Reducing Fault Dictionary Size for Million-Gate Large Circuits

YU-RU HONG
Purdue University
and
JUINN-DAR HUANG
National Chiao Tung University

In general, fault dictionary is prevented from practical applications in fault diagnosis due to its extremely large size. Several previous works are proposed for the fault dictionary size reduction. However, some of them fail to bring down the size to an acceptable level, and others might not be able to handle today's million-gate circuits due to their high time and space complexity. In this article, an algorithm is presented to reduce the size of pass-fail dictionary while still preserving high diagnostic resolution. The proposed algorithm possesses low time and space complexity by avoiding constructing the huge distinguishability table, which inevitably boosts up the required computation complexity. Experimental results demonstrate that the proposed algorithm is capable of handling industrial million-gate large circuits in a reasonable amount of runtime and memory.

Categories and Subject Descriptors: B.2.3 [**Reliability, Testing, and Fault-Tolerance**]: Diagnostics

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Fault dictionary, fault diagnosis, diagnostic resolution

ACM Reference Format:

Hong, Y.-R. and Huang, J.-D. 2009. Reducing fault dictionary size for million-gate large circuits. ACM Trans. Des. Autom. Elect. Syst., 14, 2, Article 27 (March 2009), 14 pages, DOI = 10.1145/1497561.1497570 http://doi.acm.org/10.1145/1497561.1497570

1. INTRODUCTION

Semiconductor technology advances constantly so that electronic products with higher performance and versatile functions can be provided to end users. However, denser and finer fabrication processes make chips vulnerable to defects,

Authors' addresses: Y.-R. Hong, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907; email: yujuhong@purdue.edu; J.-D. Huang, Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan; email: jdhuang@mail.nctu.edu.tw.

DOI 10.1145/1497561.1497570 http://doi.acm.org/10.1145/1497561.1497570

leading to lower manufacturing yield and reliability. Fault diagnosis techniques help physically locate the faults and determine the causes of failures in the manufacturing process. Based on the diagnosis results, manufacturers can refine their processes and designers can modify their designs to improve the yield of chips. Conventionally, diagnosis techniques are classified into two major categories, effect-cause and cause-effect analysis. Dynamic diagnosis is an effect-cause analysis which observes the faulty responses of the circuit under test (CUT) and then deduces the cause of error based on the comparison against the fault-free response. There are many research works devoted to dynamic diagnosis [Kocan and Saab 1999, Li et al. 1990, Venkataraman et al. 1996].

For the cause-effect analysis, the diagnosis procedure first selects a specific fault model to represent how the effects of physical faults affect the behavior of circuits. The fault simulator iteratively injects a modeled fault into the circuit, applies a test vector at the primary inputs, and simulates the circuit to obtain the responses. A *fault dictionary* is a database constructed during fault simulation to store the output responses in the presence of every modeled fault under every test vector. The corresponding sequence (combination of responses of every test vector) of a modeled fault in the fault dictionary is called a *fault signature*. By looking up in the precomputed fault dictionary for the observed output responses of the CUT, faults can be recognized and hence located. A set of faults share the same signature under a test set can not be distinguished. These faults form a fault equivalence class, limiting the *diagnostic resolution* of the test set. Here the diagnostic resolution (DR) is defined as the ratio of distinguishable fault pairs over all fault pairs under the fault model.

Nevertheless, two obstacles generally prevent fault dictionaries from practical applications: (1) Creating a dictionary with high diagnostic resolution requires long computation time. (2) The dictionary is usually too large and thus impractical to be used in the diagnosis process. The first obstacle has been evaluated by Pomeranz and Reddy [1992]. They show that only a small number of diagnostic runs are sufficient to compensate for the nonrecurring effort of creating a fault dictionary. Therefore, fault dictionary is still attractive as long as its size can be kept small. Unfortunately, as the circuit size increases, the memory requirement of fault dictionary grows so rapidly and becomes unacceptable.

Though many research works [Ryan et al. 1993, Boppana and Fuchs 1994] are contributed to tackle the second obstacle, some of them are still unable to bring down the dictionary size to an acceptable level, while others consume too much time or space to be applied on real industrial circuits. Arslan and Orailoglu [2002] propose to further convert a *pass-fail* dictionary into an XORed dictionary in which only one combined signature is stored for each partition of test vectors, and then select a limited number of signatures from it. This approach accomplishes a great reduction on the size of the fault dictionary. But during the selection process, it requires to construct the *distinguishability table* with space complexity up to $O(|T|*|F|^2)$, where $|T|$ and $|F|$ are the number of test vectors and faults. The complexity is too high to be acceptable for today's million-gate scale circuits.

In this article, we focus on size reduction for pass-fail fault dictionary. Given a precomputed pass-fail dictionary, we propose an *edge factor*-based algorithm to efficiently reduce the size of original dictionary by extracting signatures of a small subset of test vectors (or partitions) from it. The proposed edge factor eliminates the need to explicitly build a time- and space-consuming distinguishability table. As a result, our algorithm possesses low time and space complexity. The experimental results show that the edge factor based algorithm, when applied to an XORed dictionary, is able to deliver comparable results against the previous work [Arslan and Orailoglu 2002] but in a much shorter runtime and with much lower memory requirement.

This article makes the following contributions. 1) We address one of the major problems of fault dictionary, the huge size, and propose an edge factor based algorithm to efficiently perform size reduction for pass-fail dictionary. Using cause-effect diagnostic approaches with fault dictionary can thus be more promising. 2) We introduce the edge factor and prove that it is strictly proportional to the number of undistinguished fault pairs so that the resource-hungry distinguishability table is longer needed.

2. RELATED WORKS

For each modeled fault, a full fault dictionary stores the full response for each test vector applied to the CUT. It possesses the largest size of all kinds of fault dictionaries; the size is $O(R*|T|*|F|)$ for a full fault dictionary containing R primary outputs, $|T|$ test vectors, and $|F|$ faults. A pass-fail dictionary discards all output responses and stores only an entry for each fault-test pair to denote whether the test vector can detect the fault. Usually, a 0 stands for an undetected fault, and a 1 stands for a detected fault for the test vector. The size of a pass-fail dictionary is $O(|T|*|F|)$.

On top of the pass-fail dictionary, Arslan and Orailoglu [2002] propose a size reduction method that partitions the test set and stores a combined signature for each partition. To locate the modeled faults, the diagnosis process needs to partition all modeled faults into distinctive fault equivalence sets. With the application of more test vectors, the length of the fault signatures increases, and the fault equivalence sets break down into more smaller sets. Full resolution is achieved when each set contains only a single fault. Ideally, if every test vector equally partitions the faults in a way different from the former ones, $\lceil \log_2|F| \rceil$ test vectors are enough to distinguish all faults. This rarely happens because a test vector usually can detect only a small portion of faults. (There are more 0s than 1s in a row in the dictionary.) By taking advantages of this feature, the authors introduce an *XORed pass-fail dictionary*, storing the combined fault signatures. Let the i-th row of a pass-fail dictionary indicates which faults the test vector t_i can detect. The XORed dictionary has the same first row as the pass-fail dictionary, but its i-th row ($i > 1$) is the result of XORing its ($i - 1$)-th row with the i-th row in the pass-fail dictionary. Thus the i-th row in the XORed dictionary is a combined signature for the test vectors 1 to i. After XORing, the number of 1s increases and with more balanced numbers of 1s and 0s, the combined signatures can distinguish the faults better. Hence it is reasonable to expect that a compacted XORed dictionary can have higher diagnostic resolution than directly compacted dictionary. [Arslan and Orailoglu [2002] show that a

	f_1	f_2	f_3	f_4	f_5
t_1	0	0	0	1	1
t_2	0	1	1	1	1
t_3	1	1	0	0	1
t_4	0	1	1	0	0

Fig. 1. Pass-fail dictionary D1.

dictionary can be compacted to contain only $\lceil \log_2|F| \rceil$ combined signatures with a minor loss in diagnostic resolution. This compaction process, however, builds a distinguishability table with the space complexity $O(|T|*|F|^2)$, inevitably elevating its time complexity to the same level. The authors neither detail any algorithm to accelerate the construction of the table nor provide experimental results on large circuits. It is doubtful whether the method can deal with millions of possible faults. Therefore, a more efficient method for dictionary size reduction is demanded to handle large-scale circuits.

3. PRELIMINARIES

We define several terms used in this section. Also note that the proposed edge factor based algorithm, to be described later, can be applied on either an original or an XORed pass-fail dictionary. Without loss of generality, we simply assume that all operations are performed on an original pass-fail dictionary throughout Sections 3 and 4.

Given a fault set F and a test vector set T, a pass-fail dictionary D is defined as a |T|x|F| matrix where

$$\begin{cases} D_{ij} = 1, \text{if } t_i \text{ can detect } f_j & 1 \le i \le |T|, 1 \le j \le |F| \\ D_{ij} = 0, \text{otherwise} & 1 \le i \le |T|, 1 \le j \le |F|. \end{cases}$$

Figure 1 shows an example of a pass-fail dictionary D1. For example, the test vector t_1 can detect two faults f_4 and f_5, while the test vector t_2 can detect f_2, f_3, f_4, and f_5. Based on the pass-fail dictionary, we construct a subset of T, the selected test set T_S, which includes vectors to be stored in the resultant reduced fault dictionary. Our proposed method aims at efficiently constructing a small but effective T_S. In this article, T_S is considered as an ordered set for convenience in the test vector selection phase, though the order does not actually affect the diagnostic resolution of the selected test set.

Once the T_S is created, the corresponding fault equivalence sets are formed. A fault equivalence set F_E is a set of faults in which all fault pairs are indistinguishable with respect to T_S. For example, given D1 in Figure 1, assume $T_S = \{t_1\}$; then there are two fault equivalence sets, $F_{E1} = \{f_1, f_2, f_3\}$ and $F_{E2} = \{f_4, f_5\}$. The distinguishability information can be presented using a table by first defining a fault pair set P as

$$P = \{fp_k = (f_i, f_j) | f_i, f_j \in F, i < j\}.$$

	(f_1,f_2)	(f_1,f_3)	(f_1,f_4)	(f_1,f_5)	(f_2,f_3)	(f_2,f_4)	(f_2,f_5)	(f_3,f_4)	(f_3,f_5)	(f_4,f_5)
t_1	0	0	1	1	0	1	1	1	1	0
t_2	1	1	1	1	0	0	0	0	0	0
t_3	0	1	1	0	1	1	0	0	1	1
t_4	1	1	0	0	0	1	1	1	1	0

Fig. 2. Distinguishability table A1.

Then, the distinguishability table A is a |T|x|P| matrix, where

$$\begin{cases} A_{ij} = 1, \text{if } t_i \text{ can detect } f_{Pj} \\ D_{ij} = 0, \text{otherwise.} \end{cases}$$

A1 in Figure 2 is the distinguishability table obtained from D1 in Figure 1. The row of t_1 in A1 with 4 0s that indicate fault pairs (f_1, f_2), (f_1, f_3), (f_2, f_3), (f_4, f_5) are indistinguishable for t_1. Hence, the dictionary size reduction problem can be translated into finding a minimum number of rows in the distinguishability table to cover all fault pairs. This set-cover problem is a well-known NP-complete problem [Cormen et al. 2003]. Even worse, due to the extremely high space complexity $O(|T|*|F|^2)$ of the distinguishability table, a circuit with one million faults leads to a table whose size is of 10^{15} order even if only 1000 test vectors are present.

As in [Arslan and Orailoglu 2002], a greedy algorithm, Greedy-Set-Cover, is adopted to deal with this NP-complete problem. This algorithm selects the test vector that distinguishes most new fault pairs in every iteration. Take A1 as an example, the algorithm ends up with $T_S = \{t_1, t_3, t_2\}$ after three iterations. The time complexity of this algorithm is $O(N*|T|*|F|^2)$, where N is the size of T_S. Though the greedy algorithm seems simple, the complexity introduced by the distinguishability table still keeps it far from practical.

4. PROPOSED METHOD

The key to achieving low complexity is to avoid the construction of the distinguishability table during test selection process. In this section, we review the problem from the perspective of graph, derive an *edge factor* for the selection process, present our algorithm, and provide implementation details and complexity analysis.

4.1 Distinguishability Graph and Edge Factor

We start from transforming the distinguishability table into a *distinguishability graph* G and then identify several important properties which serve as the keystones of our algorithm. Given a fault set F, a selected test set T_S, and the corresponding distinguishability table A, the distinguishability graph G = (V, E) is defined as

$$V = F = \{f_i | 1 \leq i \leq |F|\}$$
$$E = \left\{ e_{ij} | fp_m = (f_i, f_j) \, and \sum_{t_k \in T_S}^{A_{km}} = 0 \right\}.$$

Each vertex in the distinguishability graph represents a fault. If no test vector in T_S can distinguish the fault pair (f_i, f_j) in the distinguishability table, there exists an edge e_{ij} in the graph. The graph G can be constructed by the following steps. First, construct a complete graph G with |F| vertices. Next, remove the edge e_{ij} in G if the fault pair (f_i, f_j) is distinguished by any test vector in T_S. The resultant distinguishability graph has the following properties:

Property 1. A distinguishability graph is a complete graph if $T_S = \emptyset$.

Property 1 is trivial since an empty test set is incapable of distinguishing any fault pair, so every vertex pair should be connected by an edge. It is the initial condition at which the test selection process just begins.

Property 2. In a distinguishability graph, there is an edge e_{ij} between v_i and v_j if and only if the corresponding fault pair (f_i, f_j) is indistinguishable under the given T_S.

Property 2 comes directly from the definition of the distinguishability graph.

Property 3. The given T_S partitions G into a set of disjoint connected components. Faults within the same connected component form a fault equivalence set.

Property 3 says that after removing the edges representing distinguishable fault pairs according to the distinguishability table, G is broken into a set of disjoint connected components. Because a test vector can only distinguish the detected faults from the undetected ones, it applies a cut on G. Edges across the cut are then removed. The remaining edges represent the indistinguishability among the faults within a connected component, in which the vertices (faults) form a fault equivalence set.

Property 4. In a distinguishability graph G, each connected component, representing a fault equivalence set, is a complete graph.

Property 4 is based on the definition of fault equivalence set that every two faults are indistinguishable in this set. In other words, there should be edges between all vertex pairs within a connected component. Hence every connected component in G must be a complete graph.

The above properties can be examined using the example shown in Figure 3, derived from the same example given in Section 3. As mentioned earlier, the greedy algorithm in Section 3 always selects the best test vector covering the most fault pairs in each iteration. From the perspective of distinguishability graph, the best vector should remove the most edges when added into the current T_S. Figure 4 demonstrates the effectiveness of two test vectors, t_1 and t_2, as candidates for joining an empty test set. The vector t_1 can remove more edges and thus is a better choice than t_2. Recall that for the distinguishability table, the goal of the greedy algorithm is to minimize the uncovered 0s; for the distinguishability graph, the goal becomes to minimize the remaining edges.

Based on the above properties, the number of remaining edges in the distinguishability graph for a given selected test set T_S can be calculated as follows. Assume T_S partitions all faults into n fault equivalence sets, $F_{E1}, F_{E2}, F_{E3}, \ldots, F_{En}$. From Property 4, each fault equivalence set is represented as a complete

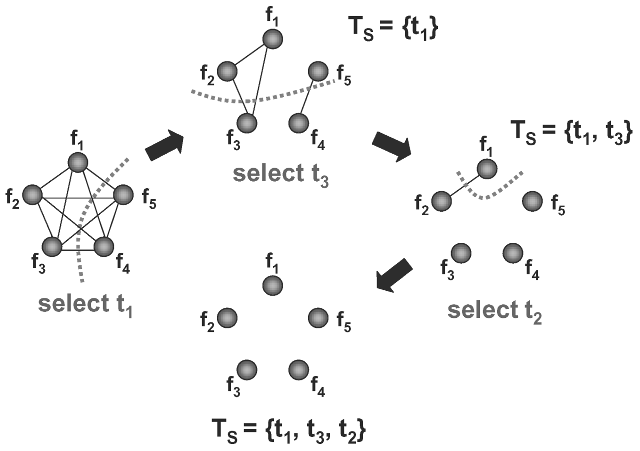

Fig. 3. Selection process resulting in a set of complete connected components.

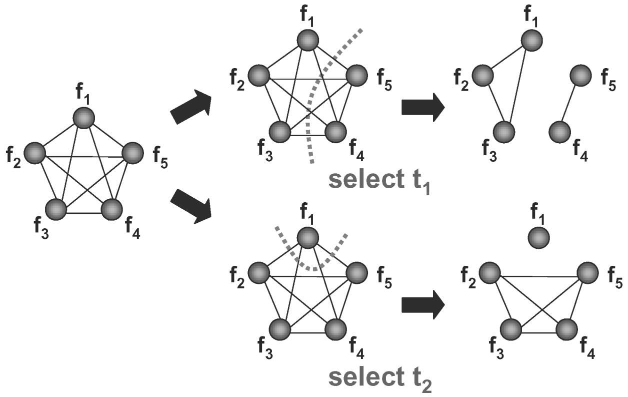

Fig. 4. Effectiveness of different test vectors.

connected component in the distinguishability graph. Hence, the number of remaining edges in the graph is

$$
\begin{aligned}
&\binom{|F_{E1}|}{2} + \binom{|F_{E2}|}{2} + \cdots + \binom{|F_{En}|}{2} \\
&= \frac{|F_{E1}|(|F_{E1}|-1)}{2} + \frac{|F_{E2}|(|F_{E2}|-1)}{2} + \cdots + \frac{|F_{En}|(|F_{En}|-1)}{2} \\
&= \frac{|F_{E1}|^2 + |F_{E2}|^2 + \cdots + |F_{En}|^2 - (|F_{E1}| + |F_{E2}| + \cdots + |F_{En}|)}{2} \\
&= \frac{EF(T_s) - |F|}{2} \quad \text{where } EF(T_s) = \sum_{i=1}^{n} |F_{Ei}|^2.
\end{aligned}
$$

Edge Factor, EF, is introduced and defined in the preceding. Note that a similar approach to calculate the number of undetected bridging faults is also used in Thadikaran et al. [1997]. Since |F| is a constant, the number of remaining edges is solely determined by the edge factor, which is the square sum of the

```
EF-Based(D, N) {
  $T_S \leftarrow \emptyset$
  do
    for i $\leftarrow$ 1 to |T|
      if $t_i \notin T_S$
      then
        $Y_i \leftarrow T_S \cup \{t_i\}$
        calculate EF($Y_i$)based on D
      end
    end
    identify $t_j$ with the smallest EF
    $T_S \leftarrow T_S \cup \{t_j\}$
  while (EF($T_S$) > |F| and |$T_S$| < N)
  return $T_S$
}
```

N is the given upper bound of the number of selected vectors
T_S is the set of selected test vectors; Y_i is a temporary test set

Fig. 5. The pseudo-code of the proposed EF-based algorithm.

cardinalities of all fault equivalence sets. That is, if we know the number of fault equivalence sets with their cardinalities; neither the distinguishability table nor the distinguishability graph is required for the EF calculation. In fact, the calculation can be done in O(|F|) time with implementation techniques in Section 4.2. A similar problem also exists for two-line single bridging faults (TSBFs). The number of line pairs for all possible TSBFs is too huge to be completely enumerated. Based on a similar notion, [Chakravarty et al. 1992, 1996] propose a specific data structure to efficiently represent those TSBFs without explicit fault enumeration. Our EF-based algorithm is outlined in Figure 5.

4.2 Implementation Techniques

Calculation of EFs dominates the computation complexity of the EF-based algorithm and hence special cares need to be taken in the implementation. Obtaining EF for a given T_S requires two tasks: 1) find which fault equivalence set each fault belongs to; 2) find the cardinality of each fault equivalence set. For the first task, a Set IDentification (SID) of a fault f_i with respect to T_S is a vector defined as

$$SID_{f_i|T_s} = [w_1 w_2 w_3 \cdots w_{|T_s|}]$$

where w_k indicates whether the k-th test vector in T_S can detect f_i. SID can be easily obtained by extracting and concatenating the corresponding rows of the selected test vectors from the pass-fail dictionary in the order of how they are selected. Since each bit within the SID vector records whether the fault is detected by a specific test vector, the length of SID depends on the size of T_S. Faults with the same SID belong to the same fault equivalent set because they can not be distinguished under the given T_S. Figure 6 gives an example of the SIDs for a given T_S from the pass-fail dictionary in Figure 1.

	f_1	f_2	f_3	f_4	f_5
$T_S = \{t_1\}$	0	0	0	1	1
$T_S = \{t_1, t_3\}$	01	01	00	10	11
$T_S = \{t_1, t_3, t_2\}$	010	011	001	101	111

Fig. 6. SIDs corresponding to growing selected test sets.

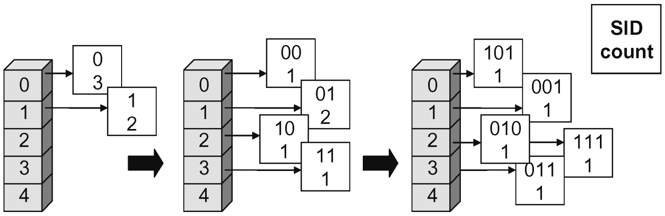

Fig. 7. Hash table facilitating the computation of EF.

For the second task, a chained hash table of size M is constructed to count the number of faults in each fault equivalence set, where M is selected as a prime number slightly larger than |F|. Each node in the hash table contains a key, which is an SID, and a counter. Our algorithm scans all the faults one by one, calculates their SIDs under a given T_S, and then searches for their SIDs in the hash table. If there is a search hit, the counter of the associated node is incremented by 1. This implies the current fault belongs to an existing fault equivalence set. Otherwise, the algorithm creates a new node and attaches it to the end of the linked list with a key as the SID and a counter initialized to 1. Thus each node has a unique key and the number of nodes is exactly the same as the number of fault equivalence sets. Note that there are at most min($2^{|Ts|}$,|F|) nodes in the hash table because the maximum number of equivanlence sets is |F| . Typically, min($2^{|Ts|}$,|F|) is bounded to |F| as T_S becomes large enough. Hence, a hash table with size M > |F| ensures the loading density < 1 all the time. Empirically, this implies on average 1.5 search is sufficient to check whether an SID is in the hash table or not [Horowitz et al. 1995]. After constructing the hash table, every node in the hash table represents a unique fault equivalence set and the associated counter is its cardinality. Figure 7 illustrates the constructing process for T_S in Figure 6. The last step is traversing the hash table and calculating the EF(T_S). Both of these two steps take about O(|F|) time.

It is now clear that neither the distinguishability table nor the distinguishability graph needs to be explicitly constructed in our algorithm. Instead, the edge factor guides the iterative test vector selection process in EF-based. By using the above implementation, it takes merely O(|F|) time to obtain the EF for a given test set. So given an upper bound of the number of selected test vectors, N, the time complexity of EF-Based is O(N*|T|*|F|). In this paper, N is set to $\lceil \log_2 |F| \rceil$ for maximum dictionary size reduction. Therefore, the time complexity is only O(|T|*|F|*$\log_2$|F|). Moreover, the length of an SID is at most

Table I. Diagnostic Resolution of the ISCAS'85 Benchmark Circuits

Circuit	$\|T\|$	$\|F\|$	$\lceil \log_2 \|F\| \rceil$	Reduction ratio of FD size	DR before reduction	DR after reduction	DR in [Arslan et al. 2002]
c432	50	524	10	0.800	0.995877	0.989374	0.993403
c499	53	758	10	0.811	0.999456	0.997121	0.997257
c880	54	942	10	0.815	0.999621	0.996950	0.996970
c1355	86	1574	11	0.872	0.999241	0.997302	0.997274
c1908	120	1879	11	0.908	0.999403	0.997000	0.997063
c2670	106	2747	12	0.887	0.997739	0.996207	0.996088
c3540	150	3428	12	0.920	0.998169	0.996568	0.996694
c5315	125	5350	13	0.896	0.999776	0.999268	0.999083
c6288	30	7744	13	0.567	0.999815	0.999578	0.999555
c7552	217	7550	13	0.940	0.999566	0.998992	0.998871

$O(\log_2|F|)$. Hence the hash table requires only $O(|F|*\log_2|F|)$ space. The size of the given pre-computed pass-fail dictionary D is $O(|T|*|F|)$, which determines the overall space complexity.

5. EXPERIMENTAL RESULTS

The proposed algorithm is implemented in C++ with the details described in Section 4. All experiments are conducted on a dual-core Intel Xeon 2.0GHz platform. For better diagnostic resolution, all fault dictionaries are preprocessed by XORing the responses of the test vectors just as in Arslan and Orailoglu [2002].

5.1 Diagnostic Resolution

Diagnostic resolution (DR) is defined as the ratio of distinguishable fault pairs over all possible fault pairs. For each circuit in the ISCAS'85 and '89 benchmark sets [Brglez and Fujiwara 1985; Brglez et al. 1989], we generate the original fault dictionary through the Atalanta test pattern generator [Lee and Ha 1993] and HOPE fault simulator [Lee and Ha 1992]. In order to meet the industrial practice, the fault collapsing techniques in Atalanta are turned on. Thus we report only the number of collapsed faults and preclude all the undetected faults.

Table I lists the diagnostic resolutions of the ISCAS'85 benchmark circuits. The second and third columns denote the number of test vectors and faults, respectively. The size of the selected test set T_S is set to $\lceil \log 2|F| \rceil$ shown in the fourth column to fairly compare our results with those reported in Arslan and Orailoglu [2002]. The fifth column denotes the reduction ratio of fault dictionary size. The sixth and seventh columns denote the diagnostic resolutions before reduction and after reduction. The last column denotes the diagnostic resolutions reported in Arslan and Orailoglu [2002]. On average, the size of dictionary is reduced by 84.16% while the DR loss is merely 0.203%. Our diagnostic resolutions are comparable to those reported in Arslan and Orailoglu [2002].

Table II shows the diagnostic resolutions of the ISCAS'89 benchmark circuits in the similar format to that in Table I. However, due to the lack of experimental

Table II. Diagnostic Resolution of the ISCAS'89 Benchmark Circuits

Circuit	|T|	|F|	$\lceil \log_2 \|F\| \rceil$	Reduction ratio of FD size	DR before reduction	DR after reduction
s9234	384	6927	13	0.966	0.995602	0.993234
s13207	460	9815	14	0.970	0.999591	0.998303
s15850	438	11725	14	0.968	0.998871	0.997933
s35932	68	39094	16	0.765	0.989591	0.989422
s38417	901	31180	15	0.983	0.999952	0.999587
s38584	654	36303	16	0.976	0.998275	0.997943

Table III. Runtime and Memory Usage of the ISCAS'89 Benchmark Circuits

Circuit	|T|	|F|	$\lceil \log_2 \|F\| \rceil$	CPU (s)	Mem (MB)
s9234	384	6927	13	1.8	2
s13207	460	9815	14	3.3	4
s15850	438	11725	14	3.8	6
s35932	68	39094	16	2.0	6
s38417	901	31180	15	22.4	28
s38584	654	36303	16	20.7	24

data on this benchmark set in Arslan and Orailoglu [2002], we are unable to compare our results with theirs. The average reduction ratio of test size is 93.8%, which is higher than that of Table I. It is predictable since $\lceil \log_2 |F| \rceil / |F|$ decreases as the number of faults increases. The average DR loss of Table II is only 0.091%. The DR loss is low, partly due to a well-known property that when the size of selected test set increases, the DR rises quickly at first and then slows down dramatically. Nonetheless, our algorithm still performs well considering that only $\lceil \log_2 |F| \rceil$ test vectors are selected. To sum up, our algorithm is capable of achieving a significant reduction ratio of fault dictionary size at the cost of a slight loss in DR. Meanwhile, it can produce the comparable DRs as those in Arslan and Orailoglu [2002] but with much lower time and space complexity.

5.2 Time and Space Efficiency

Table III gives the runtime and memory usage needed to process the ISCAS'89 benchmark circuits. The column CPU gives the runtimes in seconds for the EF-Based algorithm; the column Mem gives the memory usage in MB. The table shows that even the largest circuit can be processed within a minute and consumes at most 28MB memory. Due to the lack of runtime and memory information in Arslan and Orailoglu [2002], direct comparison of performance between their algorithm and ours is not available.

Nonetheless, the sizes of the circuits in the ISCAS'85 and '89 are still far away from those of today's SoC designs. To further validate the efficiency of our algorithm, we use an industrial digital signal processor (DSP) design for further evaluation. The generated fault dictionary of this circuit, as shown in Table IV, contains 766 test vectors and 627505 faults. The DR before reduction is 0.944663, and is 0.925079 after reduction when $\lceil \log_2 |F| \rceil$ test vectors are

Table IV. An Industrial DSP Design

$\lvert T\rvert$: 766, $\lvert F\rvert$: 627505, DR before reduction: 0.944663 Memory: around 500MB			
# of selected test vectors	Reduction ratio of FD size	DR after reduction	CPU (sec)
20 ($\lceil \log_2\lvert F\rvert \rceil$)	0.974	0.925079	451
40	0.948	0.938836	1153
60	0.922	0.942046	1727
80	0.896	0.943580	2330

Table V. Randomly Generated Large Dictionaries

$\lvert T\rvert$	$\lvert F\rvert$	$\lceil \log_2\lvert F\rvert \rceil$	CPU (sec)	Mem (MB)	$\lvert T\rvert$	$\lvert F\rvert$	$\lceil \log_2\lvert F\rvert \rceil$	CPU (sec)	Mem (MB)
1000	100000	17	98	96	1000	1100000	21	2783	1124
1000	200000	18	261	207	1000	1200000	21	3005	1225
1000	300000	19	505	303	1000	1300000	21	3237	1335
1000	400000	19	664	399	1000	1400000	21	3525	1431
1000	500000	19	844	492	1000	1500000	21	3683	1529
1000	600000	20	1249	606	1000	1600000	21	3897	1637
1000	700000	20	1464	716	1000	1700000	21	4127	1734
1000	800000	20	1657	812	1000	1800000	21	4327	1844
1000	900000	20	1867	922	1000	1900000	21	4581	1940
1000	1000000	20	2129	1018	1000	2000000	21	4790	2050

selected. The original DR is not high because 95.13% of the faults are marked as "hardly detected" in this circuit. The DR can be improved by selecting more test vectors (resulted in larger dictionaries). The results of selecting 20, 40, 60, and 80 test vectors, respectively, are shown in Table IV. The runtime is less than 39 minutes even when 80 vectors are selected. This experiment suggests the proposed algorithm can handle large real industrial circuits. It is worth noticing that the size of the distinguishability table for this circuit is at least 35TB, which is so large that it is nearly impossible for today's computer to construct and further operate on the table.

To perform more complete analysis of time and space complexity to support our claim of processing million-gate large circuits, we randomly generate 20 large pass-fail dictionaries. The ratio of 1 and 0 in these dictionaries is set to the average of ISCAS'85 and '89 benchmark circuits. Though the dictionaries are not derived from real circuits (there is no correlation among the faults), they are equally reliable for runtime and memory usage evaluation. Table V shows the results of the randomly generated circuits, of which the number of faults ranges from 0.1 million to 2 millions. Again, $\lceil \log_2|F| \rceil$ vectors are selected from each dictionary. All cases in the table are processed within 1.5 hour with memory usage lower than 2GB. Today's modest computers should be powerful enough to complete this job.

Figure 8 gives a clearer picture of the results shown in Table V. The vertical axes represent runtime and memory usage; the horizontal axis gives the number of faults. A diamond-shaped dot and a dash-shaped dot shows the runtime and the memory usage, respectively. These two lines fit perfectly for our time and space complexity analysis given in Section 4, which is $O(|T|*|F|*\log_2|F|)$ for

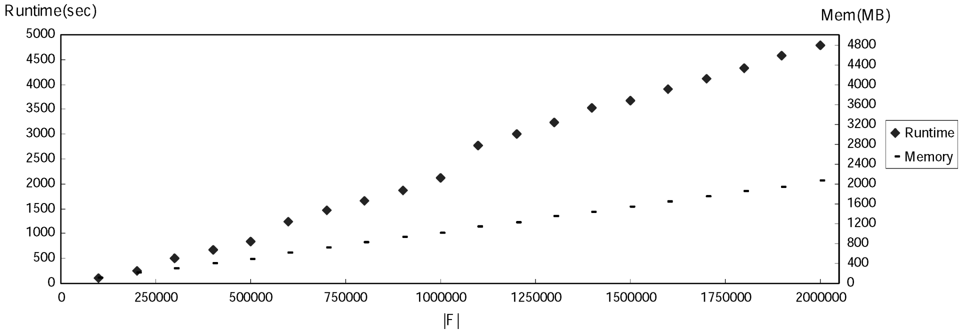

Fig. 8. Runtime and memory usage for large randomly generated dictionaries.

time and O(|T|*|F|) for memory. First, since the number of test vectors |T| is fixed, the memory usage is monotonically increasing with the number of faults, |F|. Second, the runtime can be best described as a discontinuous piecewise function of |F|, constituted by a few straight lines in Figure 8. Each individual straight line follows the trend of the given time complexity O(|T|*|F|*$\log_2$|F|). The discontinuities in the figure come from the selecting integral number of test vectors, $\lceil \log_2 |F| \rceil$. For example, discontinuities occur when the number of faults jumps from 0.5 million to 0.6 million and 1 million to 1.1 million. Figure 8 strongly supports our time and space complexity analysis in Section 4 and shows that the complexity is consistent for different sizes of dictionaries.

6. CONCLUSIONS

The large size of fault dictionary is a major reason that the dictionary-based fault diagnosis approaches are considered not practical. In this article, we propose a time- and space-efficient algorithm for size reduction of pass-fail fault dictionary. The edge factor, being strictly proportional to the number of indistinguishable fault pairs, is introduced as the guidance throughout the algorithm. With the specific implementation techniques, the edge factor can be calculated directly from the original pass-fail fault dictionary and thus the time- and memory-consuming construction of the distinguishability table and further operations on that table are completely excluded. Experimental results over a large set of various circuits validate the effectiveness and efficiency of the proposed algorithm. Hence, the proposed algorithm is capable of providing a feasible solution to fault dictionary size reduction for today's industrial million-gate circuits.

REFERENCES

ARSLAN, B. AND ORAILOGLU, A. 2002. Fault dictionary size reduction through test response superposition. In *Proceedings of the IEEE International Conference on Computer Design*. 480–485.

BOPPANA, V. AND FUCHS, W. K. 1994. Fault dictionary compaction by output sequence removal. In *Proceedings of the International Conference on Computer-Aided Design*. 576–579.

BRGLEZ, F. AND FUJIWARA H. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN. In *Proceedings of the International Symposium on Circuits and Systems.* 695–698.

BRGLEZ, F, BRYANT, D. AND KOZMINSKI, K. 1989. Combination Profiles of Sequential Benchmark Circuits. In *Proceedings of the International Symposium on Circuits and Systems*. 1929–1934.

CHAKRAVARTY, S. AND LIU, M. 1992. Algorithms for IDDQ measurement based diagnosis of bridging faults. in *J. Electro. Test.: Theo. Appli. 3*, 377–385.

CHAKRAVARTY, S. AND THADIKARAN, P. 1996. Simulation and generation of IDDQ tests for bridging faults in combinational circuits. *IEEE Trans. Comput.*, 45, 10, 1131–1140.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. AND STEIN, C. 2003. *Introduction to Algorithms*. MIT Press.

HOROWITZ, E. SAHNI, S. AND MEHTA, D. 1995. *Fundamentals of Data Structures in C++*. W.H. Freeman and Company.

KOCAN, F. AND SAAB, D. G. 1999. Dynamic fault diagnosis on reconfigurable hardware. In *Proceedings of the Design Automation Conference*. 691–696.

LEE, H. K. AND HA, D. S. 1993. On the generation of test patterns for combinational circuits. Tech. Rep. 12–93, Department of Electrical Engineering, Virginia Polytechnic Institute and State University.

LEE, H. K. AND HA, D. S. 1992. HOPE: An efficient parallel fault simulator. In *Proceedings of the Design Automation Conference*. 336–340.

LI, R., OLSON, J. H. AND CHESTER, D. L. 1990. Dynamic fault detection and diagnosis using neural networks. In *Proceedings of the IEEE International Symposium on Intelligent Control*. 2, 1169–1174.

POMERANZ, I. AND REDDY, S. M. 1992. On the generation of small dictionaries for fault location. In *Proceedings of the International Conference on Computer-Aided Design*. 272–279.

RYAN, P. G., FUCHS, W. K. AND POMERANZ, I. 1993. Fault dictionary compression and equivalence class computation for sequential circuits. In *Proceedings of the International Conference on Computer-Aided Design*. 508–511.

THADIKARAN, P., CHAKRAVARTY, S. AND PATEL, J. 1997. Algorithms to compute bridging fault coverage of IDDQ test sets. *Trans. Des. Automat. Electro. Syst.* 2, 3, 281–305.

VENKATARAMAN, S., HARTANTO, I. AND FUCHS, W. K. 1996. Dynamic diagnosis of sequential circuits based on stuck-at faults. In *Proceedings of the IEEE VLSI Test Symposium*. 198–203.

Received June 2007; revised December 2007; accepted December 2008