A Sampling Technique for Diagnostic Fault Simulation [1]

Sreejit Chakravarty
226 Bell Hall
Dept. of Computer Science
State University of New York
Buffalo, NY 14260

Technical Report Number 95-48, Department of Computer Science
State University of New York, Buffalo, NY 14260.

ABSTRACT

The quality of diagnostic test sets (DTS) are determined using **diagnostic fault simulation**(DFS). Approximation algorithms, based on "sampling", for this computationally difficult problem are explored. "Fault sampling", used very effectively for fault simulation, cannot be used for DFS. As an alternative, we propose using "EC/IC Sampling" for DFS. It samples the set of equivalence classes (EC)/ indistinguishible classes (IC). An approach to sample ECs/ICs implicitly, without explicity enumerating the set of ECs/ICs, is presented. Experimental evaluation of the proposed technique show it to be very effective.

1 Introduction

Diagnosis is the process of determining the cause(s) of circuit failures once they have been identified to be faulty by a detection test set. This information is used to improve the fabrication process.

Analogous to detection test sets, diagnosis requires **diagnostic test sets** (DTS). **Diagnostic fault simulation** is the process of evaluating the capability of a DTS to diagnose a circuit. Like fault coverage and fault efficiency computed by fault simulation algorithms diagnostic fault simulation algorithms compute quantitative measures[6, 7, 8, 13, 17, 19, 12], called **diagnostic measures**, defined below.

The diagnostic measures assume a set F of faults like stuck-at, etc. First consider only combinational circuits. Two faults $f_1, f_2 \in F$ are **equivalent**, with respect to a set of input vectors T, iff for each input vector T_i in T the reponses of the two faulty circuits respectively with f_1, f_2 in them are the same. For combinational circuits this binary relation on F is an equivalence relation and partitions it into **equivalence classes**(EC). Let $E_1, \ldots, E_m$ be the equivalence classes of F, with respect to the test set T; R be the number of faults in F; and P be the number of equivalence classes of size 1. After each diagnosis using DTS T we are left with a set of possible faults from F that are equivalent with respect to T. This set is the **residual set**.

Diagnostic Resolution(DR) of DTS T is the percentage of faults in F that can be completely distinguished from all other faults using T. More formally, $DR = \frac{P}{R} \times 100$.

Diagnostic Power(DP) of DTS T is the percentage of pairs of faults that are not equivalent with respect to T.

Expected Residual Size(ERS) of DTS T, is the expected value of the size of the residual set when the circuit is diagnosed using T. More formally, $ERS = \sum_{i=1}^{m} \frac{\|E_i\| * \|E_i\|}{R}$. The term **MAX ERS** is self explanatory.

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

We consider DR, ERS and MAX ECS in our study. For sequential circuits, when we assume an unknown initial state, the logic level of some of the primary outputs may not be defined. For this reason the binary relation defined above on S need not be an equivalence relation [17]. So, $E_1, \ldots, E_m$, may not be mutually disjoint and is more appropriately called **Indistinguishible Classes**(IC)[17]. Besides this difference, the definition of the above diagnostic measures remain unchanged.

Diagnostic simulation algorithms for stuck-at faults has been discussed in [7, 8, 13, 17, 18]. Not withstanding recent advances to speed up such algorithms[12, 19] diagnostic fault simulation remains a computationally difficult problem. The primary reason is the need to perform fault simulation without fault dropping. The trend to diagnose larger circuits and consider fault models like bridging faults, leakage faults etc. tend to increase the number of faults to be analysed. This adds to the difficulty of diagnostic fault simulation very significantly.

To cope with this increase in complexity we explore the possibility of computing approximate diagnostic measures. Our approximate algorithm uses "sampling". A similar approach, called "fault sampling"[4, 2, 20], has been used very successfully for fault simulation. We briefly review "fault sampling" in section 2.

"Fault Sampling" cannot be used for diagnostic fault simulation. We propose using "EC/IC Sampling" that samples the set of ECs/ICs for diagnostic fault simulation. The difficulty of sampling ECs/ICs is that no explicit enumeration of the set of ECs/ICs are available as is the set of faults for "fault sampling". Infact, if an explicit enumeration of the EC/ICs exists the diagnostic measures can be computed very easily.

In this paper we propose a novel method for sampling the set of ECs/ICs without explicitly computing the entire set of ECs/ICs. This technique is discussed in Section 2. A case study to evaluate the proposed sampling technique was performed. The exact diagnostic simulation algorithm is discussed in Section 3.2 and the approximate diagnostic simulation algorithm in Section 3.3. Experimental results are presented in Section 4. The results are very encouraging. The approximation algorithm is considerably faster and the error is small.

2 Sampling

Fault Sampling is a sampling technique that has been used to compute an approximate value of the fault coverage of detection test sets. Let T be the detection test set, F be the set of faults and D the subset of F detected by T. Then, **fault coverage** $C = \frac{\|D\|}{\|F\|}$.

In *fault sampling*, to contain the complexity of fault simulation, a small subset S of F is selected at random without replacement. Next, the subset Q of S detected by T is computed. Then, approximate fault coverage is: $C_{app} = \frac{\|Q\|}{\|S\|}$. It has been shown that $\|C_{app} - C\|$ is small, with a very high probability. This approach has been used very effectively.

The question we ask is: how can "sampling" be used for diagnostic simulation ? This question is important because diagnostic simulation is computationally more difficult than fault simulation. The primary reason being that, except in some special cases[19], no fault dropping is possible. Therefore, there is an even greater urgency to develop approximate diagnostic fault simulation algorithms.

Note that for diagnostic simulation the sample space to be used is the set of EC/ICs of the set of faults F with respect to the test set T. So, the sampling technique used for fault simulation cannot be used for this purpose. A difficulty in sampling the sample space of equivalence classes is that no explicit enumeration of this set exists. In addition, this set is as difficult to compute as diagnostic simulation itself. This leaves us with no choice but to **sample the set of EC/ICs implicitly**. This immediately raises the question: how can the set of EC/ICs be sampled implicitly ? We suggest the following technique.

EC/IC Sampling

Notations:
F: the set of modelled faults;
T: the diagnostic test set being evaluated;
E: $\{E_1, E_2 \ldots, E_m\}$, the set of EC/IC of F with respect to the test set T.
$P(f_i)$: The equivalence class, with respect to T, that f_i belongs to.
S: The set of sampled faults, $\{f_1, f_2, \ldots, f_n\}$
M: $\|\{P(f_i) : \|P(f_i)\| = 1\}\|$.

Step 1. Randomly select, without replacement, a set of faults S.

Step 2. For each fault $f_i \in S$, compute $P(f_i)$. Let $P(f_1), P(f_2), \ldots P(f_n)$ be the list of equivalence classes so computed. Note that if two faults f_i, f_j are equivalent (or indistinguishible) with respect to T and both $f_i, f_j \in S$ then $P(f_i), P(f_j)$ are identical. However, we retain both of them in the list. This is equivalent to sampling with replacement.

Step 3. Compute the approximate diagnostic measures as follows.

$$DR_{app} = \frac{M}{n} \quad ; \quad ERS_{app} = \frac{\sum_{i=1}^{n} \|P(f_i)\|}{n}$$

There are two issues that need to be addressed. (i) In Step 2 why do we perform sampling with replacement ? (ii) How do we compute $P(f_i)$?

The reason for (i) is that if we carefully examine the definition of ERS we note that it is equivalent to the expected value of the size of an equivalence class when one is selected at random. In this case the larger equivalence classes have a larger weightage. We use sampling with replacement in our approximation to simulate this effect.

The second issue is considerably more difficult to answer in general. Methods to compute $P(f_i)$ will depend on the fault model and the type of circuit - sequential or combinational - under investigation. In this paper we present one method to compute $P(f_i)$ selectively.

3 A Case Study

To evaluate the proposed strategy we performed a case study. The case study consist of computing the diagnostic measures for the stuck-at test sets computed using [1]. These test sets were evaluated as to their effectiveness in diagnosing BFs in combinational circuits assuming I_{DDQ} measurement. I_{DDQ} measurement based detection of BFs and classes of BFs considered in our experiments are discussed below.

A method to compute the exact value of the diagnostic measures is discussed in section 3.2 and an implementation of the proposed sampling technique is discussed in section 3.3. Experimental results are presented in Section 4. We start with some background in section 3.1.

3.1 Background

Consider the circuit shown in Figure 1(a). Its switch level diagram is shown in Figure 1(b). In the fault free case, and on application of input vector $T1 =< a = 0, b = 0, c = 1, d = 1 >$ the steady state supply current (I_{DDQ}) is negligible. However, if the BF $< e, f >$ is present and the input vector $T1$ is applied then there will be a conducting path along the pullup network ($P1$), the BF $< e, f >$, and the pulldown network ($P2$). This results in a large I_{DDQ} which signals the presence of a fault.

In Figure 1 nodes h, j, k are not outputs of any of the gates nor are they primary inputs. Such nodes are known as **internal nodes**. Unlike the rest of the nodes, internal nodes can take on the logic values 0, 1, f (for floating).Consider the input vector $T2 =< a = 1, b = 1, c = 0, d = 0 >$. There is no path from node h to either V_{DD} or GND and h is said to be floating. As a result, on application of $T2$ and even in the presence of any of the BFs involving node h and any other node I_{DDQ} will be negligible. Hence $T2$ will not detect any of these BFs. Thus, we declare an input vector T to be an I_{DDQ} test for a BF $< X, Y >$ if and only if $T(X) \neq T(Y), \& T(X), T(Y) \in \{0, 1\}$[14, 15, 16].

Several classes of BF can be defined. We use two of them in our experiment. Metal nodes are nodes that are either primary inputs or gate outputs. They are usually implemented in the metal layer. **MetalBFs** is the set of all BFs between metal nodes. **AllBFs** is the set of all BFs between any two nodes. Note that we are not suggesting the use of these fault models for diagnosis. Our intent is to strech the proposed paradigm and have selected these fault models because of the large number of faults they contain.

3.2 Exact Diagnostic Simulation

The exact diagnostic simulation algorithm consists of two basic steps, **UpdatePartitions**, and **UpdateEquivClasses**. The algorithm iterates through these two basic steps,described below, for each input vector.

3.2.1 Partitions

Let T be the test set and $T_i(x)$ the fault free logic value at x on application of $T_i \in T$. Two nodes x, y are equivalent with respect to T if and only if for each input vector $T_i \in T$ $T_i(x)$ equals $T_i(y)$. This equivalence relation on the set of nodes partitions it into equivalence classes. However, to avoid confusion with equivalence classes of BFs we refer to the equivalence classes of nodes as **partitions** and the set of equivalence classes as **node partition**.

Table 1 defines a test set and the fault free logic value for each node in the circuit of Figure 1. Nodes c, h are equivalent w.r.t the test set. The *node partition* are: $\{a\}, \{b\}, \{c, h\}, \{d, g\}, \{e, j\}, \{f\}, \{k\}$.

The *node partition* of circuit C with respect to a test set T can be computed by first computing a table similar to Table 1. Followed by pairwise comparison of the values assigned to each node. This is an expensive algorithm.

A faster algorithm is discussed in [11] and explained with the example of Table 1 and Figure 1. It is illustrated using Figure 2. Let $Part(INIT)$ be the initial set of partitions and $Part(T_i)$ be the *node partition* w.r.t $\{T_1, \ldots, T_i\}$. Assume **AllBFs** to be the fault model.

Initially, there is only one partition consisting of all nodes in the circuit. Thus, $Part(INIT) = \{P0\} =$

$\{\{a, b, c, d, e, f, g, h, j, k\}\}$. Next $T1$ is processed. Let $P1(P2)$ be the subset of $P0$ set to $0(1)$ by $T1$. In general there will be three such subsets corresponding to the values $0, 1, f$. Accordingly, $P0$ is split into two parts $P1, P2$. Thus, $Part(T_1) = \{P1, P2\} = \{\{a, b, d, g, k\}, \{c, e, f, h, j\}\}$. Next, $T2$ splits $P1(P2)$ into $P3, P4, P5(P6, P7)$ giving us $Part(T_2) = \{P3, P4, P5, P6, P7\} = \{\{b\}, \{a, d, g\}, \{k\}, \{e, f, j\}, \{c, h\}\}$.

The process uses $O(N)$ space and computing $Part(T_{i-1})$ from $Part(T_i)$ takes time $O(N)$ where N is the number of nodes[11]. Henceforth we refer to the process of computing $Part(T_i)$ from $Part(T_{i-1})$ as **UpdatePartitions**.

3.2.2 Equivalence Classes

Two BFs $< x, y >, < u, v >$ are I_{DDQ} **equivalent** with respect to a test set if and only if for each $T_i \in T$ one of (i), (ii) or (iii) is satisfied. If either (i) or (ii) is satisfied then T_i does not detect either of the two faults. If condition (iii) is satisfied then T_i detects both the faults.

(i) $T_i(x) = T_i(y), T_i(u) = T_i(v)$.

(ii) At least one of $T_i(x), T_i(y)$ equals f and at least one of $T_i(u), T_i(v)$ equals f.

(iii) $T_i(x) \neq T_i(y), T_i(u) \neq T_i(v)$ and none of $T_i(x), T_i(y), T_i(u), T_i(v)$ equals f.

Explicitly enumerating BFs in an equivalence class leads to inefficiencies. To avoid that we use Sets of Ordered Pairs of Sets (**SOPS**)[9, 10], defined below, to implicitly enumerate BFs. Let A, B be two sets of nodes. Then the ordered pair of sets $< A, B >$ denote the set of BFs $< x, y >$ such that $x \in A, y \in B$. A set of SOPS represents an equivalence class. For example the SOPS $\{< \{a, b, c\}, \{e, d\} >, < \{a, b, c\}, \{a, b, c\} >\}$ denotes the set of BFs $\{< a, e >, < a, d >, < b, e >, < b, d >, < c, e >, < c, d >, < a, b >, < a, c >, < b, c >\}$.

In the above example lines a, b, c are repeated three times. If a large number of nodes are repeated several times the space used by the representation increases. To contain that we use pointers to avoid duplication. Assume that: $P0 = \{a, b, c\}; P1 = \{e, d\}; P2 = \{f\}$. Then, the equivalence class $\{< \{a, b, c\}, \{e, d\} >, < \{a, b, c\}, \{a, b, c\} >\}$ is represented as $\{< P0, P1 >, < P0, P0 >\}$.

3.2.3 Computing Equivalence Classes

Let $EC(INIT)$ be the initial set of ECs and $EC(T_i)$ be the set of ECs with respect to $\{T_1, \ldots, T_i\}$. We use the same example and refer to Figure 2. To start with, we have $Part(INIT) = \{P0\}$ and $EC(INIT) = \{< P0, P0 >\} = \{E0\}$.

Next, we process T_1. *Node partition* $Part(T_1) = \{P1, P2\} = \{\{a, b, d, g, k\}, \{c, e, f, h, j\}\}$ is computed. $P1(P2)$ is the subset of $P0$ set to $0(1)$ by $T1$. Thus, $\{< P1, P2 >\}$ is the subset of $E0$ detected by T_1. Similarly, $\{< P1, P1 >, < P2, P2 >\}$ is the subset of $E0$ not detected by T_1. Hence, $E0$ is split into two ECs: (i) the subset of $E0$ detected by T_1; and (ii) the subset of $E0$ not detected by T_1. The new set of equivalence classes with respect to $\{T_1\}$ is $EC(T_1) = \{\{< P1, P2 >\}, \{< P1, P1 >, < P2, P2 >\}\} = \{E1, E2\}$. After computing $EC(T_1)$ the space used by $EC(INIT), Part(INIT)$ is released.

Next T_2 is processed as shown in Figure 2(c). We first compute $Part(T_2) = \{\{b\}, \{a, d, g\}, \{k\}, \{e, f, j\}, \{c, h\}\}$ $= \{P3, P4, P5, P6, P7\}$. Next each of $E1, E2$ in $EC(T_1)$ is split into two Equivalence Classes as was done for $E0$. Note that $< P3, P3 >$ and $< P5, P5 >$ do not represent any BFs and are deleted. At the end of this we have:

$$
\begin{aligned}
EC(T_2) &= \{\ \{< P4, P6 >, < P3, P7 >\},\ \ \{< P3, P6 >, < P4, P7 >, < P5, P6 >, \\
&\quad < P5, P7 >\}, \{< P3, P4 >, < P6, P7 >\},\ \ \{< P3, P5 >, < P4, P4 >, \\
&\quad < P4, P5 >, < P6, P6 >, < P7, P7 >\}\ \ \} \\
&= \{E3, E4, E5, E6\}
\end{aligned} \tag{1}
$$

More formally, the rules for computing $EC(T_i)$ from $EC(T_{i-1})$ follows. Let $EC(T_{i-1})$ be $\{E_0, \ldots, E_M\}$. First, using UpdatePartition, $Part(T_i)$ is computed from $Part(T_{i-1})$. Next, each $E_j \in EC(T_{i-1})$ is split into two set of BFs $E_j(Det), E_j(UnDet)$ where $E_j(Det)(E_j(UnDet))$ is the subset of E_j detected (not detected) by T_i.

Let $E_j \in EC(T_{i-1})$ be represented by the SOPS $\{< A_1, B_1 >, \ldots, < A_Q, B_Q >\}$ where the $A_k's, B_k's$ are pointers to partitions in the collection of partitions $Part(T_{i-1})$. $E_j(Det)$ is computed as follows. For each $< A_k, B_k > \in E_j$ we compute $Det(k)$, the subset of BFs represented by this pair of sets that are detected by T_i, as follows. Let $A_k(0), A_k(1), A_k(f)$ be pointers to partitions in $Part(T_i)$ such that for all $t \in \{0, 1, f\}$ $A_k(t)$ points to the subset of nodes in A_k set to t by T_i. Notations $B_k(0), B_k(1), B_k(f)$ are similarly defined. Then,

$$
Det(k) = \{< A_k(0), B_k(1) >, < A_k(1), B_k(0) >\} \tag{2}
$$

$$
E_j(Det) = \bigcup_{k=1}^{Q} Det(k) \tag{3}
$$

Let $UnDet(k)$ be the subset of BFs represented by $< A_k, B_k >$ that are not detected by T_i. Then,

$$
\begin{aligned}
UnDet(k) &= \{< A_k(0), B_k(0) >, < A_k(0), B_k(f) >, < A_k(1), B_k(1) >, \\
&\quad < A_k(1), B_k(f) >, < A_k(f), B_k(0) >, < A_k(f), B_k(1) >, < A_k(f) >, B_k(f) >\}
\end{aligned} \tag{4}
$$

$$
E_j(UnDet) = \bigcup_{k=1}^{Q} UnDet(k) \tag{5}
$$

In both Eqn(2), Eqn(5) if any of the pairs of pointers denote an empty set of BFs it is deleted from the list. Note that, in Eqn(2), if A_k is the same as B_k then the two pairs denote the same set of BFs. Only one of them is retained. Similarly, in Eqn(5), if A_k is the same as B_k then $< A_k(1), B_k(f) >$ is the same as $< A_k(f), B_k(1) >$ and only one is retained. We achieve this by adding two other rules for pairs of the form $< A_k, A_k >$ which are simplication of Eqn(2), Eqn(5). Using these additional rules we guarantee that every pair of partition pointers represent disjoint sets of BFs and, by definition, the ECs represent disjoint set of BFs. The use of Eqn(2) to Eqn(5) is referred to as **UpdateEquivClasses** and uses "list splitting", a paradigm used earlier for diagnostic simulation of stuck-at faults[7, 8, 18, 19].

3.2.4 Outline of the Algorithm

Note that as the number of vectors simulated increases the number of pairs of partition pointers in each SOPS representing an EC increases. However, it will never be worse than explicitly enumerating all the faults in the EC. In the initial stages of the simulation the use of SOPS results in a considerable speedup. To improve the speed of simulation during the later stages we use the following strategy. Note that once an EC has reduced to size 1 we delete it and keep track of the number of ECs of size 1. The better the diagnostic test set the more effective this strategy will be. The diagnostic simulation algorithm is outlined below.

ExactIddqDiagSim

/* Let $\{T_1, \ldots, T_N\}$ be the given diagnostic test set. */
BEGIN
  Compute $Part(INIT) = \{P0\}$. /* It will contain only one partition pointed to by $P0$ */
  Set $EC(INIT)$ to $\{< P0, P0 >\}$.
  SINGLETONS = 0;
  For each test T_i
  BEGIN
    $Part(T_i)$ = UpdatePartition($Part(T_{i-1})$);
    $EC(T_i)$ = UpdateEquivClass($EC(T_{i-1})$);
    Traverse $EC(T_i)$ and delete ECs of size 1. When such an EC is deleted increment SINGLETONS.
    Release Space used by $Part(T_{i-1}), EC(T_{i-1})$.
  END
  Traverse $EC(T_N)$ and compute DR, ERS, MAXECS defined in section 1. The value of SINGLETONS have to be taken into account in this calculation.
END(* **IddqDiagSim** *)

To complete the example we started in the previous subsection refer to Figure 3. For now, ignore the numbers within [], {}. Figure 3(a) shows how each partition is split into its subpartitions using UpdatePartitions. In Figure 3(b) we show how each E_j in $EC(T_2)$, from Figure 2(c), is split into its two subsets - $E_j(Det), E_j(UnDet)$.

To compute the diagnostic measures first compute the size of each EC. This is done by computing the number of BFs represented by each pair of partition pointers. For example, since both $P10, P12$ point to partitions of size 2 $< P10, P12 >$ representes 4 BFs. In Figure 3(b) the number of BFs represented by each pair of partition pointers is shown next to it, within []. Note that $< P12, P12 >$ represents 1 not 4 BFs. $< P9, P9 >$, $< P12, P12 >$ represent empty sets and are deleted.

The set of BFs represented by distinct pairs of partition pointers are disjoint. Each BF occurs in an EC at most once. Therefore, size of an EC is the sum of the size of the set of BFs represented by the pairs of partition pointers constituting an EC. For each EC of Figure 3(b) its size is shown within {}. Thus: SINGLETONS = 0; the total number of faults is 45; $DR = \frac{0}{45} \times 100 = 0\%$; MaxECS is 7; and

$$ERS \quad = \quad \frac{(49 + 16 + 49 + +49 + 9 + 36 + 49 + 16)}{45} = 6.07.$$

3.3 Approximate Diagnostic Simulation

We use the EC/IC Sampling technique outlined in Section 2 for approximate diagnostic simulation. We first illustrate the process using the example of Figure 1 and Table 1. For this example, we assume the AllBF fault model.

Let the sample of faults be $S = \{f_1 =< e, h >, f_2 =< b, g >,\ f_3 =< a, b >, f_4 =< d, k >\}$. From Figure 3 we note that $P(f_1) = E12, P(f_2) = E12,\ P(f_3) = E11, P(f_4) = E13$. Therefore, $\|P(f_1)\| = 6, \|P(f_2)\| = 6, \|P(f_3)\| = 3, \|P(f_4)\| = 7$. Hence, $DR_{app} = 0; ERS_{app} = \frac{1}{4} \times (6 + 6 + 3 + 7) = 5.5$. Comparing with the exact values we note that the approximate value of DR is the same as the exact value and the error for ERS is about 10%.

An important point is that our approximation is based on the list of ECs $E11, E11, E12, E13$ instead of the entire set of ECs $\{E7, \ldots, E14\}$. This complete list of ECs is never computed. We next illustrate how this is done.

Let the sample of faults S be as defined above. The idea is to compute the equivalence classes of S with respect to the test set of Table 1. For each equivalence class A of S we also compute the equivalence class B of F such that $A \subset B$. Unnecessary equivalence class of F, as soon as they are recognized to be so, are deleted.

Let $EC(S, INIT)$ be the initial sample of equivalence classes of F. It consists of an ordered pair $< A, B >$ where: A is the set of sampled faults S; and B is $E0$ the set of all faults in the fault model being used. $E0$ is defined in Figure 2 and $EC(S, INIT)$ is shown in Figure 4(a).

Let $EC(S, T_i)$ be the sample of the equivalence classes of F i.e. a sample of $EC(T_i)$ (See section 3.2.3) with respect to the fault sample S. Each member of $EC(S, T_i)$ consists of an ordered pair $< A, B >$. A is an equivalence class of S with respect to $\{T_1, \ldots, T_i\}$. B points to the member of $EC(T_i)$ that A is a subset of. All members of $EC(T_i)$ s.t. it is not a subset of any equivalence class of S are not included in the sample.

First, $T1$ is processed and $EC(S, T1)$ is computed from $EC(S, INIT)$. None of f_1, f_2, f_3, f_4 is detected by T_1. So, $\{f_1, f_2, f_3, f_4\}$ is split as shown in Figure 4(b). From Figure 2 we see that $E0$ is split into $E1, E2$ where $E1(E2)$ is the subset of $E0$ detected(not detected) by $T1$. We therefore delete $E1$ and $EC(S, T1)$ is as shown in Figure 4(b).

Next $T2$ is processed and $EC(S, T2)$ is computed from $EC(S, T1)$. For each ordered pair $< A, B >$ of $EC(S, T1)$ split A into the subset $A_d(A_u)$ of faults in A detected (not detected) by $T1$. Similarly, split B into the subset $B_d(B_u)$ of faults in B detected (not detected) by $T1$. Replace $< A, B >$ by $< A_u, B_u >, < A_d, B_d >$. If $A_u(A_d)$ is empty then $< A_u, B_u >$ $(< A_d, B_d >)$ is deleted. Continuing with our example, we have only one ordered pair $< A, B >$. From Figure 4(c), $A_u = \{f_4\}, A_d = \{f_1, f_2, f_3\}$, $B_d = E5, B_u = E6$. $E5, E6$ are defined in Figure 3(b). Hence, $EC(S, T2)$ is as shown in Figure 4(c). The processing of $T3$ is shown in Figure 4(d). This completes the example. The list of sampled ECs is read off from $EC(S, T3)$.

Note that the approximation algorithm is a modification of *UpdateEquivClasses*. The equivalence classes of the set of sampled faults S is computed. This is done by splitting each equivalence class A of S into two subsets: the subset A_u (A_d) of A not detected (detected) by the current vector T_i being processed (**SplitFaultClass**). Let B be the set of equivalence classes of the modelled faults such that $A \subset B$. Then, B is split into its detected (B_d) and undetected (B_u) subsets (**SplitEquivClass**). If A_d (A_u) is empty then B_d (B_u) is eliminated and not processed any further. This elimination of the equivalence classes of the modelled faults, as soon as they are detected to be irrelevant, is what contributes to the speed up of the process. It also contributes to the inaccuracies in the values computed. A high level description of the above process follows. A similar method can be used for other fault models.

ApproxIddqDiagSim

```
/* Let {T_1, ..., T_N} be the given diagnostic test set. */
BEGIN
  Let S = {f_1, ..., f_n} be the set of sampled faults.
  Compute Part(INIT) = {P0}. /* It will contain only one partition pointed to by P0 */
  Set E0 to < P0, P0 > and EC(S, INIT) to {< S, E0 >}.
  For each test T_i
```

BEGIN
$Part(T_i) = \text{UpdatePartition}(Part(T_{i-1}))$;
/* Compute $EC(S,T_i)$ from $EC(S,T_{i-1})$. $EC(S,T_{i-1}) = \{< A_1, B_1 >, \ldots, < A_P, B_P >\}$. */
For each $< A, B > \in EC(S, T_{i-1})$
BEGIN
SplitFaultClass(A) to get A_u, A_d.
SplitEquivClass(B) to get B_u, B_d.
If A_u (A_d) is empty then delete B_u (B_d).
Replace $< A, B >$ by non-empty pairs from $\{< A_u, B_u >, < A_d, B_d >\}$. END
END
Traverse $EC(S, T_N)$ and compute DR_{app}, ERS_{app}, $MAXECS_{app}$ defined in section 2.
END(* **ApproxIddqDiagSim** *)

4 Experimental results

Table 2 lists ISCAS85[3] and ISCAS89[5] circuits used in our experiment. Only combinational versions of ISCAS89 circuits were used. Vectors is the size of the stuck-at detection test set from [1] used in our experiment. Sample is the number of fault sample used. Gates(Nodes) is the number of gates (nodes) in the circuit. The number of faults in the sample equals the number of nodes in the circuit. METAL (ALL) is the number of faults in MetalBFs (AllBFs). No fault collapsing was used. Note that the sample of faults is very small compared to the total number of faults.

We first compare the values of DR, ERS computed by ExactIddqDiagSim and ApproxIddqDiagSim. Table 3 is for MetalBFs. The columns labels are self-explanatory. The ERROR columns were computed as follows: $ERROR : \frac{\|EXACT - APPROX\|}{EXACT} \times 100$. For some circuits, like 9234 onwards, exact diagnostic simulation had to be aborted for lack of resources. The average error for DR is 3.5% and the standard deviation is low. For ERS the average error is 7.2% and the standard deviation is 4.91. Similar results are tabulated in Table 4 for AllBFs. The average error for DR is about 5.33% with a standard deviation of 5.19. The approximate values of ERS is not as good as it was for MetalBFs.

Table 5 tabulates the CPU time for MetalBFs (under columns labelled METAL) and AllBFs (under columns labelled ALL). EXACT and APPROX are the CPU seconds for exact and approximate diagnostic simulation on a SUN 4/30. RATIO ($= \frac{EXACT}{APPROX}$) is the speedup factor. Note that for both fault models the speedup is between 14-15, on an average. The approximation algorithm is also less memory intensive making it possible to run it for some of the larger circuits like S13207.1, S15850. The speedup is better for the larger circuits, for which data is available, like S5378, C7552.

References

[1] ATALANTA. Sofware available from Prof. D. S. Ha, Bradley Department of Electrical Engineering, VPISU, Blacksberg, Va.

[2] V. D. Agrawal, "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, Vol. 5, No. 3, pp. 189 - 202, Fall 1981.

[3] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Int'l Symp. Circuits and Systems*, pp. 1929 - 34, May 1989.

[4] T. T. Butler, T. G. Hallin, J. J. Kulzer, K. W. Johnson, "LAMP: Application to Switching System Development," *Bell Systems Technical Journal*, Vol. 53, pp. 1535 - 1555, Oct. 1974.

[5] F. Brglez and H. Fujiwara, "A Neural 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.

[6] P. Camurati, D. Medine, P. Prinetto, M. S. Reorda, "A Diagnostic Test Pattern Generation Algorithm," *IEEE Int'l Test Conference*, pp. 52 - 58, 1990.

[7] S. Chakravarty, Y. Gong, "Diagnostic Simulation of Stuck-at Faults in Combinational Circuits," *IEEE VLSI Test Symposium*, pp. 128 - 133, 1994.

[8] S. Chakravarty, Y. Gong, S. Venkataraman, "Diagnostic Simulation of Stuck-at Faults in Combinational Circuits," Submitted to *Journal of Electronic Testing: Theory and Applications.*

[9] S. Chakravarty and M. Liu, "Algorithms for Current Monitor Based Diagnosis of Bridging Faults in Combinational Circuits," *Journal of Electronic Testing: Theory and Applications*, Vol. 3, pp. 377 - 385, 1992.

[10] S. Chakravarty and S. Suresh, "I_{DDQ} Measurement Based Diagnosis of Bridging Faults in Full Scan Circuits," *IEEE Int'l Conference on VLSI Design*, pp. 179 - 182, 1994.

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

[12] J. M. Jou and S. C. Chen, "A Fast and Memory Efficient Diagnostic Fault Simulation Algorithm for Sequential Circuits," *1994 IEEE/ACM Int'l Conf. on Comput.-Aided Design*, pp. 723-726.

[13] K. Kubiak, S. Parkes, W. K. Fuchs and R. Saleh, "Exact Evaluation of Diagnostic Test Resolution," *IEEE/ACM Design Automation Conference*, pp. 347 - 352, June 1992.

[14] K. J. Lee, M. Breuer, "Design and Test Rules to Facilitate I_{DDQ} Test of Bridging Faults," *IEEE Trans. on Comput.-Aided Design*, Vol. 11, No. 5, pp. 659 - 669, May 1992.

[15] W. Maly, "Realistic Fault Modeling for VLSI Testing," *IEEE/ACM Design Automation Conference*, pp. 173 - 180, 1987.

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

[17] E. Rudnick, W. K. Fuchs, J. H. Patel, "Diagnostic Fault Simulation of Sequential Circuits," *IEEE Int'l Test Conference*, pp. 178 - 186, 1992.

[18] G. Ryan, W. K. Fuchs and I. Pomerantz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," *IEEE/ACM Int'l Conf. on Comput.-Aided Design*, pp. 508 - 511, 1993.

[19] S. Venkataraman, et. al., "Rapid Diagnostic Fault Simulation of stuck-at faults in Sequential Circuits Using Compact Lists," *IEEE/ACM 32nd Design Automation Conference*, pp. 133-138, 1995.

[20] R. L. Wadsack, "Design Verification of WE32100 CPUs", *IEEE Design and Test of Computers*, Vol. 1, No. 3, pp. 66 - 75, Aug 1984.

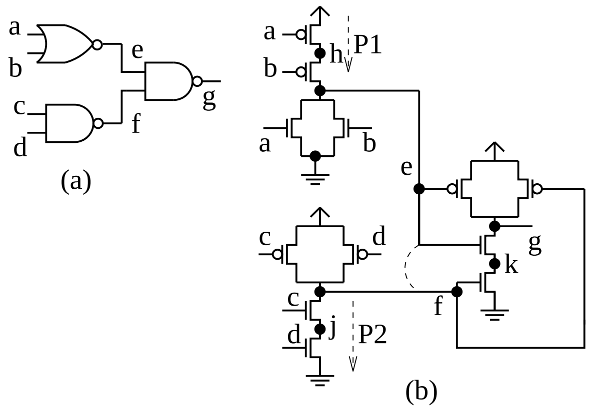

Figure 1: Example of I_{DDQ} Tests for BFs.

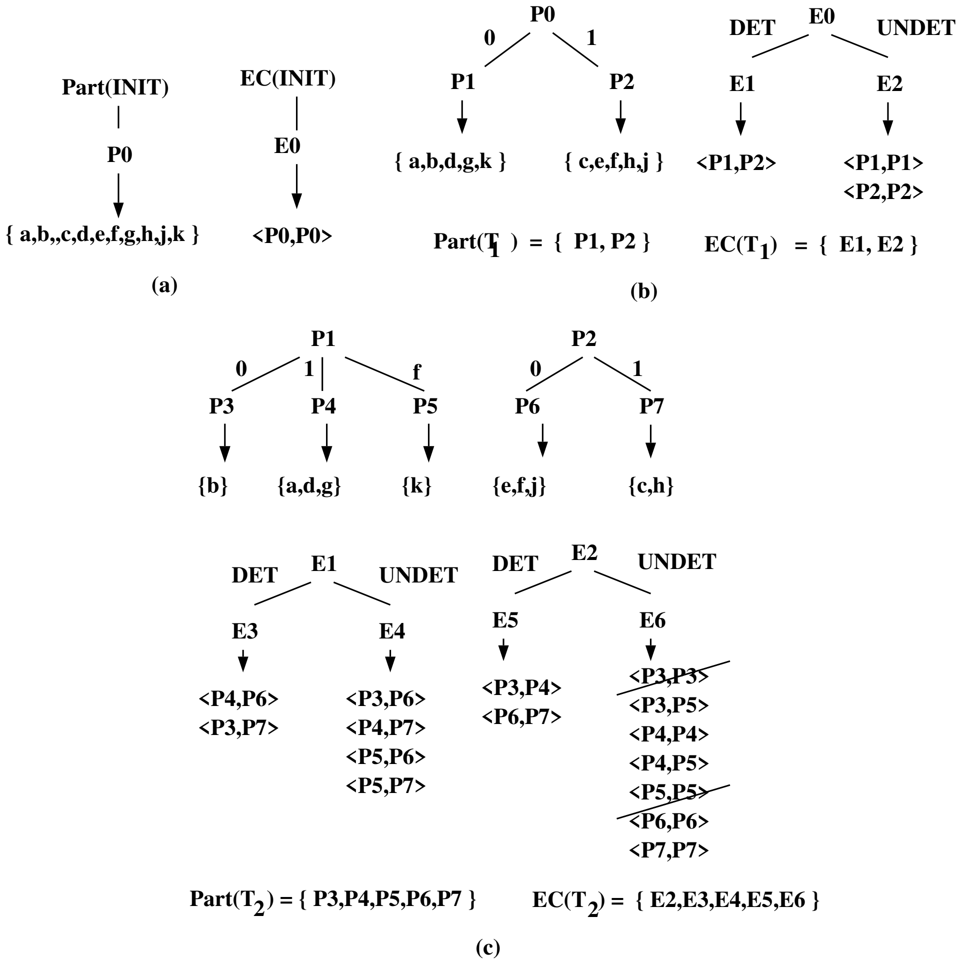

Figure 2: Example Illustrating UpdateEquivClass

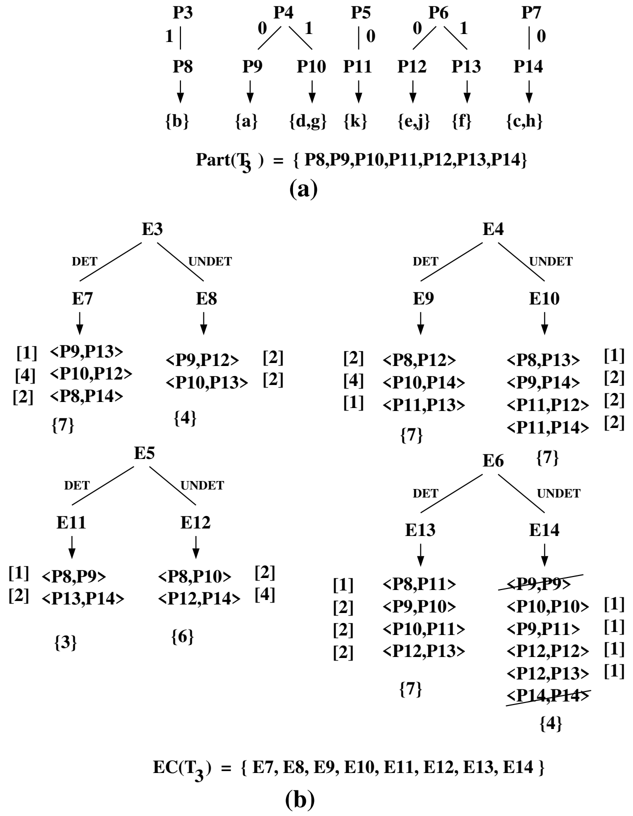

Figure 3: Example Illustrating UpdateEquivClass (Contd)

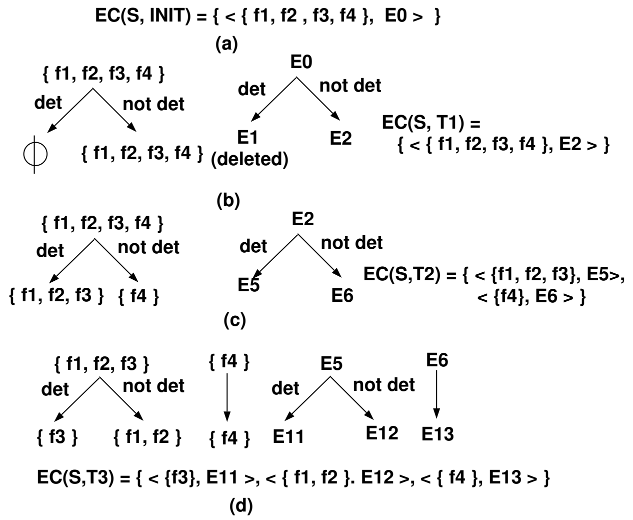

Figure 4: Example Illustrating Approximate Diagnostic Simulation

	a	b	c	d	e	f	g	h	j	k
$T1$	0	0	1	0	1	1	0	1	1	0
$T2$	1	0	1	1	0	0	1	1	0	f
$T3$	0	1	0	1	0	1	1	0	0	0

Table 1: Example Test Set.

Circuit	Vectors	Sample	Gates	Nodes	METAL	ALL
C432	47	466	196	466	19110	108345
C499	53	1027	243	1027	29403	526851
C880	53	935	443	935	97903	436645
C1355	85	1163	587	1163	171991	675703
C1908	120	1594	913	1594	416328	1269621
C2607	105	2795	1502	2795	1127251	3904615
C3540	153	3579	1719	3579	1476621	6402831
C5315	113	5496	2485	5496	3086370	15100260
C6288	28	5088	2448	5088	2995128	12941328
C7552	210	7371	3719	7371	6913621	27162135
S208.1	34	234	122	234	7381	27261
S382	36	365	182	365	16471	66430
S386	71	478	172	478	14706	114003
S420.1	75	494	252	494	31626	121771
S444	32	403	205	403	20910	81003
S510	62	512	236	512	27730	130816
S526	59	553	217	553	23436	152628
S526n	59	552	218	552	23653	152076
S641	59	696	433	696	93528	241860
S713	55	756	447	756	99681	285390
S820	114	916	312	916	48516	419070
S298	33	308	136	308	9180	47278
S838.1	103	1014	512	1014	130816	513591
S953	86	873	440	873	96580	380628
S1196	114	1260	561	1260	157080	793170
S1238	149	1319	540	1319	145530	869221
S1423	71	1589	748	1589	279378	1261666
S1488	127	1951	667	1951	222111	1902225
S1494	119	1965	661	1965	218130	1929630
S5378	251	4665	2993	4665	4477528	10878780
S344	22	346	184	346	16836	59685
S349	23	351	185	351	17020	61425
S382	36	365	182	365	16471	66430
S9234	367	9604	5844	9604	17073246	46113606
S13207.1	469	13491	8651	13491	37415575	90996795
S15850	440	16585	10383	16585	53898153	137522820

Table 2: Circuit Characterisitics

Circuit	DR			ERS		
	EXACT	APPROX	ERROR	EXACT	APPROX	ERROR
432	78.76	75.97	3.54	1.41	1.46	3.55
499	66.45	64.76	2.54	3.07	2.99	2.61
880	58.82	60.53	2.91	2.07	1.91	7.73
1355	65.36	66.55	1.82	2.03	1.97	2.96
1908	7.47	7.15	4.28	23.16	21.03	9.20
2670	13.45	12.77	5.06	9.36	9.21	1.60
3540	15.53	15.48	0.32	20.81	19.59	5.86
5315	27.32	27.13	0.70	8.56	7.15	16.47
6288	77.11	76.30	1.05	1.71	1.73	1.17
7552	19.92	19.54	1.91	12.64	11.90	5.85
s208.1	43.04	39.74	7.67	2.96	2.84	4.05
s298	29.53	32.79	11.04	3.41	2.86	16.13
s344	43.93	41.62	5.26	3.50	3.80	8.57
s349	44.44	43.59	1.91	3.69	3.88	5.15
s382	38.08	38.36	0.74	2.99	2.92	2.34
s386	56.80	62.97	10.86	1.86	1.68	9.68
s420.1	46.78	44.33	5.24	2.81	2.51	10.68
s444	41.01	39.95	2.58	3.15	2.95	6.35
s510	91.82	87.89	4.28	1.12	1.12	0.00
s526	40.78	40.51	0.66	2.91	2.97	2.06
s526n	38.94	37.50	3.70	3.02	2.73	9.60
s641	7.70	7.90	2.60	16.57	13.24	20.10
s713	7.21	8.33	15.53	19.77	17.20	13.00
s820	53.49	53.82	0.62	2.17	2.23	2.76
s832	59.42	62.10	4.51	1.89	1.80	4.76
s838.1	13.77	14.20	3.12	251.96	239.46	4.96
s953	81.79	81.44	0.43	1.27	1.21	4.72
s1196	60.84	61.82	1.61	2.19	2.01	8.22
s1238	82.45	80.90	1.88	1.25	1.35	8.00
s1423	47.52	46.63	1.87	10.99	9.17	16.56
s1488	64.80	63.60	1.85	1.74	1.66	4.60
s1494	70.06	71.25	1.70	1.60	1.52	5.00
s5378	11.79	11.60	1.61	18.06	16.72	7.42
s9234		41.62			3.80	
s13207.1		6.58			68.03	
s15850		5.53			396.66	
AVE			3.50			7.02
STD DEV			3.40			4.91

Table 3: Comparison of Exact and Approximation results for METAL

Circuit	DR			ERS		
	EXACT	APPROX	ERROR	EXACT	APPROX	ERROR
432	49.63	51.07	2.90	20.48	16.36	20.12
499	28.86	28.72	0.49	33.13	45.94	38.67
880	40.44	41.28	2.08	4.30	3.48	19.07
1355	30.47	31.04	1.87	6.71	7.84	16.84
1908	6.70	5.90	11.94	37.30	29.34	21.34
2670	12.38	12.70	2.58	24.61	21.37	13.17
3540	16.04	15.93	0.69	30.39	20.03	34.09
5315		22.40			19.05	
6288	24.95	24.33	2.48	5.71	6.60	15.59
s208.1	16.49	15.39	6.67	10.35	9.10	12.08
s344	16.43	15.32	6.76	7.76	5.45	29.77
s349	17.68	19.94	12.78	8.61	8.90	3.37
s382	29.01	27.94	3.69	5.33	6.02	12.95
s386	21.12	20.92	0.95	10.04	10.30	2.59
s420.1	20.83	20.85	0.10	13.14	12.64	3.81
s444	26.78	29.78	11.20	9.84	12.06	22.56
s510	45.62	48.24	5.74	7.03	6.45	8.25
s526	34.72	36.17	4.18	7.96	5.79	27.26
s526n	35.24	30.98	12.09	7.83	9.66	23.37
s641	10.37	12.64	21.89	18.56	15.95	14.06
s713	9.28	8.33	10.24	30.47	30.60	0.43
s820	34.04	30.90	9.22	26.61	20.59	22.62
s832	34.72	33.73	2.85	27.94	20.01	28.38
s838.1	10.95	11.24	2.65	2014.14	2122.20	5.37
s953	41.78	37.46	10.34	10.05	12.60	25.37
s1196	38.27	38.49	0.57	8.05	6.05	24.84
s1238	42.79	40.86	4.51	13.25	16.81	26.87
s1423	25.40	25.61	0.83	17.68	16.03	9.33
s1488	23.00	22.91	0.39	29.99	31.69	5.67
s1494	23.60	24.02	1.78	30.06	33.21	10.48
s5378		21.35			15.64	
s9234		3.63			5032.54	
s13207.1		7.46			67.63	
s15850		6.62			1753.60	
AVE			5.33			17.18
STD DEV			5.19			10.13

Table 4: Comparison of Exact and Approximation results for ALL

Circuit	METAL			ALL		
	EXACT	APPROX	RATIO	EXACT	APPROX	RATIO
C432	7.98	1.98	4.03	62.97	7.97	7.90
C499	9.99	2.97	3.36	350.99	45.98	7.63
C880	28.99	4.98	5.82	256.98	21.99	11.69
C1355	53.94	9.98	5.40	668.98	38.97	17.17
C1980	264.97	66.98	3.96	1241.97	17.98	69.08
C2670	732.95	32.96	22.24	3251.99	120.97	26.88
C3540	1074.96	49.99	21.50	8344.97	254.97	32.73
C5315	2385.95	93.99	25.39		607.96	
C6288	979.96	87.98	11.14	4229.99	360.97	11.72
C7552	11089.98	204.99	54.10			
S208.1	1.98	0.03	66.00	11.98	1.98	6.05
S298	1.96	0.02	98.00	20.98	2.97	7.06
S344	3.97	0.98	4.05	14.97	1.96	7.64
S349	3.98	0.95	4.19	18.97	2.97	6.39
S382	5.96	0.98	6.08	33.98	3.99	8.52
S386	7.96	1.97	4.04	122.98	9.98	12.32
S420.1	17.98	2.98	6.03	109.97	12.99	8.47
S444	7.98	0.99	8.06	33.96	3.98	8.53
S510	11.98	1.99	6.02	104.98	12.96	8.10
S526	9.97	1.97	5.06	108.99	11.97	9.11
S526n	10.97	1.97	5.57	124.98	10.99	11.37
S641	25.95	3.96	6.55	114.97	9.98	11.52
S713	24.96	3.98	6.27	113.94	10.98	10.38
S820	48.97	6.96	7.04	702.98	58.98	11.92
S832	44.99	7.99	5.63	661.97	65.99	10.03
S838.1	55.99	6.99	8.01	264.99	50.97	5.20
S953	70.98	6.98	10.17	396.96	36.98	10.73
S1196	116.99	15.98	7.32	1455.98	86.98	16.74
S1238	120.97	16.97	7.13	1777.96	113.99	15.60
S1423	112.98	14.99	7.54	942.97	51.99	18.14
S1488	142.97	19.97	7.16	3701.99	179.98	20.57
S1494	141.98	19.98	7.11	3929.98	196.00	20.05
S5378	5326.99	153.98	34.60		702.00	
S9234		472.97			2427.97	
S13207.1		1433.98			4528.98	
S15850		2176.95			7472.97	
AVE			14.68			14.31
STD DEV			20.67			12.10

Table 5: Comparison of time required for exact and approximate computation