INTEGRATED FAULT DIAGNOSIS TARGETING REDUCED SIMULATION

Vamsi Boppana
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801
vamsi@crhc.uiuc.edu

W. Kent Fuchs
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
kfuchs@ecn.purdue.edu

Abstract

***Integrated** fault diagnosis techniques attempt to overcome the limitations associated with **static** (pre-computed information usage) and **dynamic** (run-time analysis) techniques by using a limited amount of pre-computed information and coupling this with simulation at diagnosis time, for rapid fault diagnosis. A significant problem with previous integrated techniques is that the pre-computed information is not targeted specifically toward reducing the run-time costs. In this paper, we present a new approach to integrated fault diagnosis, by specifically creating the pre-computed information to provide later savings in the simulation costs at diagnosis time. Experimental results on the ISCAS 85 and ISCAS 89 circuits illustrate the savings achieved by this technique.*

1 Introduction

Fault diagnosis techniques can be broadly classified into three groups. The first group, called static fault diagnosis, uses pre-computed information in the form of fault dictionaries for matching with the faulty responses produced by defective circuits [1]. In contrast, dynamic techniques diagnose the faulty behavior of the circuit while the test set is applied [1]. Integrated diagnosis techniques focus on using small amounts of pre-computed information and coupling this with efficient dynamic algorithms to perform fault location [2–4]. Integrated techniques aim to overcome the size limitations of static fault diagnosis and the run-time limitations of dynamic techniques by providing flexibility in choosing the amount of pre-computed information, which, in turn, has an effect on the performance at diagnosis time. Integrated techniques typically store primary output information as part of the pre-computed information, although recent research has extended this to include internal node information [4].

This research was supported in part by the Semiconductor Research Corporation (SRC) under grant 95-DP-109, by the Office of Naval Research (ONR) under grant N00014-95-1-1049, and by an equipment grant from Hewlett-Packard.

A significant problem associated with previous integrated fault diagnosis solutions is that pre-computed information is decided without considering its full impact on the run-time simulation costs. In contrast, in this paper, we choose the pre-computed information specifically targeting a reduction in the fault simulation costs to be incurred at diagnosis time. In our strategy, we model the fault simulation costs in terms of computations associated with each (fault, vector) pair. We provide a technique that makes it possible to perform these computations independent of each other, thus providing an opportunity to evaluate the utility of a computation independent of other computations. We also identify computations that are not essential for the diagnosis process. This information is stored and used at diagnosis time to avoid unnecessary computations.

2 Computations in Diagnosis

Let the circuit under diagnosis possess t test vectors and f modeled faults.

1 (Set of Computations C_S) *The set of computations involved in the diagnostic process can be expressed as* $C_S = \{C(i,j),\ \forall\, i \in (1,\ldots,f),\ \forall j \in (1,\ldots,t)\}$, *where the computation* $C(i,j),\ \forall\ i\ \in\ (1,\ldots,f),\ \forall j\ \in (1,\ldots,t)$ *represents the simulation required to obtain the modeled fault response due to fault* i *at vector* j.

2.1 Independence of Computations

In order to be able to decide on the utility of each computation for the diagnosis process, it is necessary for the computations described to be independent of each other. This, however, is not true in general for sequential circuits. The simulation of a vector in a sequential circuit, without the simulation of all the vectors preceding it, is not a meaningful computation. This problem does not arise for combinational circuits, because there is no state information associated with these circuits. Hence, it is possible to obtain the response due to each modeled fault at any vector by the simulation of that vector alone. For **combinational circuits**, the computations $C(i,j),\ \forall\, i \in (1,\ldots,f),\ \forall j \in (1,\ldots,t)$ are independent of each other. For sequential circuits, a solution to the problem is provided by the following result.

2 (State Storage Point) *A point (i, j), $\forall\, i \in (1, \dots, f)$, $\forall j \in (1, \dots, t)$ is called a state storage point if the state of the faulty circuit i after application of the vectors $1, \dots, (j-1)$ is stored.*

Result 1 *A computation $C(i, j)$, $\forall\, i \in (1, \dots, f)$, $\forall j \in (1, \dots, t)$ on a* **sequential circuit** *is independent of any other computations if the point (i, j) is a state storage point.*

2.2 Unnecessary Computations

Unnecessary computations are best illustrated by the help of diagnostic experiment trees [1, 5–7]. The following definitions are from the work by Boppana and Fuchs [7].

3 (General Diagnostic Experiment Tree $T(V, E)$) *A general diagnostic experiment tree consists of a set of vertices $V(T)$ and a set of directed edges $E(T)$, i.e., each edge e is of the form (u, v), $u, v \in V(T)$ with the direction of the edge being* from u to v. *Each vertex $v \in V(T)$ of the tree is associated with a set of faults $F(v)$ that is a subset of the list of all modeled faults F, and each edge $e \in E(T)$ is associated with a list of outputs $O(e)$ that is a subset of all the primary outputs of the circuit.*

4 (Level of a node $L(v)$) *For each node $v \in V(T)$ of the tree, the level $L(v)$ is defined as the length of the path between $r(T)$ and v.*

5 (Vector-based Diagnostic Experiment Tree $T_V(V, E)$) *A diagnostic experiment tree in which each level represents the application of a test vector and each edge $e \in E(T_V)$ is associated with a list of outputs $O(e)$ that is the set of* all *the primary outputs of the circuit is called a vector-based diagnostic experiment tree.*

As an example, consider the tree of Figure 1 with ten faults, four test vectors and two primary outputs.

Sequential circuits are handled by allowing don't cares ($X's$) as special symbols of the alphabet along with the standard 0 and 1. Also, potential detection pointers are used to keep track of the potential distinguishability relations [6].

2.2.1 Unnecessary Computations of Type 1

The following identifies sets of computations that produce the same results. Such an identification results in the fact that it is sufficient to perform only one of each set of computations at diagnosis time.

Result 2 *If the faults from the set $F = (f_{i_1}, f_{i_2}, \dots f_{i_j})$ are all at the same node of a diagnostic tree $T_V(V, E)$, situated at level l, then the responses produced by the following sets of computations are identical.*

- $C(i_1, 1) = C(i_2, 1) = \dots = C(i_j, 1)$
- $C(i_1, 2) = C(i_2, 2) = \dots = C(i_j, 2)$

  $\vdots$
- $C(i_1, l) = C(i_2, l) = \dots = C(i_j, l)$

As an example, consider the faults $(1, 2, 3, 4)$ at level 3 of the diagnostic experiment tree shown in Figure 1. In this example, computations $C(1, 1), C(2, 1), C(3, 1)$ and $C(4, 1)$ all produce the same response 00.

2.2.2 Unnecessary Computations of Type 2

We now note yet another set of computations that fails to provide additional diagnostic capability. We first state the following result for combinational circuits.

Result 3 *If there are two nodes in the diagnostic tree such that the sets of faults associated with them are the same (say $F = (f_{i_1}, f_{i_2}, \dots f_{i_j})$), but they are at two distinct levels l and m, such that $l < m$, then it is easy to see that the following computations do not distinguish any more pairs of faults from F.*

- $C(i_1, l+1),\ C(i_2, l+1),\ \dots,\ C(i_j, l+1)$
- $C(i_1, l+2),\ C(i_2, l+2),\ \dots,\ C(i_j, l+2)$

  $\vdots$
- $C(i_1, m),\ C(i_2, m),\ \dots,\ C(i_j, m)$

As an example, consider the nodes $(5, 6)$ at level 1 and level 4. Then, our result implies that the computations $C(5, 2), C(5, 3)$ and $C(5, 4)$ do not distinguish any additional faults.

The corresponding result for sequential circuits is obtained by imposing the additional restriction that the fault class at the level l should be free of any potential distinguished pointers. It is to be noted that this restriction is far more stringent than the minimal restriction that just states that the set of faults should not even be potentially distinguished between levels l and m. But we choose to use this simple restriction, because it is easy to implement and our experimental results demonstrate that it provides significant savings.

The main motivation for identifying these computations comes from the fact that diagnostic experiment trees typically have a very large number of such unnecessary computations. By pre-computing information that enables the easy identification of such unnecessary computations, we need to perform only a small number of computations at run-time to achieve the required diagnosis result.

3 The Integrated Diagnosis Algorithm

3.1 Pre-computed Information

The primary focus here is on developing a compact storage structure that can be pre-computed and repeatedly used at diagnosis time. We now show that the storage of the class structure belonging to a few levels can be used to effectively control the simulation costs associated with diagnosis. The storage of class structure information marks a significant departure from the storage schemes used in previous static and integrated techniques.

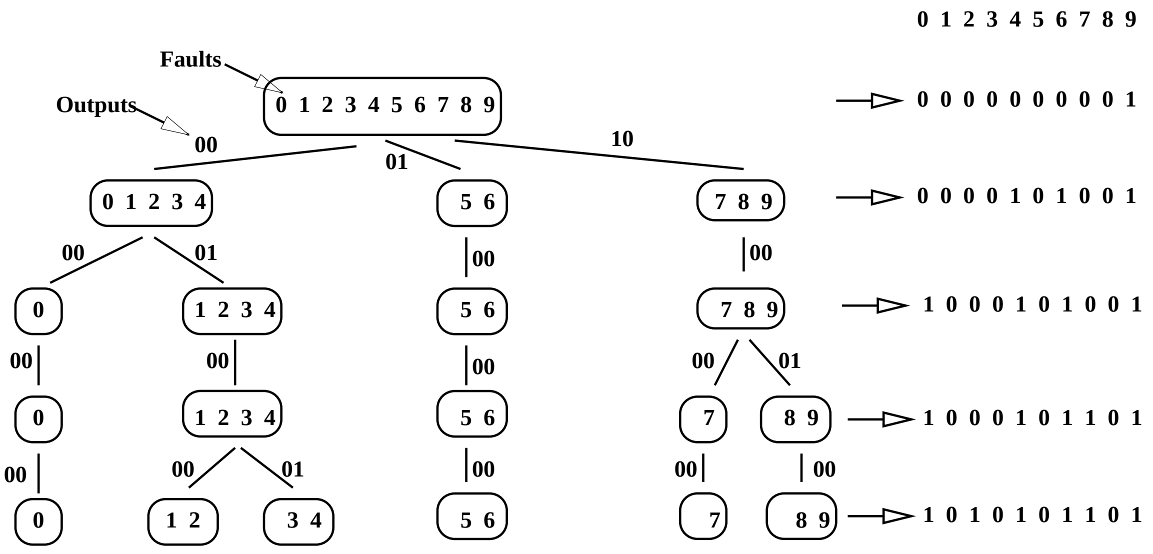

Figure 1: A Vector-based Diagnostic Experiment Tree and the Encoding of the Class Structure

We first claim that a string of f bits is sufficient to represent the class structure at any particular level. Let the faults be ordered such that they appear in the same order at the last level of the vector-based diagnostic experiment tree while traversing from left to right. We now present the encoding and decoding algorithms (see Figure 2) for determining the bit-code and class structure corresponding to a level of the tree.

```
Encoding Algorithm
// Let us start from the first fault of the first class
while (more faults to be handled)
  if (next fault in a different class or no next fault)
    place a 1 as the code entry at this fault location
  else
    place a 0 as the code entry at this fault location
end
Decoding Algorithm
// Let the pointer point to the first element of the code
while (pointer not reached the end of code)
  if (pointer points to a 0)
    current fault is still in the same class
  else // pointer points to a 1
    current fault is the last element of this class;
    start new class after this
end // while
```

Figure 2: The algorithms for encoding and decoding the class structure of a particular level

As an example of these algorithms, consider the tree of Figure 1, for which the codes corresponding to each level of the tree are listed alongside that level. In particular, for example, the code for level 2 is obtained to be 1000101001.

Our pre-computed storage structure is different for combinational and sequential circuits. For combinational circuits, it consists of storing the bit-code corresponding to the class structure at desired levels. For sequential circuits, at each level where the class structure is stored, we store another f-bit sequence that is used to indicate if there is any potential distinguished pointer associated with each class at that level (For Type 2 savings). Further, if there is a necessity to simulate a fault at a level, then the state corresponding to the fault is also stored. The amount of storage can be controlled by the careful selection of the levels at which the class structure is stored. Typically, it is beneficial to store the class structure more frequently during the first few levels of the diagnostic tree and less frequently thereafter.

3.2 The Diagnosis Algorithm

The integrated diagnosis algorithm presented in Figure 3 shows the two places in which the computations identified to be diagnostically unnecessary have been eliminated. This results in significant savings in the simulation costs. It is interesting to note that the algorithm degenerates into the conventional simulation-based dynamic diagnosis algorithm (with fault dropping) when there is no storage associated with any level.

4 Experimental Results

In this section, experimental results on ISCAS 85 and ISCAS 89 benchmark circuits are provided to demonstrate the effectiveness of our integrated diagnosis algorithm. For the purpose of experiments performed here, the number

```
// $F$ is the set of all faults in the circuit
// Initially, the class **C** is the same as **F**
// Routine **is_class_exactly_the_same(C)** checks
// the next stored level to see if there is any split in **C**;
// For sequential circuits, it also ensures that there are no
// potentially distinguished pointers associated with **C**
// Routine **compute$F$()** computes the set of
// consistent faults and drops inconsistent faults
// A class is consistent if its faults are consistent
repeat
  for each consistent class **C** at this stored level
    if (**is_class_exactly_the_same(C)**)
    // no need to simulate anything; Unnecessary (Type 2)
    else
      for each child class **K** of **C** at next stored level
        choose **one** fault from **K**
        load state if needed, simulate till next stored level
        Use the same simulation results for each fault in $K$
        // By simulating **one** element,
        // eliminated Unnecessary (Type 1)
  // At this stage all the necessary responses have been obtained
  **compute$F$()**
  move to next stored level
until no more vectors left
```

Figure 3: The integrated diagnosis algorithm

Table 1: Savings in Fault Simulation Costs

Circuit	No. Vecs	No. Faults	Cost Dyn.	Cost Int.	Int./ Dyn.%
c432	55	560	4769.5	134.8	2.8
c880	75	942	8050.7	238.3	3.0
c1355	88	1574	7545.5	153.4	2.0
c1908	280	1876	26807.5	159.9	0.6
c2670	236	2595	30677.8	438.6	1.4
c3540	350	3425	82897.8	492.7	0.6
c5315	264	5350	69203.9	1038.2	1.5
c6288	46	7744	12912.5	188.8	1.5
c7552	450	7550	48282.7	740.1	1.5
s298	259	308	7789.4	2361.2	30.3
s344	108	342	3858.2	800.9	20.8
s526	192	555	38554.6	1929.5	5.0
s641	211	467	6936.5	903.9	13.0
s820	968	850	88279.9	3494.1	4.0
s832	967	870	92682.3	3563.3	3.8
s1238	478	1355	76471.4	834.8	1.1
s1423	88	1515	38753.9	746.1	1.9
s1488	1192	1486	44512.0	3811.5	8.6
s1494	1285	1506	121448.7	4679.6	3.9
s5378	900	4603	244739.7	6651.9	2.7
s35932	383	39094	757886.0	6848.0	0.9

of storage levels has been set to the number of vectors. The integrated algorithm is compared to the conventional, simulation-based dynamic diagnosis algorithm using fault dropping. Modeled faults (stuck-ats) were chosen at random and the output responses produced by the circuit under the influence of these faults were presented as the defective unit responses to both of the diagnosis algorithms. We present the simulation costs for both of the algorithms in terms of the number of computations required. These results are shown in Table 1. The table presents the average number of simulations required for the two algorithms. Specifically, the columns in the table represent, for each benchmark circuit, the number of test vectors, the number of faults, the average number of units of computation required for the dynamic diagnosis algorithm per fault, the average number of units of computation required for the integrated diagnosis algorithm per fault, and a percentage ratio of the computations performed by the integrated algorithm with respect to the dynamic algorithm. Hence, we see that a substantially large number of computations have been eliminated from the diagnosis process.

5 Conclusions

We have presented a new approach to integrated fault diagnosis by identifying the pre-computed information so as to reduce the fault simulation costs associated with diagnosis time. The diagnosis algorithm uses the pre-computed information to avoid unnecessary computations during diagnosis. Our experiments on benchmark circuits have shown that significant savings can be achieved in the fault simulation costs by the use of this technique.

References

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

[2] P. Ryan, S. Rawat, and W. K. Fuchs, "Two-stage fault location," in *Proceedings of the International Test Conference*, October 1991, pp. 963–968.

[3] P. G. Ryan, "Compressed and dynamic fault dictionaries for fault isolation," in *CRHC Technical Report UILU-ENG-94-2234*, September 1994.

[4] V. Boppana, I. Hartanto, and W. K. Fuchs, "Fault diagnosis using state information," in *Proceedings of Fault Tolerant Computing Symposium*, June 1996, pp. 96–103.

[5] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill Book Company, New York, 1978.

[6] S. Venkataraman, I. Hartanto, W. K. Fuchs, E. M. Rudnick, S. Chakravarty, and J. H. Patel, "Rapid diagnostic fault simulation of stuck-at faults in sequential circuits using compact lists," in *Proceedings of the Design Automation Conference*, June 1995, pp. 133–138.

[7] V. Boppana, I. Hartanto, and W. K. Fuchs, "Full fault dictionary storage based on labeled tree encoding," in *Proceedings of VLSI Test Symposium*, April 1996, pp. 174–179.