. Any algorithm that uses fault dictionaries to diagnose the large number of bridging faults, that we consider will require will require a large amount of space. We therefore do not use fault dictionaries but a rather space efficient representation of the fault list(SOPS) which is discussed in section 3. Secondly, no fault simulation is required. We use only logic simulation. Computationally efficient rules for fault dropping are used by the algorithm. Besides diagnosing the algorithm an important point that we would like to make is that the results point to the computational feasibility of diagnosing the large number of bridging faults. In addition, I_{DDQ} measurement based diagnosis is also shown to be very effective in diagnosing faults.

Diagnosis for a variety of fault models have been discussed. The diagnosis algorithms in [1,2,7,8,15,17,18,19] assume that the faults are stuck-at faults and input-output experiments are used to detect the presence of faults. The work in [3] is for *leakage faults*, in combinational or LSSD based circuits, where I_{DDQ} measurements are used. The algorithms in [4,5] are I_{DDQ} measurement based diagnosis algorithms for bridging faults that involves only the output nodes of the circuit. The diagnosis algorithm for bridging faults in [13] assumes input-output experiments and uses fault dictionaries. The diagnosis algorithm for bridging faults in [6] is for a subset of static CMOS circuits and assumes input-output experiments.

2 Definitions

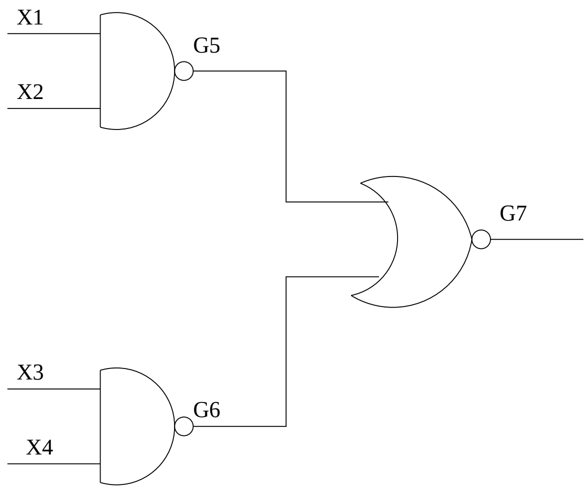

Figure 1: Gate Level Description

The two line single bridging faults(TSBFs) are classified into two classes, as follows. If X is a primary input or an output node then Gate(X)is X itself. Else, Gate(X) is Y where X is an internal node of gate Y. The TSBF $< X, Z >$ is said to be a feedback TSBF(FTSBF) iff there exists a path either from Gate(X) to Gate(Z) or vice-versa. Else , $< X, Z >$ is a non-feedback

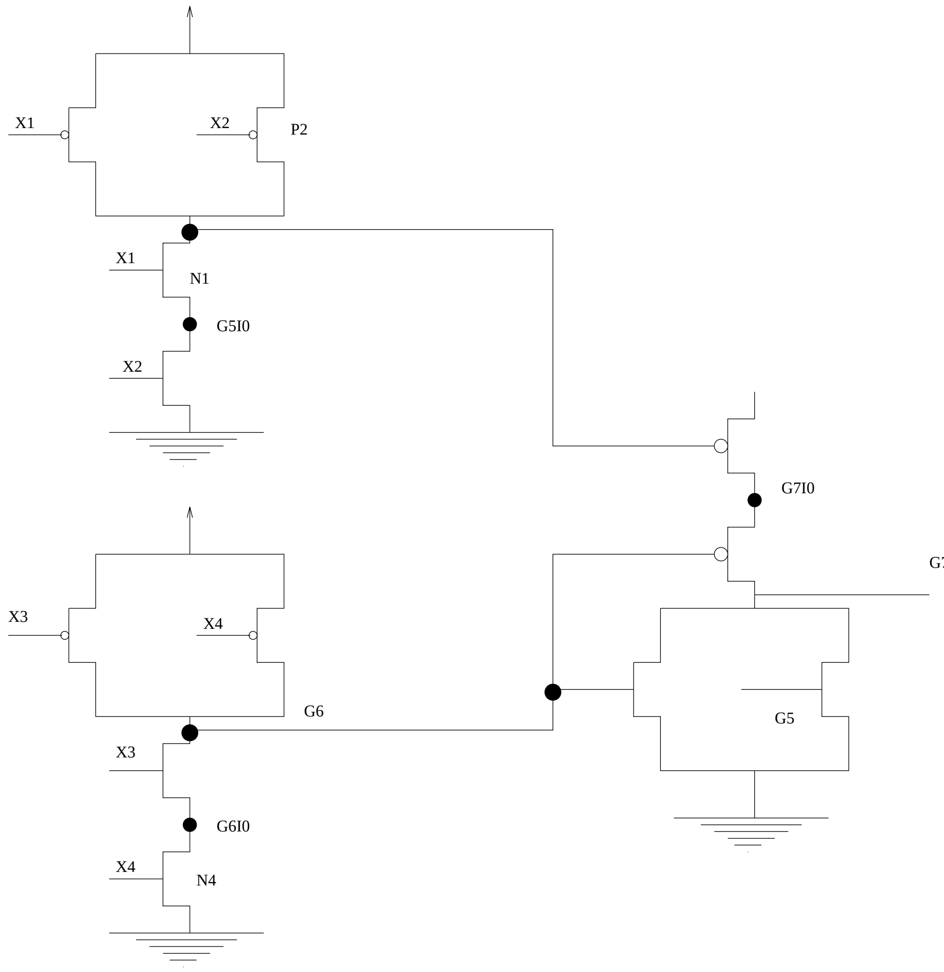

Figure 2: Switch Level Description

TSBF(NFTSBF). Figure 1 shows the gate level circuit and Figure 2 shows switch level of the same circuit. In Figure 1, $< G5, G6I0 >$ is a NFTSBF whereas <G5, G6I_0 > is a FTSBF. Note that G6I_0, G7I_0, G5I_0 are the internal nodes of gates G6, G7 and G5 respectively where the suffix I_j denotes the j^{th} internal node.

Below we describe Prop 2.1 which identifies an input vector to be an I_{DDQ} test for a TSBF. Note that on application of an input vector T, the value of node X, denoted by $T(X)$, can be either 0,1 or f(floating).

Let $T_1 =< X1 = 1, X2 = 0, X3 = 0, X4 = 1 >$. On application of T_1 to the circuit of Figure 2, a path is activated from V_{DD} to G5I_0 through P2, G5 and N1. Also the path from G6I_0 through N4 to V_{SS} is activated. If there is a bridging fault between G5I_0 and G6I_0 there will be a conducting path from V_{DD} to V_{SS} through these activated paths. Consequently there will be a non-zero I_{DDQ} which can be detected and used to signal the presence of a fault[9, 16]. Prop 2.1 is a somewhat more pessimistic than the conditions used in [14].We formalize this example in the following discussion.

Prop 2.1: T is an I_{DDQ} test for the NFTSBF $< X, Y >$ iff $T(X), T(Y) \in \{0, 1\}$ and $T(X) \neq T(Y)$. Let F be a FTSBF s.t. there exists a path from Gate(X) to Gate(Y) and $T'(Y)$ the value at Y on application of T, obtained by setting Gate(X) to u and doing the 3-valued simulation.

Let $T_2 =< X1 = 0, X2 = 0, X3 = 1, X4 = 1 >$ then T_2(G5I_0) is f. Therefore, T_2 cannot be an

I_{DDQ} test for <G5I_0, G6I_0 >. Consider the FTSBF $< X3, G7I0 >$. T_2(G7I_0) $= u$ and $T_2(X3) = 1$. Therefore, T_2 is not a test for $< X3, G7I0 >$. However, $T_3 =< X1 = 1, X2 = 0, X3 = 1, X4 = 1 >$ is a test for the FTSBF <G7I_0, $X3$ >. Note that in this case T_3'(G7I_0) $= 0$. Therefore the path from G7I_0 to V_{SS} exists independent of the value at X1 i.e. also in the faulty circuit.

3 Basic Idea

We introduce SOPS(set of ordered pairs of sets) which is used to represent the set of possible bridging faults as well as the diagnosis algorithm with the help of an example. The circuit of Figure 3 (s27 of ISCAS89) is used as an example. It is assummed that the fault is <G8, G10I_0 > and the response is computed using this assumption. A test set along with the response is given in Table 1. An entry of 1(0) in the last column of row i implies that there was a large(small) I_{DDQ} reading on application of T_i.

3.1 Notation

We use **SOPS**, a compact representation of TSBFs. SOPS is a set of ordered pairs of the form $< A_1, B_1 >, < A_2, B_2 >, \ldots, < A_k, B_k >$ where A_i is a set of lines in the circuit, and B_i can be a either a set of lines in the circuit or the special symbol ψ. If B_i is a set of lines then the ordered pair $< A_i, B_i >$, denotes the set of TSBFs $< X, Y >$ where $X \in A_i, Y \in B_i$. If B_i is the special symbol ψ then $< A_i, \psi >$ denotes the TSBFs $< X, Y >$ where $X, Y \in A_i$. For example, the SOPS L = {<{G17 G2},{G5 G13}> <{G3I_2 G6 G5I_3}, ψ>} represents the following sets of TSBFs :

{ <G17 G5> <G17 G13> <G2 G5> <G2 G13> <G3I_2 G6> <G3I_2 G5I_3>
<G6 G5I_3> }
G17 G2 G5 G13 G6 are *output* nodes
G3I_2 G5I_3 are *internal* nodes.

3.2 Example

To convert the sequential circuit into a scan circuit we delete D Flip-Flops and assume its inputs to be a primary *output* and its output to be primary *input*. The algorithm initially assumes all the faults are possible. For the circuit s27, the TSBF is respresented by $SOPS$ L_0.

L_0 =
<{G0 ... G9 G10I_0 G11I_0 G13I_0 G8I_0 G15I_0 G12I_0 G16I_0 G9I_0}, ψ>

Next the input vectors are processed and faults are dropped from the current set of possible faults to get a new set of possible faults. Let L_i be the SOPS denoting the possible set after processing the input vector T_i. Let A be the set of all lines in the circuit, sets B, C, D be the subsets of A set to 0,1,f respectively on application of T_1 s.t. $B \cup C \cup D = A$.

B = {G0 G9I_0 G12I_0 G8I_0 G9 G12 G7 G10 G5 G17 G3 G2}
C = {G1 G16I_0 G15I_0 G13I_0 G11I_0 G16 G15 G8 G14 G13 G11 G6}
D = {G10I_0}

Now since T_1 *didn't* give a faulty response, therfore the fault is *between* the lines that are at 0, or lines that are at 1 or between a *floating* line and a line set to 0 or 1.These TSBFs are represented

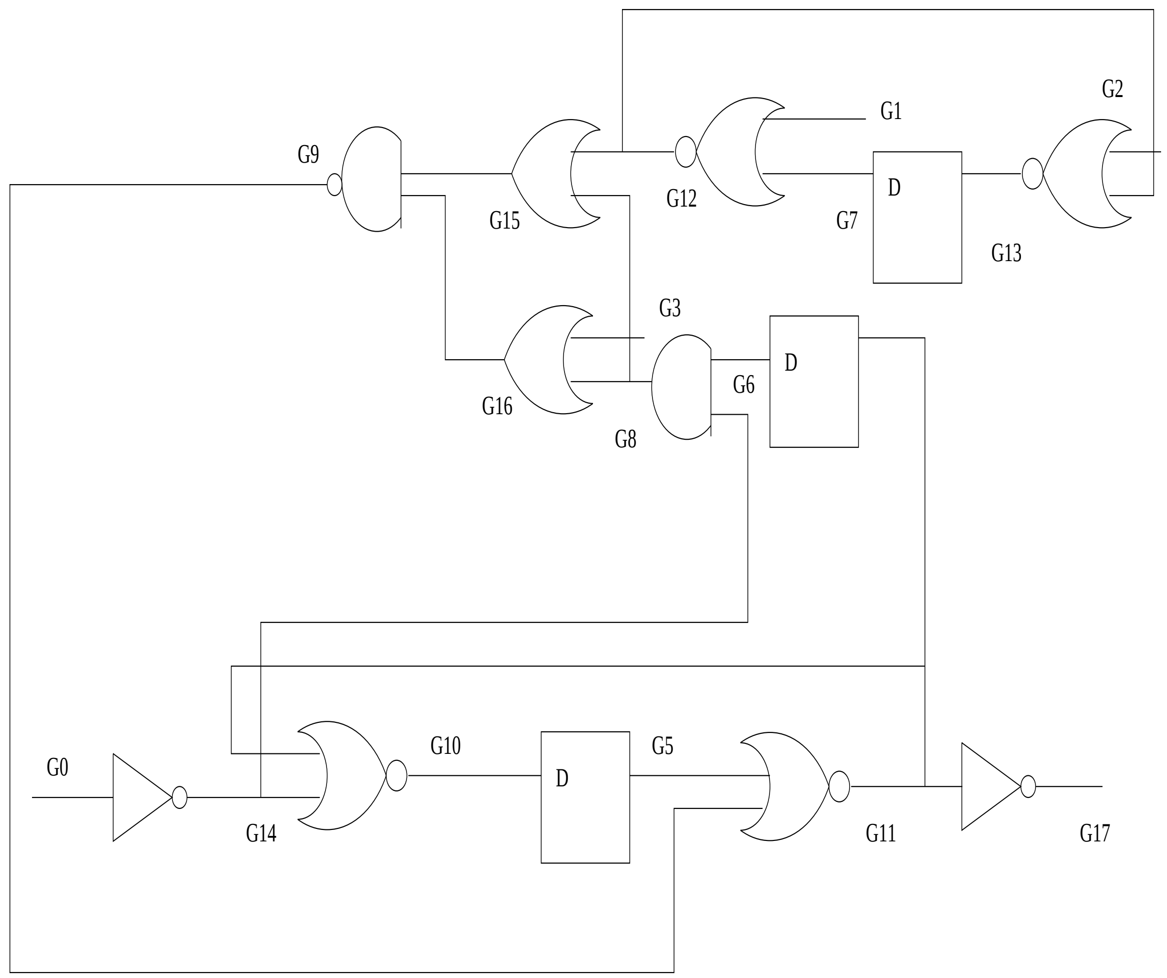

Figure 3: Example Circuit $s27$

by $SOP\ L_1$.

$L_1 = \{< B \cup D, \psi >, < C \cup D, \psi >\}$

Next, we process T_2, which gives a faulty response , and E, F, G be subsets of $B \cup D$ (first ordered pair) that are set to 0,1,f on application of T_2.

E = {G9I_0 G2 G17 G5 G10 G7 G9 G8I_0}
F = {G0 G10I_0 G3 G12 G12I_0}
$G = \phi$

Similarily let H, J, K be the subsets of $C \cup D$(second ordered pair), that are set to 0,1,f on application of T_2.

H = {G1 G13 G14 G8 G15I_0 G16I_0}
J = {G13I_0 G10I_0 G6 G11 G15 G16 G11I_0}
$K = \phi$

Now since T_2 *did* give a faulty response, the fault is *between* two nodes at 0 and 1. These TSBFs make the new $SOP\ L_2$

$L_2 = \{< E, J >, < F, H >\}$

Similarily we process T_3. Since T_3 *didn't* give a faulty response, the fault is between nodes at 0 or 1 or between a node at f and a line at 0 or 1. On iterating through the entire $SOP\ L_2$, from the resulting TSBFs we get the new $SOP\ L_3$

Continuing this way, after processing T_{10},

L_{10} = {<G10, G12>, <G8,G10I_0>}

From this we will deduce the set of possible bridging faults.
{G10 G8}, {G10 G10I_0}, {G12 G8}, {G12 G10I0}

This is the basic idea of the algorithm but needs soem modificationto correctly diagnose FTSBFs. This is discussed in the next section.

4 Algorithm DIAG-BFC-INTER

Notations

- N is the size of the test set.
- $T = \{T_1, \ldots, T_N\}$ is the test set.
- $R = \{r_1, \ldots, r_N\}$ is the response vector where $\forall i, r_i \in$ {0,1} and r_i = 0(1) if there wasn't (there was) a large I_{DDQ} on application of T_i.
- C is the circuit being diagnosed.
- X is the set of all nodes(includes *internal nodes*) in C.
- m is the number of gates in C.
- $T_i(y)$ is the value of the node y on application of T_i
- n is the number of primary inputs of C
- $\{X_1, \ldots, X_n\}$ is the set of primary inputs of C
- $A^j(T_i)$ is the subset of A s.t. $\forall\ y \in A_j(T_i)$, $T_i(y) = j, j \in$ {0, 1, f}, f implies floating.
- $< A, \psi >$ is the set of pairs of lines $< y_1, y_2 >$ s.t. $y_1, y_2 \in A$
- $< A, B >, B \neq \psi$, is the set of pairs of lines (y_1, y_2) s.t. $y_1 \in A$ and $y_2 \in B$
- L_i and M_i are the sets of pairs of lines,denoting the set of possible faults deduced by the algorithm respectively, after processing all the vectors $T_1, \ldots, T_i$ and *only* the *good* vectors alone.

Step 1. $M_0 = L_0 = \{ < X, \psi > \}$.
Step 2. For each T_i, $1 \le i \le N$ do **Steps 3, 4.**
Step 3. Let $L_{i-1} = < A_1, B_1 >, \ldots, < A_q, B_q >$. For $1 \le j \le q$ process $< A_j, B_j >$ to form new ordered pairs as follows. These new ordered pairs are added to L_i and $< A_j, B_j >$ is deleted from L_{i-1}.
Compute $A_j^0(T_i)$,$A_j^1(T_i)$, $A_j^f(T_i)$. If $B_j \neq \psi$, compute $B_j^0(T_i)$, $B_j^1(T_i)$, $B_j^f(T_i)$. We have two cases.

- **Case 3.1**
  - Rule 1. For $B_j = \psi$ and $r_i = 0$

    $L_{i+1} = L_i$ - $\{< A_j, \psi >\}$ +
    $\{< A_j^0(T_i) \cup A_j^f(T_i), \psi >, < A_j^1(T_i) \cup A_j^f(T_i), \psi >\}$
  - Rule 2. For $B_j = \psi$ and $r_i = 1$

    $L_{i+1} = L_i$ - $\{< A_j, \psi >\}$ + $\{< A_j^0(T_i), A_j^1(T_i) >\}$

- **Case 3.2**
  - Rule 3. For $B_j \neq \psi$ and $r_i = 0$

    $L_{i+1} = L_i$ - $\{< A_j, B_j >\}$ +
    $\{< A_j^0(T_i), B_j^0(T_i) >, < A_j^1(T_i), B_j^1(T_i) >, < A_j^f(T_i), B >, < A, B_j^f(T_i) >\}$
  - Rule 4. For $B_j \neq \psi$ and $r_i = 1$

    $L_{i+1} = L_i$ - $\{< A_j, B_j >\}$ +
    $\{< A_j^0(T_i), B_j^1(T_i) >, < A_j^1(T_i), B_j^0(T_i) >\}$

Note: While processing T_i, before adding a *ordered pair*, we check L_i, if it is already present, if true we *don't* add. Further if either A or B is empty then $< A, B >$ denotes an empty set of faults and is therefore not added to the list L_i. Similar rules apply for $< A, \psi >$ also.
Step 4. Compute M_{i+1} from M_i as follows. If $n_i = 0$ then $M_{i+1} = M_i$. Else, use Rule 3,4 to compute M_{i+1} from M_i as was done above.
Step 5. Let FTSBF(M_N) = { set of FTSBFs in M_N }. Then, the final set of possible faults PS = $L_N \cup$ FTSBF(M_N).

To complete the example that was started in the previous section we note that
L_{10} = {<G10 G12>, <G8 G10I0>}
M_{10} = {<G8I_0 G16I_0>, <G9 G12I_0>}

Therefore, PS = {<G10 G8>, <G10 G10I0>, <G12 G8>, <G12 G10I0>,
< G8I_0 G9> <G8I_0 G12I_0> <G16I_0 G9> < G16I_0 G12I_0>}

Thus from a total of 153 faults we have narrowed it down to 8.

5 Proof of Correctness

To show that DIAG-BFC-INTER is correct we have to show that for any TSBF F, if F can result in the observed response, then F must be a member of the final set PS. To show this we will show that if F is an FTSBF then F will be in M_N and if F is an NFTSBF then F will be in L_N. The correctness of the algorithm follows from these two lemmas and the definition of PS.

Lemma 5.1. Let F be any NFTSBF s.t. the presence of F in the given circuit C leads to the observed I_{DDQ} response. Then, $F \in L_N$.

Proof: The proof is by induction on N, the size of the test set. Since $L_0 = \{< X, \psi >\}$ and contains all the TSBFs, it forms the basis step. Assume that $F \in L_{i-1}$. We next show that $F \in L_i$.

Let $L_{i-1} = \{< A_1, B_1 >, < A_2, B_2 >, \ldots, < A_q, B_q >\}$
$F =< a, b >$ and without loss of generality assume that $F \in< A_1, B_1 >$. We have two cases.

- **Case 1.** $B_1 = \psi$
  If $r_i = 0$ then Rule 1 drops the set of faults $< A_1^0(T_i), A_1^1(T_i) >$ from the set $< A_1, \psi >$ This set doesn't include F because F will be dropped only if $T_i(a), T_i(b) \in \{0, 1\}$ and $T_i(a) \neq T_i(b)$. But T_i *does* not detect F. Therefore, by Prop 2.1, either $T_i(a) = T_i(b)$ or one of $T_i(a), T_i(b)$ if f.

  If $r_i = 1$ then Rule 2 drops the set of faults $\{< A_i^0(T_i) \cup A_i^f(T_i), \psi >, < A_1^1(T_i) \cup A_1^f(T_i), \psi >\}$ from $< A_1, \psi >$. Thus, F will be dropped only if $T_i(a), T_i(b) \in \{0, f\}$ or $T_i(a), T_i(b) \in \{1, f\}$. But, T_i detects F. Therefore, by Prop 2.1, $T_i(a) \neq T_i(b)$, $T_i(a) \neq T_i(b)$, $T_i(a), T_i(b) \in \{0, 1\}$.

- **Case 2.** $B_i = \psi$.
  If $r_i = 0$ then Rule 3 drops set of faults $\{< A_1^0(T_i), B_1^0(T_i) >, < A_i^1(T_i), B_1^1(T_i) >\}$. Thus, F will be dropped only if $T_i(a) = T_i(b) = 0$ or $T_i(a) = T_i(b) = 1$. But, T_i does not detect F. Therefore by Prop 2.1, either $T_i(a) \neq T_i(b) \in \{0, 1\}$ or atleast one of $T_i(a), T_i(b)$ equals f.

  If $r_i = 1$ then Rule 4 drops the set of faults $\{< A_1^0(T_i), B_1^0(T_i) >, < A_1^1(T_i), B_1^1(T_i) >, < A_1^f(T_i), B_1^1(T_i) >, < A, B_1^f(T_i) >\}$. Therefore if F is dropped only if $T_i(a) = T_i(b) \in \{0, 1\}$ or one of $T_i(a), T_i(b)$ is f. But, T_i detects F. Thus by Prop 2.1 $T_i(a) \neq T_i(b)$ and $T_i(a), T_i(b) \in \{0, 1\}$.□

Lemma 5.2. Let F be any FTSBF s.t. the presence of F in the given circuit C leads to the observed I_{DDQ} response. Then $F \in M_N$.

Proof: Note that T_i is used to drop faults only if $r_i = 1$ i.e. Rules 2,4 are used. The proof is very similar to Lemma 5.1. We leave it to the interested reader. □

6 Experimental Results

In this section, we present experimental results obtained by running the DIAG-BFC-INTER on ISCAS89, sequential benchmarks circuits. The experiments were conducted on a SPARCstation 2 with 32MB of memory. For each circuit we generated 25 instances of the diagnosis problem, and used 100 test vectors. Random number generators were used to generate faults and vectors. For each *random* instance of the problem, a *random* test set $T_1, \ldots, T_{100}$ was generated. As T_i was generated, fault simulation was used to generate the *response bit* r_i, using Prop 2.1.

We used two figures of merit, to evaluate the algorithm. The first is *time*(CPU secs) and the second is *size*, of the final set of faults. The results are tabulated in Table 2 and are explained below. A point to note, the results shown in the table are *average* values.

The final value of the *size* and *time* are averaged for 25 instances of the problem and is displayed in the table. *Nodes* is the number of gates and primary inputs, *TotNodes* includes the internal nodes, *Totfaults* is the total number of TSBFs in the circuit, *size* is the number of TSBFs left as the set of possible faults and % is the percentage of the number faults left. The effectiveness of the diagnosis algorithm should be clear from this. The processing is very efficient. Even for the largest of the circuits it required less than an hour of CPU time.

Summary

We studied the problem of diagnosing TSBFs between any two nodes in the circuit. The diagnosis process uses results of I_{DDQ} measurements. A time and space efficient diagnosis algorithm was presented. Experiments using this algorithm established the effectiveness of reducing the possible set of faults using I_{DDQ} measurement, using a small number of randomly generated tests. The computational feasibility of I_{DDQ} measurement was also established.

References

1. M. Abramovici, “A hierarchical, path-oriented approach to fault diagnosis in modular combinational circuits”, *IEEE Trans. on Comput.*, Vol. C-31, No. 7, July 1982, pp. 672 - 677.

2. M. Abramovici and M. A. Breuer, “Multiple fault diagnosis of MOS Combinational Networks”, *IEEE Trans. on Comput.*, Vol. C-29, No. 6, June 1980, pp. 451 - 460.

3. R. C. Aitken, “Fault Location with Current Monitoring”, *IEEE International Test Conference*, 1991, pp. 623 - 632.

4. D. J. Burns, “Locating High Resistance Shorts in CMOS Circuits by Analyzing Supply Current Measurement Vectors”, *Int'l Symposium for Testing and Failure Analysis*, pp. 231 - 237, 1989.

5. S. Chakravarty, MinSheng Liu, “Algorithms for Current Monitor Based Diagnosis of Bridging and Leakage Faults”, Tech. Rep. No. 92-05, Department of Computer Science, State University of New York, Buffalo, NY 14260.

6. S. Chakravarty and Y. Gong, “ An algorithm for diagnosing bridging two line faults in Combinational circuits.” Tech Rep No 92-26, Dept Of Computer Scince, SUNY at Buffalo.

7. H. Cox and J. Rajski, “A Method for Fault Analysis for Test Generation and Fault Diagnosis”, *IEEE Trans. on Comput.-Aided Design*, Vol. 7, No. 7, July 1988, pp. 813 - 833.

8. Y. M. El-Ziq and S. Y. H. Su, “Fault Diagnosis of MOS Combinational Networks”, *IEEE Trans. on Comput.*, Vol. C-31, No. 2, Feb. 1982.

9. D. Feltman et. al., “Current Sensing for Built-In Testing of CMOS circuits”, *1988 Int'l Conf. on Computer-Aided Design*, pp. 454 - 457.

10. F. J. Ferguson and J. P. Shen, “Extraction and Simulation of Realistic Faults using Inductive Fault Analysis”, in *Proc. 1988 Int'l. Test Conference*, pp. 475-484.

11. F. J. Ferguson and J. P. Shen, “A CMOS Fault Extractor for Inductive Fault Analysis”, *IEEE Trans. on Comput.-Aided Design*, Vol. 7, No. 11, pp. 1181 - 1194, 1988.

12. S.D. Millman, et.al. “Diagnosing CMOS bridging faultswith stuck-at fault dictionaries,” Proc. Int. Test Conf. pp860-870, 1990.

13. P. Nigh and W. Maly, "Test Generation for Current Testing", *IEEE Design and Test of Computers*, Feb. 1990, pp. 26 - 38.

14. J. Richman, K. Bowden, " The Modern Fault Dictionary", In *Proc. International Test Conference*, pp. 696 - 702, 1985.

15. J. Soden and C. Hawkins, "Test Considerations for Gate Oxide Shorts in CMOS ICs", *IEEE Design & Test of Comput.*, Aug. 1986, pp. 56-64.

16. J. Waicukauski, J. V. Gupta and S. Patel, "Diagnosis of BIST failures by PPSFP Simulation", In *Proc. International Test Conference*, pp. 480 - 484, 1987.

17. J. Waicukauski, E. Lindbloom, "Failure Diagnosis of Structured VLSI", *IEEE Design and Test of Computers*, Vol. 6, No. 4, pp. 49 - 60, August 1989.

18. T. Yano, H. Okamoto, "Fast Fault Diagnosis Method using fault dictionary for electron beam tester", In *Proc. Int'l Test Conference*, 1987, pp. 561 - 565.

19. F. J. Ferguson and T. Larabee, "Test Pattern Generation for Realisitc Bridging Faults in CMOS ICs", *IEEE Int'l Test Conf.*, pp. 492 - 499, 1991.

T_i	0	1	2	3	17	5	6	11	7	13	14	8	15	12	16	9	10I0	11I0	13I0	8I0	12I0	16I0	9I0
T_1	0	1	0	0	0	0	0	1	1	0	1	1	1	0	1	0	2	1	1	0	1	0	1
T_2	1	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0
T_3	0	1	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1	1	0	0
T_4	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0	1	1	2	1	0	1	1	1
T_5	1	0	0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	2	1	2	1	1	0
T_6	0	1	1	0	1	1	0	0	0	0	0	1	0	0	0	1	0	2	0	1	1	0	1
T_7	0	0	1	1	1	1	0	1	0	0	0	1	1	1	1	0	0	0	2	0	2	1	2
T_8	1	0	1	0	1	1	1	0	0	1	0	0	0	0	0	1	1	2	0	2	1	1	1
T_9	0	0	1	0	1	1	0	0	0	0	0	1	1	1	0	1	0	2	2	1	0	1	1
T_{10}	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	0	2	1	1	0	1	2	2

Table 1.

Circuits	Nodes	TotNodes	TotFaults	Faults	Time	%
s1196	561	1041	541320	23	6.79	0.004
s1238	540	1073	575128	19	10.60	0.003
s13207	8651	11865	70383180	1467	567.12	0.002
s1423	748	1255	786885	6	12.32	0.000
s1488	667	1401	980700	2	10.41	0.000
s1494	661	1407	989121	25	10.51	0.000
s15850	10383	14256	101609640	1985	832.64	0.000
s13207.1	8651	11865	70383180	1743	884.5	0.000
s15850.1	10383	14256	101609640	2183	1329.06	0.002
s208.1	122	199	19701	15	1.29	0.07
s420.1	252	417	86736	11	4.33	0.012
s526n	218	468	109278	5	2.89	0.004
s838.1	512	853	363378	303	8.44	0.08
s9234.1	5844	8218	33763653	3852	445.60	0.010
s35932	17828	30032	450945496	2782	1845.78	0.000
s38417	20717	38533	742376778	3687	2173.04	0.000
s38584	23843	40323	812952003	5643	3243.53	0.000
s38584.1	23843	40323	812952003	6281	3539.66	0.000
s27	17	25	300	2	0.13	0.6
s298	136	261	83930	10	1.06	0.02
s344	184	293	42778	4	1.41	0.008
s349	185	297	43956	4	2.02	0.008
s386	172	360	64620	3	1.58	0.004
s444	205	376	70500	8	2.5	0.010
s510	236	449	100576	3	3.94	0.002
s526	217	469	109746	5	3.05	0.004
s5378	2993	4426	9792525	453	131.70	0.004
s641	433	593	175528	14	5.94	0.006
s713	447	645	207690	15	7.04	0.006
s820	312	780	303810	5	5.22	0.000
s832	310	792	313236	5	6.42	0.000
s9234	5844	8218	33763653	2837	404.61	0.008
s953	440	788	310078	345	7.34	0.10

Table 2.