On I_{DDQ} Measurement Based Analysis of Bridging Faults in CMOS Circuits[1]

Sreejit Chakravarty and Paul J. Thadikaran
Department of Computer Science
State University of New York at Buffalo
Amherst, NY 14260
sreejit@cs.buffalo.edu
pjt@cs.buffalo.edu
(TECH REPORT No. 93-42)

Abstract

An algorithmic paradigm for I_{DDQ} measurement based analysis that target all two node bridging faults (BFs) in CMOS circuits is presented. In contrast to the pessimistic criteria used in our prior work, here we use the criteria for identifying I_{DDQ} tests proposed in the literature by other researchers. Algorithms for simulation, diagnosis and I_{DDQ} subset selection of BFs in combinational circuits as well as a simulation algorithm and a subset selection algorithm for sequential circuits are presented.

In addition to showing the efficiency of the proposed algorithms our preliminary experimental results suggest that: (i) for combinational circuits, our heuristic for I_{DDQ} subset selection is considerably better than a "QUIETEST-like heuristic" for the same problem; (ii) for combinational circuits, I_{DDQ} test sets derived from randomly generated test sets are often considerably better than I_{DDQ} test sets derived from stuck-at test sets; and (iii) for sequential circuits, I_{DDQ} test sets derived from randomly generated test sets are comparable to I_{DDQ} test sets derived from stuck-at test sets.

1 Introduction

In static CMOS circuits by node we mean either the internal node of a gate or its output nodes. If two or more distinct nodes of a circuit, due to a defect, are connected we get a bridging fault (BF). In this discussion we restrict ourselves to BFs in which only two nodes are involved and henceforth use BFs to refer to such faults.

BFs can be detected using I_{DDQ} measurements[11, 18, 21]. We elaborate on this in section 2. The tests used in conjunction with I_{DDQ} measurement are known as I_{DDQ} tests. Studies have shown that BFs are highly probable in static CMOS circuits [22]. This motivates the study of the I_{DDQ} measurement based analysis problems for BFs.

By analysis of BFs we mean generation of I_{DDQ} tests for BFs (**test generation**), **fault simulation** of I_{DDQ} tests for BFs, **diagnosis** of BFs using I_{DDQ} test results, and selecting

[1]Research supported by NSF Grant No. MIP-9102509.

I_{DDQ} tests from a given set of input vectors (I_{DDQ} **subset selection**). The I_{DDQ} subset selection problem is unique to the I_{DDQ} measurement methodology and is defined as follows. Given a set of input vectors $T = \{T_1, \ldots, T_n\}$, let X be the coverage of all BFs when T is used as the I_{DDQ} test set. We are required to select a subset S of vectors from T, of minimum cardinality, s.t. if S is used as the I_{DDQ} test set then also the coverage of BFs is X[19]. The motivation for the I_{DDQ} subset selection problem is discussed in [19]. Briefly, its importance lies in the fact that I_{DDQ} measurement is a very slow process. If this is to be done for each vector in T it is going to be very time consuming. The testing process can be considerably speeded up if we can select a smaller subset i.e. S for I_{DDQ} measurement without sacrificing coverage. The smaller the size of S the faster the test time. Note that all vectors in the set T will be applied but I_{DDQ} measurements will be done only for those vectors in S.

In our analysis algorithms we need to use properties of an input vector that identifies it to be an I_{DDQ} test for a given BF. The criteria used to identify I_{DDQ} tests is based on the work in [17, 20] and is different from those in our prior work [3, 4, 5, 6, 7]. This will be elaborated on in section 2.

Our analysis targets all BFs unlike work by others where a subset of the BFs are targeted [13, 14, 12]. This subset of BFs are extracted by analyzing the layout of the circuit based on the "point defect" assumption. This extraction process tries to enumerate the most likely faults. However, it suffers from some shortcoming viz,: the layout is often not available; the process of fault extraction may itself be too time consuming because of the size of the layout, thus negating the advantage of using a small fault list as observed in [4]; and it is difficult to estimate the size of the "points" used in the extraction process.

In this paper we show that by using a clever representation of all BFs [6, 7], and a judicious combination of four simple rules (based on logic simulation alone) to manipulate the fault list, we can derive very effective algorithms for fault simulation, diagnosis and I_{DDQ} subset selection problems. The faultlist representation and the "rules" are presented in section 3. In sections 4,5 and 6 we present the algorithms respectively for simulation, diagnosis and I_{DDQ} subset selection. Experimental evaluation of these algorithms are presented in section 7 where we also present some important observations.

The observations, very briefly, are: (i) the proposed heuristic for I_{DDQ} subset selection for combinational circuit is considerably "better" than a "QUIETEST-like heuristic" for the same problem; (ii) for combinational circuits, I_{DDQ} test sets derived from randomly generated test sets are very often "better" than I_{DDQ} test sets derived from stuck-at tests; (iii) for sequential circuits, I_{DDQ} test sets derived from randomly generated test sets are comparable to I_{DDQ} test sets derived from stuck-at tests; and (iv) targeting all two node BFs for I_{DDQ} measurement based analysis is computationally feasible. The first observation is important because considerably smaller I_{DDQ} subsets are selected using the proposed heuristic thus reducing the total test time. The second observation is somewhat surprising. This observation is important because it cautions the use of test sets for other fault models to derive I_{DDQ} tests for BFs. In addition it suggests that I_{DDQ} measurement may not have as severe an effect on the test time when used with LFSR based random testing as it would with a deterministically generated test set. We note that stuck-at test sets performed better for diagnosis. However, we caution our readers that this observation is based just on our experiments with one stuck-at test set and only one randomly generated test set.

Previous work on I_{DDQ} measurement based fault simulation [3, 4, 10] uses the notion of "Equivalence classes". Algorithms based on this idea can only be used for BFs between two lines in the circuit that are outputs of a gate and falls apart when internal nodes are included. In addition these algorithms were only for combinational circuits. We are unaware of any fault simulation algorithms, even for a subset of BFs, in sequential circuits. Here we present an algorithm that targets all BFs, including those that involve internal nodes, in combinational as well as sequential circuits.

I_{DDQ} measurement based diagnosis of BFs is discussed in [2, 5, 6, 7]. The algorithm we present is a modification of the algorithm in [5] to include BFs involving internal nodes of the circuit. It is a simplification of the diagnosis algorithm in [7]. It is presented here primarily for the sake of completeness.

To the best of our knowledge the I_{DDQ} subset selection problem for BFs was never addressed in the literature. In [19] this problem was addressed for leakage faults in sequential circuits. We present algorithms for this problem that targets all BFs, including those that involve internal nodes, in combinational as well as sequential circuits.

2 Iddq Tests for BFs

Let $c(d)$ be the output of gate G_1 (G_2) in circuit of Fig. 1 ; F be a fault such that c, d are shorted; $v(l)$ be the value at node l on application of vector v to the good circuit and $v'(l)$ be the value at node l on application of v to the faulty circuit.

Fig. 2 shows the switch-level description of the circuit in Fig 1. Nodes $\{a, b, c, d\}$ are the same as those shown in the gate-level description. Node $\{e\}$ is an internal node of the circuit. The complete set of nodes in the circuit is given by $\{a, b, c, d, e\}$.

As shown in Fig. 2, in the good circuit on application of $v = < a = 1, b = 0 >$, $\exists$ a conducting path P_1 in G_1 from V_{dd} to c and a conducting path P_2 in G_2 from d to Gnd. In the presence of $F = < c, d >$ paths P_1, P_2 will be activated in the faulty circuit and there will be a conducting path from V_{dd} to Gnd. This will result in a large I_{DDQ} compared with the fault free I_{DDQ}. This large I_{DDQ} is used to signal the presence of a fault.

Note that on application of $v = < a = 0, b = 0 >$, $v(e) = f$(floating). In presence of $F = < e, d >$ there will be no conducting path from e to Gnd, hence no appreciable I_{DDQ}. Thus v is not a test for fault F.

We are going to use the following criteria from [17] to identify I_{DDQ} tests for a BF between f_1, f_2. An input vector v is an I_{DDQ} test for $< f_1, f_2 >$ iff $v(f_1) \neq v(f_2)$ and $v(f_1) \neq f$ and $v(f_2) \neq f$.

Note that this condition differs from the criteria used in [3, 4, 6, 7], where a BF is classified as a **nonfeedback single bridging fault(NFBF)** or a **feedback single bridging fault (FBF)** as follows. Let F be an BF $< f_1, f_2 >$. Then F is a NFBF iff there does not exist a path in C either from f_1 to f_2 or from f_2 to f_1. If a BF is not an NFBF then it is a FBF. In Fig. 1 $< a, c >$ is a FBF and $< d, c >$ is NFBF.

For NFBF $< x, y >$ the same criteria for v to be an I_{DDQ} test as in [17] was used. However, for FBF $< x, y >$ in addition to $v(x) \neq v(y)$ an additional condition was imposed. This is explained with the help of an example that uses Fig. 1. Consider the fault $< a, c >$. Let $v_1 =< a = 0, b = 1 >$ and $v_2 =< a = 0, b = 0 >$. In the fault free case , for both v_1, v_2 $a = 0$ and $c = 1$. According to [17] both qualify as an I_{DDQ} test for $< a, c >$. However, according to [3, 4, 6, 7] v_1 does not but v_2 qualifies as a test for $< a, c >$. v_1 was rejected because it leads to oscillation. The oscillating power supply current level may be detected as an elevated I_{DDQ} current. This however depends on the frequency of oscillation and the capabilities of the current sensor. Using a "pessimistic approach", in [3, 4, 6, 7] inputs leading to oscillation are rejected as I_{DDQ} tests. As stated above, in this paper we will use the more "optimistic approach".

3 Fault List and Fault Dropping

Here we discuss our representation of sets of BFs and some general rules for dropping faults from faultlists so represented. In later sections we will see how a judicious combination of these rules result in efficient algorithms for all I_{DDQ}-based analysis problems of interest.

3.1 FaultLists

In the algorithm BFs are represented using Set of Ordered Pairs of Sets (SOPS) of the form $\{< A_1, B_1 >, < A_2, B_2 >, \ldots, < A_k, B_k >\}$ where A_i is a set of nodes in the circuit and B_i can be either a set of nodes in the circuit or the special symbol Ψ.

If B_i is a set of the nodes then the ordered pair $< A_i, B_i >$ denotes the set of BFs $< x, y >$ where $x \in A_i$ and $y \in B_i$. If B_i is the special symbol Ψ then $< A_i, \Psi >$ denotes the set of BFs $< x, y >$ where $x, y \in A_i$. For example, the SOPS L = $\{< \{2, 3\}, \{7, 8\} >, < \{9, 17, 18\}, \Psi >\}$ represents the following set of BFs: $\{< 2, 7 >, < 2, 8 >, < 3, 7 >, < 3, 8 >, < 9, 17 >, < 9, 18 >, < 17, 18 >\}$.

3.2 Rules for Fault Dropping

Let A be a set of nodes. On application of T_i, every node in A can have the value $0, 1$ or f. This can be derived by logic simulation. The following notation is used in subsequent discussions: $A^0 = \{n \mid T_i(n) = 0 \text{ and } n \in A\}$; $A^1 = \{n \mid T_i(n) = 1 \text{ and } n \in A\}$; $A^f = \{n \mid T_i(n) = f \text{ and } n \in A\}$.

Our algorithms will process input vectors $T_1, T_2, \ldots, T_n$. After each vector is processed, (depending on the algorithm) faults will be dropped from the faultlist. Let L_{i-1} be the SOPS representing the faultlist after processing the input vectors $\{T_1 \cdots T_{i-1}\}$. Next assume $\{T_i\}$ is processed and faults from L_{i-1} are dropped. The new set of faults is represented by the SOPS L_i. The rules for dropping faults are described below:

3.2.1 Dropping faults detected by T_i

For each ordered pair $< A, B > \in L_{i-1}$ faults detected by T_i are dropped by replacing $< A, B >$ with a new SOPS as described below. We have two cases (given by Rule 1 and Rule 2).

- **Rule 1**. Let $< A, B >, B = \Psi$. The following SOPS replaces $< A, B >$: $\{< A^0, \Psi >, < A^1, \Psi >, < A^f, \Psi >, < A^0, A^f >, < A^1, A^f >\}$. Rule 1 should drop BFs detected by T_i. Thus, according to section 2, all BFs $< x, y > \in < A^0, A^1 >$ should be dropped. Note that this is exactly the set of faults dropped by Rule 1.

- **Rule 2**. Let $< A, B >, B \neq \Psi$. The following SOPS replaces $< A, B >$: $\{< A^0, B^0 \cup B^f >, < A^1, B^1 \cup B^f >, < A^f, B >\}$.

  Rule 2 should drop BFs detected by T_i. Thus, according to section 2, all BFs $< x, y > \in < A^0, B^1 > \, or < A^1, B^0 >$ should be dropped. Note that this exactly the set of faults dropped by Rule2.

If set of bridging faults represented by the ordered pair $< A, B > = \Phi$ then it is not added to the SOPS L_i

3.2.2 Dropping faults not detected by T_i

On application of T_i, for an ordered pair $< A, B >$, we again have two Rules to drop faults not detected by T_i.

- **Rule 3**. $< A, B >, B = \Psi$. The following SOPS replaces $< A, B >$: $\{< A^0, A^1 >\}$.

  Rule 3 should drop faults not detected by T_i. Thus, according to section 2, all BFs $< x, y > \notin < A^0, A^1 >$ should be dropped. Note that this is exactly the set of faults dropped by Rule 3.

- **Rule 4**. $< A, B >, B \neq \Psi$. The following SOPS replaces $< A, B >$: $\{< A^1, B^0 >, < A^0, B^1 >\}$.

Rule 4 should drop faults not detected by T_i. Thus, according to section 2, all BFs $< x, y > \notin < A^0, B^1 > or < A^1, B^0 >$ should be dropped. Note that this is exactly the set of faults dropped by Rule 4.

4 Fault Simulation

Fault Simulation. For a given circuit C and test set T, compute the percentage of the BF's in C detected using T and I_{DDQ} measurement.

The following information is assumed to be available: (i) The gate-level description of the fault free circuit; and (ii) the definition of the input vectors in T. We first state the algorithm for combinational circuits and then present an example in sections 4.1 and 4.2 respectively. In sections 4.3 and 4.4, we give the algorithm for sequential circuits and an example. Experimental results are presented in Section 7.

4.1 Algorithm COMB_FaultSim

Notations.
N is the number of input test vectors.
$T = \{T_1 \cdots T_N\}$ is the test set.
C is the given circuit.
X is the set of nodes in C.
m is the number of nodes in C.
$T_i(y)$ is the value at node y on application of T_i.
L_i is the SOPS after processing vectors $T_1 \cdots T_i$.

- Step 1. $L_0 = \{< X, \Psi >\}$, the set of all BFs.

- Step 2. For each $T_i, 1 \leq i \leq N$ do Step 3.

- Step 3. Let $L_{i-1} = \{< A_1, B_1 >, \cdots, < A_q, B_q >\}$. For $1 \leq j \leq q$ process $< A_j, B_j >$ to form new ordered pairs as follows. These new ordered pairs are added to L_i and $< A_j, B_j >$ deleted from L_{i-1}.

  - Compute using logic simulation, $A_j^0(T_i)$, $A_j^1(T_i)$, $A_j^f(T_i)$. If $B \neq \Psi$ compute $B_j^0(T_i)$, $B_j^1(T_i)$, $B_j^f(T_i)$.

– If $B = \Psi (B \neq \Psi)$ apply Rule 1(Rule 2) to compute the set of new ordered pairs to replace $< A_j, B_j >$.

- Step 4. Let $L_N = \{< C_1, D_1 >, \cdots, < C_p, D_p >\}$. Note that for every fault in L_N, $\exists$ only one j s.t. $f \in < C_j, D_j >$. Let S be the union of the set of bridging faults represented by ordered pairs in L_N. Then $\mid S \mid = \sum_{j=1}^{p} Count(< C_j, D_j >)$ where $Count(< C_j, D_j >) = \mid C_j \mid \cdot \mid D_j \mid$ if $D_j \neq \Psi$ else $Count(< C_j, D_j >) = \frac{|C_j| \cdot |(C_j - 1)|}{2}$. Then, coverage $= 1 - \frac{|S|}{Tot} * 100$ where $Tot = \frac{m \cdot (m-1)}{2}$.

4.2 Example

Switch-level circuit of Fig. 4 is shown in Fig. 5. The test set $T = \{T_1, T_2\}$ is defined in Table 1.

Initially, $L_0 = \{< \{0, 1, 2, 3, 4, 5, 6, 7\}, \Psi >\}$. First, T_1 is processed. Logic simulation implies $A^0 = \{1, 2, 3, 5, 6\}$, $A^1 = \{0, 4\}$, $A^f = \{7\}$. Since $B = \Psi$, Rule 1 is used to obtain $L_1 = \{< \{1, 2, 3, 5, 6\}, \Psi >, < \{0, 4\}, \Psi >, < \{1, 2, 3, 5, 6\}, \{7\} >, < \{0, 4\}, \{7\} >\}$.

Next T_2 is processed. Each ordered pair in L_1 is replaced as follows.

1. $< A_1, B_1 > = < \{1, 2, 3, 5, 6\}, \Psi >$. Logic simulation implies $A_1^0 = \{2, 3\}$, $A_1^1 = \{1, 5, 6\}$, $A_1^f = \{7\}$. Since $B_1 = \Psi$, Rule 1 is used. Thus, $< A_1, B_1 >$ is replaced by: $\{< \{2, 3\}, \Psi >, < \{1, 5, 6\}, \Psi >\}$.

2. $< A_2, B_2 > = < \{0, 4\}, \Psi >$. Logic simulation implies $A_2^0 = \{0\}$, $A_2^1 = \{4\}$, $A_2^f = \Phi$. Use of Rule 1 implies that the SOPS replacing $< A_2, B_2 >$ is Φ.

3. $< A_3, B_3 > = < \{1, 2, 3, 5, 6\}, \{7\} >$. Logic Simulation implies $A_3^0 = \{0, 3\}$, $A_3^1 = \{1, 5, 6\}$, $A_3^f = \Phi$, $B_3^0 = \Phi$, $B_3^1 = \Phi$, $B_3^f = \{7\}$. Since $B_3 \neq \Psi$ use of Rule 2 implies that the SOPS replacing $< A_3, B_3 >$ is : $\{< \{2, 3\}, \{7\} >, < \{1, 5, 6\}, \{7\} >\}$.

4. $< A_4, B_4 > = < \{0, 4\}, \{7\} >$. Logic simulation implies $A_4^0 = \{0\}$, $A_4^1 = \{4\}$, $A_4^f = \Phi$, $B_4^0 = \Phi$, $B_4^1 = \Phi$, $B_4^f = \{7\}$. Use of Rule 2 implies that $< A_4, B_4 >$ is replaced by the following SOPS: $\{< \{0\}, \{7\} >, < \{4\}, \{7\} >\}$.

From (1)-(4), $L_2 = \{< \{2, 3\}, \Psi >, < \{1, 5, 6\}, \Psi >, < \{0, 2, 3\}, \{7\} >, < \{1, 5, 6\}, \{7\} >, < \{0\}, \{7\} >, < \{4\}, \{7\} >\}$.

In Step 4 of the algorithm, from L_2 we obtain count of undetected BFs to be 1+3+3+3+2+3+1+1 = 17. Thus the coverage = (11/28)*100 = 39%.

4.3 Algorithm SEQ_FaultSim

Notations.
N is the number of input test vectors.

$T = \{T_1 \cdots T_N\}$ is the test set.
C is the given circuit.
X is the set of nodes in C.
m is the number of nodes in C.
$T_i(y)$ is the value at node y on application of T_i.
L_i is the SOPS after processing vectors $T_1 \cdots T_i$.
N_i is the fault free Next State Vector obtained on application of T_i. To start with, this is assumed to be the all zero state (Global Reset)

- Step 1. $L_0 = \{< X, \Psi >\}$, the set of all BFs.

- Step 2. For each $T_i, 1 \leq i \leq N$ do Step 3.

- Step 3. Let $L_{i-1} = \{< A_1, B_1 >, \cdots, < A_q, B_q >\}$. For $1 \leq j \leq q$ process $< A_j, B_j >$ to form new ordered pairs as follows. These new ordered pairs are added to L_i and $< A_j, B_j >$ deleted from L_{i-1}.

  - Compute using logic simulation, N_i (the fault free next state vector), $A_j^0(T_i \cup N_{i-1})$, $A_j^1(T_i \cup N_{i-1})$, $A_j^f(T_i \cup N_{i-1})$. If $B \neq \Psi$ compute $B_j^0(T_i \cup N_{i-1})$, $B_j^1(T_i \cup N_{i-1})$, $B_j^f(T_i \cup N_{i-1})$.

  - If $B = \Psi (B \neq \Psi)$ apply Rule 1(Rule 2) to compute the set of new ordered pairs to replace $< A_j, B_j >$.

- Step 4. Let $L_N = \{< C_1, D_1 >, \cdots, < C_p, D_p >\}$. Note that for every fault in $L_N, \exists$ only one j s.t. $f \in < C_j, D_j >$. Let S be the union of the set of bridging faults represented by ordered pairs in L_N. Then $\mid S \mid = \sum_{j=1}^{p} Count(< C_j, D_j >)$ where $Count(< C_j, D_j >) = \mid C_j \mid \cdot \mid D_j \mid$ if $D_j \neq \Psi$ else $Count(< C_j, D_j >) = \frac{|C_j| \cdot |(C_j - 1)|}{2}$. Then, coverage $= 1 - \frac{|S|}{Tot} * 100$ where $Tot = \frac{m \cdot (m-1)}{2}$.

The fault simulation algorithm for sequential circuits is obtained by modifying the fault simulation algorithm for combinational circuits. These modifications are adequate for the following reasons. Consider the fault list F_{i-1} , input vector T_i and circuit C. Assume that the present state P_i is the present state obtained by fault free logic simulation of $T_1, \cdots, T_{i-1}$. We assume that this is same as the present state obtained by simulating $T_1, \cdots, T_{i-1}$ in the presence of BFs F_{i-1}. After processing T_i all faults $< x, y >$ s.t. $T_i(x) \neq T_i(y)$ are dropped from F_{i-1} to derive F_i.

Therefore, under the above assumption all faults $< a, b >$ in F_i are such that the present state P_{i+1} obtained by simulating $T_1, \cdots, T_i$ in the presence of the BFs in F_i is the same as the present state obtained by fault free logic simulation of $T_1, \cdots, T_i$.

To complete an informal presentation of the inductive proof we note that we are assuming that all flip/flops can be initially set to zero, i.e. the all zero fault free initial state, in presence of any of the BFs.

4.4 Example

For this example we use the sequential circuit in Fig. 3. Switch-level circuit is shown in Fig. 5. The test set $T = \{T_1, T_2\}$ is defined in Table 2.

Initially,$N_0 =< 0 >$, $L_0 = \{< \{0,1,2,3,4,5,6,7\}, \Psi >\}$. First, T_1 is processed. Logic simulation implies $N_1 =< 1 >$, $A^0 = \{1,2,3,5,6\}$, $A^1 = \{0,4\}$, $A^f = \{7\}$. Since $B = \Psi$, Rule 1 is used to obtain $L_1 = \{< \{1,2,3,5,6\}, \Psi >, < \{0,4\}, \Psi >, < \{1,2,3,5,6\}, \{7\} >, < \{0,4\}, \{7\} >\}$.

Next T_2 is processed. From Logic simulation we get $N_2 =< 1 >$. Each ordered pair in L_1 is replaced as follows.

1. $< A_1, B_1 >=< \{1,2,3,5,6\}, \Psi >$. Logic simulation implies $A_1^0 = \{2,3\}$, $A_1^1 = \{1,5,6\}$, $A_1^f = \{7\}$. Since $B_1 = \Psi$, Rule 1 is used. Thus, $< A_1, B_1 >$ is replaced by: $\{< \{2,3\}, \Psi >, < \{1,5,6\}, \Psi >\}$.

2. $< A_2, B_2 > = < \{0,4\}, \Psi >$. Logic simulation implies $A_2^0 = \{0\}, A_2^1 = \{4\}, A_2^f = \Phi$. Use of Rule 1 implies that the SOPS replacing $< A_2, B_2 >$ is Φ.

3. $< A_3, B_3 > = < \{1,2,3,5,6\}, \{7\} >$. Logic Simulation implies $A_3^0 = \{0,3\}, A_3^1 = \{1,5,6\}, A_3^f = \Phi, B_3^0 = \Phi, B_3^1 = \Phi, B_3^f = \{7\}$. Since $B_3 \neq \Psi$ use of Rule 2 implies that the SOPS replacing $< A_3, B_3 >$ is : $\{< \{2,3\}, \{7\} >, < \{1,5,6\}, \{7\} >\}$.

4. $< A_4, B_4 >=< \{0,4\}, \{7\} >$. Logic simulation implies $A_4^0 = \{0\}, A_4^1 = \{4\}, A_4^f = \Phi, B_4^0 = \Phi, B_4^1 = \Phi, B_4^f = \{7\}$. Use of Rule 2 implies that $< A_4, B_4 >$ is replaced by the following SOPS: $\{< \{0\}, \{7\} >, < \{4\}, \{7\} >\}$.

From (1)-(4), $L_2 = \{< \{2,3\}, \Psi >, < \{1,5,6\}, \Psi >, < \{0,2,3\}, \{7\} >, < \{1,5,6\}, \{7\} >, < \{0\}, \{7\} >, < \{4\}, \{7\} >\}$.

In Step 4 of the algorithm, from L_2 we obtain count of undetected BFs to be 1+3+3+3+2+3+1+1 = 17. Thus the coverage = (11/28)*100 = 39%.

5 Diagnosis

Diagnosis Problem. Given a combinational circuit C , detected by an I_{DDQ} test set T to be faulty, identify the fault in C. We assume that the following information is available: (i)the gate-level description of the fault free circuit; (ii) the definition of the input vectors in T; (iii) for each input vector T_i in T if C gave a faulty response or not i.e. whether an abnormally high or a normal I_{DDQ} was measured. Further more, we assume that C has **only one** BF. The objective is to determine what this fault is.

Note that the logic response of the circuit is not used. We first state the algorithm and then illustrate with an example.

5.1 Algorithm DIAG

Notation. In addition to the notations used in section 4, we use the following.
$R = \{r_1, \cdots, r_N\}$ is the response vector where $\forall i, r_i \in \{0,1\}$ and $r_i = 0(1)$ if there was not (there was) a large I_{DDQ} on application of T_i.

- Step 1. $L_0 = \{< X, \Psi >\}$, is the set of all BFs.
- Step 2. For each T_i, $1 \leq i \leq N$ do Step 3.
- Step 3. Let $L_{i-1} = \{< A_1, B_1 >, \cdots, < A_q, B_q >\}$. For $1 \leq j \leq q$ process $< A_j, B_j >$ to form new ordered pairs as below. These new ordered pairs are added to L_i and $< A_j, B_j >$ and $< A_j, B_j >$ is deleted from L_{i-1}.
  - Compute using logic simulation, $A_j^0(T_i)$, $A_j^1(T_i)$, $A_j^f(T_i)$. If $B \neq \Psi$ compute $B_j^0(T_i)$, $B_j^1(T_i)$, $B_j^f(T_i)$.
    We have four cases. For each of these the rules to be used are as follows:
    1. if $B_j = \Psi$ and $r_i = 0$, use Rule 1.
    2. if $B_j = \Psi$ and $r_i = 1$, use Rule 3.
    3. if $B_j \neq \Psi$ and $r_i = 0$, use Rule 2.
    4. if $B_j \neq \Psi$ and $r_i = 1$, use Rule 4.
- Step 4: Let $L_N = \{< C_1, D_1 >, \cdots, < C_p, D_p >\}$. The union of the set of BFs represented by these ordered pairs is the possible set of BFs. The number of undetected BFs is given by $\sum_{j=1}^{p} Count(< C_j, D_j >)$, where $Count(< C_j, D_j >$ is same as defined in Section 4.1.

5.2 Example

The circuit of Figure 4, Test set $T = \{T_1, T_2\}$ of Table 1 and the response vector $R = \{0, 1\}$ is used for this example.

Initially $L_0 = \{< \{0,1,2,3,4,5,6,7\}, \Psi >\}$. Logic simulation of T_1 implies, $A^0 = \{1,2,3,5,6\}$, $A^1 = \{0,4\}$, $A^f = \{7\}$. Since $B = \Psi$ and $r_1 = 0$, Rule 1 is used. Thus $L_1 = \{< \{1,2,3,5,6\}, \Psi >, < \{0,4\}, \Psi >, < \{1,2,3,5,6\}, \{7\} >, < \{0,4\}, \{7\} >\}$.

Next T_2 is processed and faults from L_1 are dropped as stated below for each ordered pair.

1. $< A_1, B_1 >=< \{1,2,3,5,6\}, \Psi >$. Logic simulation implies $A_1^0 = \{2,3\}$, $A_1^1 = \{1,5,6\}$, $A_1^f = \Phi$. Since $B_1 = \Psi$ and $r_2 = 1$, Rule 3 is used. Thus $\{< \{2,3\}, \{1,5,6\} >\}$ replaces $< A_1, B_1 >$.

2. $< A_2, B_2 >=< \{0,4\}, \Psi >$. Logic simulation gives $A_2^0 = \{0\}, A_2^1 = \{1\}, A_2^f = \Phi$. Since $B_2 = \Psi$ and $r_2 = 1$, Rule 3 is used. Thus $\{< \{0\}, \{1\} >\}$ replaces $< A_2, B_2 >$.

3. $< A_3, B_3 >=< \{1,2,3,5,6\}, \{7\} >$. Logic simulation implies $A_3^0 = \{2,3\}, A_3^1 = \{1,5,6\}, A_3^f = \Phi, B_3^0 = \Phi, B_3^1 = \Phi, B_3^f = \{7\}$. Since $B_3 = \Psi$ and $r_2 = 1$, Rule 4 is used. Thus, $< A_3, B_3 >$ is replaced by the empty set.

4. $< A_4, B_4 >=< \{0,4\}, \{7\} >$. Logic simulation implies $A_4^0 = \{0\}, A_4^1 = \{4\}, A_4^f = \Phi, B_4^0 = \Phi, B_4^1 = \Phi, B_4^f = \{7\}$. Here we use Rule 4 because $B_4 = \Psi$ and $r_2 = 1$ to obtain Φ.

From 1-4, $L_2 = \{< \{2,3\}, \{1,5,6\} >, < \{0\}, \{1\} >\}$. Since we have iterated through all the test vectors Step 4 is executed. The total number of possible BFs are 6+1 = 7. Note that the number of possible BFs reduces from 28 to 7.

6 Selecting I_{DDQ} Test Patterns

The I_{DDQ} subset selection problem was defined in section 1. Here, we present two heuristics. The first one is adapted from [19], where it was used for subset selection of a test set used to detect leakage faults. We show how this heuristic can be used for the subset selection problem in combinational as well as sequential circuits. The second one is an adaptation of the heuristic from [9] for the set cover problem and so far we can use it for combinational circuits only.

6.1 COMB_QUIETEST Heuristic

Notation. In addition to the notations used in sections 4 and 5, we use the following:

$S_i \subseteq \{T_1, \cdots, T_i\}$ is the subset selected so far for I_{DDQ} measurement.

- Step 1. $L_0 = \{< X, \Psi >\}$ the set of all BFs, $S_0 = \Phi$.
- Step 2. For each $T_i, 1 \leq i \leq N$ do Steps 3 through 5.
- Step 3. Let $L_{i-1} = \{< A_1, B_1 >, \cdots, < A_q, B_q >\}$. Q_{i-1} = duplication of L_{i-1}. For $1 \leq j \leq q$ process $< A_j, B_j >$ to form new ordered pairs as follows. These new ordered pairs are added to Q_i and $< A_j, B_j >$ deleted from Q_{i-1}.

- Compute using logic simulation, $A_j^0(T_i), A_j^1(T_i), A_j^f(T_i)$.
  If $B \neq \Psi$ compute $B_j^0(T_i), B_j^1(T_i), B_j^f(T_i)$.
  If $B = \Psi (B \neq \Psi)$ apply Rule 1(Rule 2) to compute the set of new ordered pairs to replace $< A_j, B_j >$.

- Step 4: Compute C_i, the number of BFs in Q_i.

- Step 5: if $C_i <$ the number of BFs in L_{i-1} then $S_i = S_{i-1} \cup \{T_i\}$, $L_i = Q_i$ else $L_i = L_{i-1}$. Free Q_i.

The resulting set S_N is the selected subset of the test set. Note that here we are greedily picking the vectors. If the next vector processed improves coverage we select it.

6.2 Example

We use the circuit in Fig. 4 and the initial test set $T = \{T_1, T_2, T_3\}$ in Table 2. Initially, $C_0 = 0$, $L_0 = \{< \{0,1,2,3,4,5,6,7\}, \Psi >\}$ and $S_0 = \Phi$. Steps 3-5 are iterated for each test vector $T_i, 1 \leq i \leq 3$.

Applying T_1 to L_0 we obtain :
$Q_1 = \{< \{1,2,3,5,6\}, \Psi >, < \{0,4\}, \Psi >, < \{1,2,3,5,6\}, \{7\} >, < \{0,4\}, \{7\} >\}$ and $C_1 = 12$. Since C_1 is less than the number of BFs in L_0 , we have : $S_1 = \{T_1\}$ and $L_1 = Q_1$.

Now, process T_2. We get,
$Q_2 = \{< \{2,3\}, \Psi >, < \{1,5,6\}, \Psi >, < \{2,3\}, \{7\} >, < \{1,5,6\}, \{7\} >, < \{0\}, \{7\} >, < \{1\}, \{7\} >\}$ and $C_2 = 16$. Since C_2 is less than the number of BFs in L_1 we have: $S_2 = \{T_1, T_2\}$ and $L_2 = Q_2$.

Next T_3 is processed. $Q_3 = \{< \{2,3\}, \Psi >, < \{5,6\}, \Psi >, < \{5,6\}, \{1\} >, < \{2,3\}, \{7\} >, < \{1\}, \{7\} >, < \{5,6\}, \{7\} >, < \{0\}, \{7\} >, < \{4\}, \{7\} >\}$ and $C_3 = 16$

(From Step 5, since $C_3 \not< $ the number of BFs in L_2, we have $S_3 = S_2 = \{T_1, T_2\}$ and $L_3 = L_2$.)

This example illustrates the fact that instead of using I_{DDQ} measurement on T_1, T_2, and T_3 we only need to do so for T_1 and T_2 to get the same coverage for BFs.

6.3 SEQ_QUIETEST Heuristic

Notation. In addition to the notations used in sections 4 and 5, we use the following:

$S_i \subseteq \{T_1, \cdots, T_i\}$ is the subset selected so far for I_{DDQ} measurement.

- Step 1. $L_0 = \{< X, \Psi >\}$ the set of all BFs, $S_0 = \Phi$.

- Step 2. For each $T_i, 1 \leq i \leq N$ do Steps 3 through 5.
- Step 3. Let $L_{i-1} = \{< A_1, B_1 >, \cdots, < A_q, B_q >\}$. Q_{i-1} = duplication of L_{i-1}. For $1 \leq j \leq q$ process $< A_j, B_j >$ to form new ordered pairs as follows. These new ordered pairs are added to Q_i and $< A_j, B_j >$ deleted from Q_{i-1}.
  - Compute using logic simulation, N_i (the fault free next state vector), $A_j^0(T_i \cup N_{i-1}), A_j^1(T_i \cup N_{i-1}), A_j^f(T_i \cup N_{i-1})$.
    If $B \neq \Psi$ compute $B_j^0(T_i \cup N_{i-1}), B_j^1(T_i \cup N_{i-1}), B_j^f(T_i \cup N_{i-1})$.
    If $B = \Psi (B \neq \Psi)$ apply Rule 1(Rule 2) to compute the set of new ordered pairs to replace $< A_j, B_j >$.
- Step 4: Compute C_i, the number of BFs in Q_i.
- Step 5: if $C_i <$ the number of BFs in L_{i-1} then $S_i = S_{i-1} \cup \{T_i\}$, $L_i = Q_i$ else $L_i = L_{i-1}$, free Q_i.

The I_{DDQ} subset selection algorithm for sequential circuits is a minor modification of a similar algorithm for I_{DDQ} subset selection in combinational circuits. The modifications are similar to the one used to derive the fault simulation algorithm for BFs in sequential circuits from the fault simulation algorithm for combinational circuits.

6.4 Example

We use the circuit in Fig. 3 and the initial test set $T = \{T_1, T_2, T_3\}$ in Table 4. Initially, $N_0 =< 0 >$, $C_0 = 0$, $L_0 = \{< \{0, 1, 2, 3, 4, 5, 6, 7\}, \Psi >\}$ and $S_0 = \Phi$. Steps 3-5 are iterated for each test vector $T_i, 1 \leq i \leq 3$.

Applying T_1 we obtain $N_1 =< 1 >$ and from L_0 we obtain :
$Q_1 = \{< \{1, 2, 3, 5, 6\}, \Psi >, < \{0, 4\}, \Psi >, < \{1, 2, 3, 5, 6\}, \{7\} >, < \{0, 4\}, \{7\} >\}$ and $C_1 = 12$. Since C_1 is less than the number of BFs in L_0 , we have : $S_1 = \{T_1\}$ and $L_1 = Q_1$.

Now, process T_2. We get $N_2 =< 1 >$ and,
$Q_2 = \{< \{2, 3\}, \Psi >, < \{1, 5, 6\}, \Psi >, < \{2, 3\}, \{7\} >, < \{1, 5, 6\}, \{7\} >, < \{0\}, \{7\} >, < \{1\}, \{7\} >\}$ and $C_2 = 16$. Since C_2 is less than the number of BFs in L_1 we have: $S_2 = \{T_1, T_2\}$ and $L_2 = Q_2$.

Next T_3 is processed. $N_3 =< 0 >$ and $Q_3 = \{< \{2, 3\}, \Psi >, < \{5, 6\}, \Psi >, < \{5, 6\}, \{1\} >, < \{2, 3\}, \{7\} >, < \{1\}, \{7\} >, < \{5, 6\}, \{7\} >, < \{0\}, \{7\} >, < \{4\}, \{7\} >\}$ and $C_3 = 16$

(From Step 5, since $C_3 \not< $ the number of BFs in L_2, we have $S_3 = S_2 = \{T_1, T_2\}$ and $L_3 = L_2$.)

This example illustrates the fact that instead of using I_{DDQ} measurement on T_1, T_2, and T_3 we only need to do so for T_1 and T_2 to get the same coverage for BFs.

6.5 SET COVER Heuristic

Let F $= \{S_1, S_2, \ldots, S_k\}$ be a collection of finite sets such that $U = \bigcup_{i=1}^{k} S_i$. Then $F' \subseteq F$ is said to *cover* F iff $\bigcup_{s \in F'} S = U$.
SET COVER: Given F, compute a cover F' of F of minimum cardinality.

A well-known heuristic for SET COVER is given in [9]. The greedy approach of section 6.1 may give bad results and is very sensitive to the order in which the vectors are processed. Our next heuristic is motivated by the heuristic for SET COVER from [9]. This heuristic is adapted to our needs as stated below. As of now, we can use it only for combinational circuits.

Notations: Given below are the notations used in addition to those defined in earlier sections.

T^i is the set of unselected test vectors left after i iterations of the algorithm.

R_i is the set of BFs obtained by dropping faults detected by test vectors in S_i after i iterations.

SET COVER Heuristic

- Step 1. $T^0 = T; S_0 = \Phi; R_0 = \{< X, \Psi >\}$.
- Step 2. Repeat Steps 3 and 4 until $\forall T_j \in T^{i-1}; T_j$ does not drop any faults from R_{i-1}.
- Step 3. As in Step 3 of COMB_QUIETEST heuristic, simulate all the vectors in T^{i-1} and select T_j s.t. T_j drops the maximum number of faults from R_{i-1}.
- Step 4. $T^i = T^{i-1} - \{T_j\}$; $S_i = S_{i-1} \cup \{T_j\}$; R_i is obtained by simulating T_j and dropping faults from R_{i-1}.

Step 3 is the step that selects the next vector to be included in the subset and is very expensive. It evaluates all the remaining vectors and adds the best one to the subset. We give below an example to illustrate the algorithm. Experimental results are given in section 7.

6.6 Example

The circuit in Fig 4 and $T = \{T_1, T_2, T_3\}$ of Table 2 is used to illustrate the SETCOVER heuristic. Let F_j^i = Number of faults dropped from L_j by test T_i and L_j^i = the possible SOPS replacing L_j on application of T_i. Initially, $S_0 = \Phi$ and $L_0 = \{< \{0, 1, 2, 3, 4, 5, 6, 7\}, \Psi >\}$. Each T_i is evaluated as detailed below.

Applying T_1 to L_0 we obtain $L_0^1 = \{< \{1,2,3,5,6\}, \Psi >, < \{0,4\}, \Psi >, < \{1,2,3,5,6\}, \{7\} >, < \{0,4\}, \{7\} >\}$ and $F_1^1 = 10$. Applying T_2 to L_0 we obtain $L_0^2 = \{< \{0,2,3\}, \Psi >, < \{1,4,5,6\}, \Psi >, < \{0,2,3\}, \{7\} >, < \{1,4,5,6\}, \{7\} >\}$ and $F_1^2 = 12$. Applying T_3 to L_0 we obtain $L_0^3 = \{< \{2,3\}, \Psi >, < \{4,1,0\}, \Psi >, < \{5,6,7\}, \Psi >, < \{2,3\}, \{5,6,7\} >, < \{4,1,0\}, \{5,6,7\} >\}$ and $F_1^3 = 6$. Since F_1^2 is the maximum, T_2 drops the maximum number of faults and hence $T^1 = T^0 - \{T_2\}$; $S_1 = \{T_2\}$; $L_1 = L_0^2$. This completes one iteration.

Now we execute Step 3 with T^1 as the test set and L_1 as the initial SOPS. Applying T_1 to L_1 we obtain $L_1^1 = \{< \{2,3\}, \Psi >, < \{1,5,6\}, \Psi >, < \{2,3\}, \{7\} >, < \{0\}, \{7\} >, < \{1,5,6\}, \{7\} >, < \{4\}, \{7\} >\}$ and $F_1^1 = 5$.

Applying T_3 to L_1 we obtain $L_1^3 = \{< \{2,3\}, \Psi >, < \{1,4\}, \Psi >, < \{5,6\}, \Psi >, < \{1,4\}, \{5,6\} >, < \{2,3\}, \{7\} >, < \{0\}, \{7\} >, < \{1,4\}, \{7\} >, < \{5,6\}, \{7\} >\}$ and $F_1^3 = 2$.

Since F_1^1 is the maximum, T_1 drops the maximum number of faults and hence $T^2 = T^1 - \{T_1\}$; $S_2 = \{T_1, T_2\}$; $L_2 = L_1^1$.

Now we execute Step 3 with T^2 as the test set and L_2 as the initial SOPS. Applying T_3 to L_2 we obtain $L_2^3 = \{< \{2,3\}, \Psi >, < \{5,6\}, \Psi >, < \{1\}, \{5,6\} >, < \{2,3\}, \{7\} >, < \{0\}, \{7\} >, < \{1\}, \{7\} >, < \{5,6\}, \{7\} >, < \{4\}, \{7\} >\}$ and $F_3^2 = 0$ and since there are no more test vectors to be simulated the algorithm terminates. Since T_3 does not drop any fault, it is not included in the set S_3.

Hence , the final I_{DDQ} subset is $S_3 = \{T_1, T_2\}$. Note that for this example we got the same result as in section 6.2 for the COMB_QUIETEST Heuristic. However, as we will see in section 7 this is usually not the case and the second heuristic does considerably better than the first.

7 Experimental Results

We present the results obtained by running the algorithms stated earlier on ISCAS85 and ISCAS89 benchmark circuits. We first discuss the results for combinational circuits.

Fault simulation was performed using the algorithm **Comb_FaultSim** with both stuck-at-test sets (Table 5) and random test sets (Table 6). The stuck-at tests were generated using the stuck-at test generator provided with Octtools[15]. Only one random test set was used for the results presented in Table 6. The number of random tests is about the same as the number of stuck-at tests. Note that the fault coverage is approximately the same even when the number of random tests are much smaller (C5315, C2670, etc.). The running times are also about the same. It therefore seems that a randomly generated I_{DDQ} test set for BFs is definitely of comparable quality, if not better, when compared with the classical stuck-at test set. We will come back to this issue of random versus stuck-at test set.

Diagnosis was performed using both a stuck-at test set and a random test set. For each circuit, we randomly generated 25 faults and the average values are tabulated in Tables

9,10. Note that the average number of residual faults is very small compared to the total number of BFs and the CPU time required is also very small. This shows the efficiency and effectiveness of the algorithm. On comparing the two tables we note that the number of residual faults when stuck-at test sets are used is much smaller than when random test sets are used. Thus, in contrast to simulation, stuck-at tests appear to be better for I_{DDQ} measurement based diagnosis when compared with random testing.

The results of the subset selection algorithms are shown in Tables 11,12,15 and 16. Tables 11,12 are for the COMB_QUIETEST heuristics and Tables 15,16 are for the SETCOVER heuristic. From Tables 11,12,15,16 we note that the COMB_QUIETEST heuristic is very fast when compared with the SETCOVER heuristic. Compare Tables 11,12 and 15,16. We note that whether we start with stuck-at test set or a random test set we get a smaller test set when SETCOVER heuristic instead of COMB_QUIETEST heuristic is used. The difference in the quality of the solution is quite substantial. This is more pronounced in the case of the random test set. We consider the solution size to be more important and therefore propose the use of SETCOVER for the I_{DDQ} subset selection problem for combinational circuits.

Next compare Tables 11,12. We note that the subset selected by COMB_QUIETEST is smaller if we start with a random test set rather than a stuck-at test set of comparable quality. By comparing Tables 15 and 16 we arrive at the same conclusion about the SETCOVER heuristic. These results suggest that a random test set is a better source for I_{DDQ} tests for BFs when compared with stuck-at tests. Its implication is discussed in section 1. However, as cautioned earlier, this is based on a comparison of one stuck-at test set with only one random test set.

In the following discussion we explain the results obtained using the algorithms for sequential circuits. Fault simulation was performed using the algorithm **Seq_FaultSim** with both stuck-at test sets and random test sets. The stuck-at test sets used were those available through the public distribution site of HOPE [16]. This test set was obtained using GENTEST [8] . Again here only one random test set was used for the results presented in Tables 8 and 9. The number of random tests used was about the same as the number of stuck-at tests. The fault coverage obtained using random test set is usually the same as the coverage obtained using stuck-at test set. However, unlike combinational circuits the stuck-at test set does considerably better for some circuits (eg. s382). The running times are also about the same in most cases.

Results of the subset selection algorithm for sequential circuits SEQ_QUIETEST are given in Tables 13 and 14. A comparison of these two tables suggest that the subset obtained using the random test set is about the same as that obtained from stuck-at test set. Note the significance of this conclusion. Even deterministic stuck-at test generation for sequential circuits is considered a time consuming and difficult problem. However, for I_{DDQ} testing we need not use deterministic test generation and random test generation suffices even when considering all BFs in sequential circuits.

8 Conclusion

A paradigm for I_{DDQ} measurement based analysis of all two node BFs was proposed. Algorithms for I_{DDQ} measurement based simulation, diagnosis and I_{DDQ} subset selection for all two node BFs in combinational circuits was derived using only logic simulation and a judicious combination of four simple rules. Experimental results show that for combinational circuits randomly generated I_{DDQ} test sets perform considerably better than stuck-at test sets. Also, the proposed SETCOVER heuristic gave better results than the "QUIETEST-like heuristic".

We also showed that the fault simulation and one of the I_{DDQ} subset selection algorithm for combinational circuits can also be used with some modification for sequential circuits. For sequential circuits, random test sets and stuck-at test sets are comparable.

References

[1] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", special session on ATPG and Fault simulation, *IEEE Int'l. Symposium on Circuits and Systems*, 1985.

[2] 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.

[3] S. Chakravarty, "On Computing Tests for Bridging and Leakage Faults: Complexity Results and Universal Test Sets", *IEEE VLSI DESIGN '92*, pp. 49-54.

[4] S. Chakravarty, P. J. Thadikaran, "Simulation and Generation of Iddq Tests for Bridging Faults in Combinational Circuits",*1993 IEEE VLSI Test Symposium*, pp. 25-32.

[5] S. Chakravarty and M. Liu, "Algorithms for Current Monitor Based Diagnosis of Bridging and Leakage Faults". *29th ACM/IEEE Design Automation Conference*, pp. 353-356, 1992.

[6] S. Chakravarty, MinSheng Liu, "Algorithms for Current Monitor Based Diagnosis of Bridging and Leakage Faults", *Journal of Electronic Testing: Theory and Applications*, Vol. 3, pp. 377 - 385, 1992. Kluwer Academic Publishers, Boston.

[7] S. Chakravarty, S. Suresh, "I_{DDQ} Measurement Based Diagnosis of Bridging Faults in Full Scan Circuits". To be presented at *IEEE VLSI DESIGN '94*.

[8] W. T. Cheng, T. J. Chakraborty, "GENTEST: An automatic Test Generation System for Sequential Circuits", *Computer*, April 1989.

[9] D. S. Johnson, "Approximation algorithms for Combinatorial problems", *J. Comb. and Sys. Sci.,* 1974, pp. 256-278.

[10] W. Debany, "Coverage of Node Shorts Using Internal Access and Equivalence Classes", *VLSI Design*, Vol. 1, No. 1, pp. 71 - 86, 1993.

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

[12] F. J. Ferguson and T. Larabee, "Test Pattern Generation for Realistic Bridging Faults in CMOS ICs", *Proc. 1991 Int'l Test Conf.*, pp. 492 - 499.

[13] 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.

[14] A. Jee and F. J. Ferguson, "Carafe: An Inductive Fault Analysis Tool for CMOS VLSI Circuits", *1993 IEEE VLSI Test Symposium*, pp. 92-98.

[15] T. Larabee, "Test Pattern Generation Using Boolean Satisfiability", *IEEE Trans. on Comput.-Aided Design*, Vol. 11, No. 1, pp. 4-15, January 1992.

[16] H. K. Lee and D. S. Ha, "HOPE: A Fast Parallel Fault Simulator for Sequential Circuits," *Proc. of Design Automation Conference*, June 1992.

[17] K. J. Lee and M. A. Breuer, "Design and Test Rules for CMOS Circuits to Facilitate I_{DDQ} Test of Bridging Faults", *IEEE Trans. on Comput.-Aided Design*, Vol. No. 5, pp. 659-669, May 1992.

[18] W. Maly and P. Nigh, "Built-In Current Testing: Feasibility Study", *1988 Int'l Conf. on Computer-Aided Design*, pp. 340-343.

[19] W. Mao, R. K. Gulati, D. K. Goel, M. D. Ciletti, "QUIETEST: A Quiescent Current Testing Methodology for Detecting Leakage Faults", In *Proc. 1990 IEEE Int'l Conf. on Computer-Aided Design*, pp. 280 - 283.

[20] P. Nigh and W. Maly, "Test Generation for current testing", *IEEE Design and Test of Computers*, Feb. 1990, pp. 26 - 38.

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

[22] W.Maly, J.P.Shen and F.J.Ferguson, "Systematic Characterization of Physical Defects for Fault Analysis of MOS IC cells", *Int'l Test Conference*, Sept. 1984, pp. 390-399.

Table 1		
	Node 0	Node 1
T_1	1	0
T_2	0	1

Table 2	
	Node 0
T_1	1
T_2	0

Table 3		
	Node 0	Node 1
T_1	1	0
T_2	0	1
T_3	1	1

Table 4	
	Node 0
T_1	1
T_2	0
T_3	1

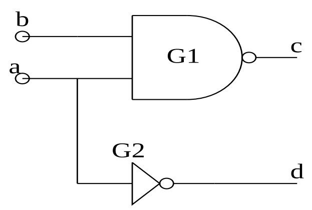

Figure 1:

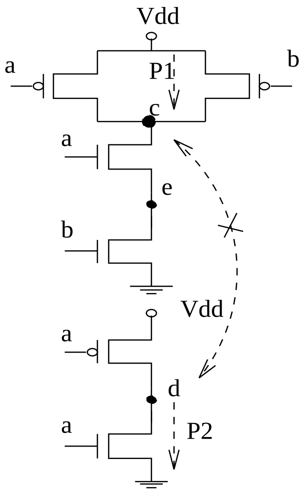

Figure 2:

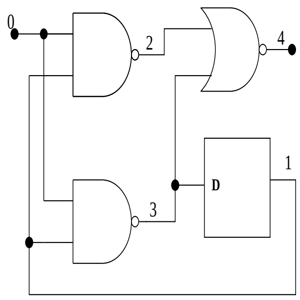

Figure 3:

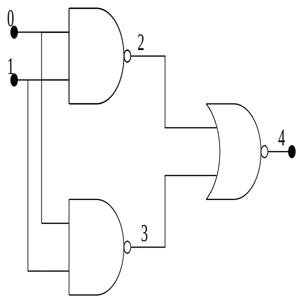

Figure 4:

Table 5: Simulation using Stuck-at Test sets (Combinational)										
	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Faults	108345	526851	436645	675703	1269621	3904615	6402831	15100260	12941328	27162135
Coverage	98.72	99.66	99.85	99.76	99.66	99.78	99.85	99.91	98.64	99.92
Time (CPU Sec)	3.70	13.42	7.95	14.28	34.80	248.02	128.18	456.72	36.12	477.12
# of Test Vectors	74	100	98	99	146	310	227	370	83	278

Table 6: Simulation using random Test sets (Combinational)										
	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Faults	108345	526851	436645	675703	1269621	3904615	6402831	15100260	12941328	27162135
Coverage	98.8	98.76	99.64	98.82	98.54	99.82	99.83	99.95	99.98	99.32
Time (CPU Sec)	4.23	14.85	7.88	16.85	73.77	55.55	83.37	120.18	32.25	848.80
# of Test Vectors	100	100	100	150	150	150	175	200	200	200

Table 7: Simulation using Stuck-at Test sets (Sequential)													
	s1238	s1423	s1488	s1494	s298	s344	s382	s444	s526	s641	s713	s820	s832
# of Faults	869221	1261666	1902225	1929630	47278	59685	66430	81003	152628	241860	285390	419070	435711
Coverage	98.98	89.11	95.61	96.20	97.82	97.74	99.33	99.18	73.72	97.92	93.74	93.69	91.24
Time (CPU Sec)	130.52	20.53	446.52	353.65	5.3	3.2	55.50	65.62	136.92	9.52	10.23	184.87	176.67
# of Test Vectors	349	36	590	469	162	91	2463	1881	754	133	107	411	377

Table 8: Simulation using Random Test sets (Sequential)													
	s1238	s1423	s1488	s1494	s298	s344	s382	s444	s526	s641	s713	s820	s832
# of Faults	869221	1261666	1902225	1929630	47278	59685	66430	81003	152628	241860	285390	419070	435711
Coverage	98.76	85.16	96.31	96.30	94.25	98.77	77.07	75.54	75.36	95.31	94.48	91.04	90.73
Time (CPU Sec)	133.97	19.38	419.18	339.17	7.0	3.03	175.8	154.47	147.82	8.92	8.02	201.37	187.85
# of Test Vectors	350	35	590	470	160	90	2400	1800	750	130	100	400	370

Table 9: Diagnosis using Stuck-at Test sets										
	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Faults	108345	526851	436645	675703	1269621	3904615	6402831	15100260	12941328	27162135
Avg. # of residual faults	4	7	5	5	74	11	28	11	12747	20
Time (CPU Sec)	0.37	0.69	1.52	1.09	2.45	12.47	4.84	35.20	7.81	27.09

Table 10: Diagnosis using Random Test sets										
	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Faults	108345	526851	436645	675703	1269621	3904615	6402831	15100260	12941328	27162135
Avg. # of residual faults	58	46	5	92	826	8	21	11	5	7443
Time (CPU Sec)	0.51	0.54	0.57	1.03	5.61	2.69	2.70	7.69	3.65	43.44

Table 11: Subset Selection using Stuck-at-Test sets - COMB_QUIETEST Heuristic

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Test Vectors	74	100	98	99	146	310	227	370	83	278
Subset Size	68	99	88	99	140	308	218	364	83	269
Coverage	98.72	99.66	99.85	99.76	99.66	99.78	99.85	99.91	98.64	99.92
Time (CPU Sec)	3.88	13.58	7.75	14.52	34.25	247.65	129.37	458.67	36.42	478.43

Table 12: Subset Selection using Random Test sets - COMB_QUIETEST Heuristic

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Test Vectors	100	100	100	150	150	150	175	200	200	200
Subset Size	81	87	80	102	127	120	171	165	65	189
Coverage	98.8	98.76	99.64	98.82	98.54	99.82	99.83	99.95	99.98	99.32
Time (CPU Sec)	4.12	14.72	8.12	17.27	74.53	56.08	83.78	120.47	32.87	855.53

Table 13: Subset Selection using Stuck-at-Test sets - SEQ_QUIETEST Heuristic

	s1238	s1423	s1488	s1494	s27	s298	s344	s382	s444	s526	s641	s713	s820	s832
# of Test Vectors	349	36	590	469	15	162	91	2463	1881	754	133	107	411	377
Subset Size	272	35	93	82	9	52	40	130	176	47	81	77	141	130
Coverage	98.98	89.11	95.61	96.20	91.80	97.82	97.74	99.33	99.18	73.72	97.92	93.74	93.69	91.24
Time (CPU Sec)	130.72	21.05	449.45	356.58	0.02	5.28	3.27	56.78	65.72	136.93	9.80	10.78	183.52	176.45

Table 14: Sequential Subset Selection using Random Test sets - SEQ_QUIETEST Heuristic

	s1238	s1423	s1488	s1494	s27	s298	s344	s382	s444	s526	s641	s713	s820	s832
# of Test Vectors	350	35	590	470	15	160	90	2400	1800	750	130	100	400	370
Subset Size	256	34	91	87	7	46	47	52	43	51	73	69	150	146
Coverage	98.76	85.16	96.31	96.30	92.86	94.25	98.77	77.07	75.54	75.36	95.31	94.48	91.04	90.73
Time (CPU Sec)	134.63	19.27	421.00	339.22	0.02	7.23	3.08	176.62	157.33	148.97	9.02	8.02	203.30	188.33

Table 15: Subset Selection using Stuck-at-Test sets - SETCOVER Heuristic

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Test Vectors	74	100	98	99	146	310	227	370	83	278
Subset Size	52	93	61	96	126	246	180	319	70	235
Coverage	98.72	99.66	99.85	99.76	99.66	99.78	99.85	99.91	98.64	99.92
Time (CPU Sec)	26.88	160.53	72.88	190.55	671.85	4181.08	2850.45	13269.90	592.50	11822.63

Table 16: Subset Selection using Random Test sets - SETCOVER Heuristic

	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
# of Test Vectors	100	100	100	150	150	150	175	200	200	200
Subset Size	52	62	52	70	96	80	145	109	38	NA
Coverage	98.8	98.76	99.64	98.82	98.54	99.82	99.83	99.95	99.98	NA
Time (CPU Sec)	40.07	180.13	84.45	264.05	937.40	854.52	1864.20	3167.98	602.78	NA

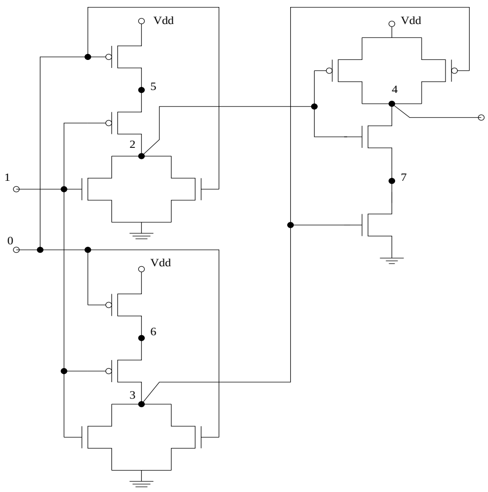

Figure 5: