# A Diagnostic Test Generation System

Yu Zhang* and Vishwani D. Agrawal

Auburn University, Department of Electrical and Computer Engineering, Auburn, AL 36849, USA

*yzz0009@auburn.edu, vagrawal@eng.auburn.edu*

***Abstract*** *— A diagnostic automatic test pattern generation (DATPG) system is constructed by adding new algorithmic capabilities to conventional ATPG and fault simulation programs. The DATPG aim to generate tests to distinguish fault pairs, i.e., two faults must have different output responses. Given a fault pair, by modifying circuit netlist a new single fault is modeled. Then we use a conventional ATPG to target that fault. If a test is generated it distinguishes the given fault pair. We use a fast diagnostic fault simulation algorithm to find undistinguished fault pairs from a fault list for a given test vector set. In this fault simulator faults are partitioned into different groups according to their output responses obtained by a conventional fault simulator. Thus, fault pairs that a simulated vector can distinguish between are split among separate groups. Faults that form a single fault groups are dropped from further simulation with subsequent vectors. We use a proposed diagnostic coverage (DC) metric, defined as the ratio of the number of fault groups to the number of total faults. The diagnostic ATPG system starts by first generating conventional fault coverage vectors. Those vectors are then simulated to determine the DC, followed by repeated applications of diagnostic test generation and simulation. We observe impoved DC in all benchmark circuits. Cases of low DC have helped identify new open problems.*

## 1 Introduction

A common objective of testing is to detect all or most modeled faults. Although fault coverage (percentage or fraction) has a somewhat nonlinear relationship with the tested product quality or defect level (parts per million) for practical reasons fault coverage continues to be a measure of the test quality [4].

Most test generation systems are built around *core ATPG algorithms* [4] for (1) finding a test vector for a target fault and (2) simulating faults to find how many have been detected by a given vector. The system then attempts to find tests of *high* fault coverage because the primary objective is fault detection, i.e., presence or absence of faults. In some test scenarios we must *diagnose* or identify the fault, making the test intent different from the original detection coverage. In practice, detection tests with high coverage may not have adequate diagnostic capability. One often uses tests for multiple fault models [25, 28] or multiple tests for the same model [3]. Basically, we generate tests that are redundant for fault detection and then hope that they will provide better diagnostic capability. To reduce the excess tests we may resort to optimization or removal of unnecessary tests [11, 24].

A contribution of this paper is in providing basic core algorithms for a diagnostic test generation system. Given a pair of faults, we should either find a distinguishing test or prove that the faults are equivalent in a diagnostic sense; *diagnostic equivalence* implies that faulty functions at all outputs of a multioutput circuit are identical [23]. The new exclusive test algorithm of Sections 3 and 4 represents a non-trivial improvement in computational efficiency over previously published work [1]. Next, we provide a diagnostic coverage metric in Section 5.1 similar to the detection fault coverage, such that a 100% diagnostic coverage means that each modeled fault is distinguished from all other faults. Diagnostic fault simulation of Section 5.2 is another core algorithm presented. A fault simulator is an essential tool for obtaining meaningful tests. Years of research has produced highly efficient fault simulation algorithms and programs [4]. The key features of the diagnotic algorithm presented here are (1) it accepts fault detection data from any conventional fault simulator thus benefitting from the efficiency of a matured program, (2) fault dropping is used to delete diagnosed faults from the list of faults as fault simulation progresses, and (3) for a given set of input vectors it provides fault coverage (*FC*), diagnostic coverage (*DC*), and necessary data for fault dictionary. In Section 6 a diagnostic test generation system combines the core algorithms and the new coverage metric into an ATPG system.

## 2 Background

*Exclusive test* for a pair of faults has been defined as a test that detects one fault but not the other [1]. Its primary intent is to distinguish between the fault pair. For a multiple output circuit, this definition is applied separately to each output. An exclusive test can detect both faults as long as they are not being detected at the same outputs. Perhaps a more appropriate term would be

* Student presenter.

distinguishing test. However, following existing usage of the term, we will continue with *exclusive test*.

This background is reproduced from previous publications[1, 9]. Consider a pair of faults, $f_1$ and $f_2$. Figure 1 illustrates generation of an exclusive test. A fault-free digital circuit under test (CUT) is shown as $C_0$. Blocks $C_1$ and $C_2$ are the same circuit with faults $f_1$ and $f_2$, respectively. The circuit is assumed to be combinational and can have any number of inputs. For clarity, we will only consider single output functions. Any input vector that produces a 1 output in Figure 1, i.e., detects a s-a-0 fault at the primary output is an exclusive test for the fault pair ($f_1$, $f_2$). This is the Boolean satisfiability version of the exclusive test problem [31],

$$(C_0 \oplus C_1) \oplus (C_0 \oplus C_2) = 1 \quad (1)$$

Which simplifies to,

$$C_1 \oplus C_2 = 1 \quad (2)$$

The second simplified form, shown in Figure 2, implies that the two faulty circuit outputs differ for the exclusive test. There are two ways to generate exclusive tests. We can either modify a single-fault ATPG program [9] or modify the circuit and use an unmodified ATPG program [1, 26]. This way we can take advantage of newly developed ATPG programs while for a specially designed DATPG [9] the entire program has to be modified to implement any new algorithm.

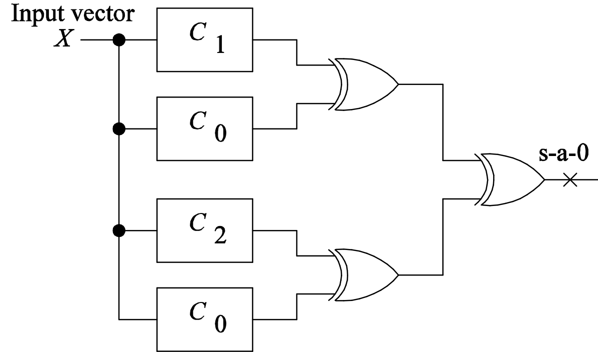


**Figure 1. The general exclusive test problem.**

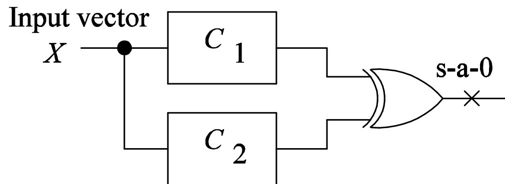


**Figure 2. The simplified exclusive test.**

## 3 Boolean Analysis of Exclusive Test

To further simplify the solution shown in Figure 2 and equation (2), we first transform it into an equivalent single-fault ATPG problem shown in Figure 3. Here we have introduced a new primary input variable *y*. The function *G* in Figure 3 can be expressed as Shannon's expansion [4] about *y* with cofactors $C_1$ and $C_2$ [31]:

$$G(X, y) = \bar{y}C_1 + yC_2 \quad (3)$$

The condition for detecting s-a-0 or s-a-1 fault on *y*, using Boolean difference [4] , is

$$\frac{\partial G}{\partial y} = G(X, 0) \oplus G(X, 1) = C_1 \oplus C_2 \quad (4)$$

This is identical to the second exclusive test condition of equation 1. Thus, we establish that a vector *X* that detects either s-a-0 or s-a-1 fault on *y* in the circuit *G(X, y)* of Figure 3 will also detect the output s-a-0 fault in the circuit of Figure 2. We call *G* as *ATPG circuit*. It maps the given complex ATPG problem as a single fault ATPG problem. Next, we synthesize a compact circuit for *G(X, y)*.

Suppose fault $f_1$ is line $x_1$ s-a-a and fault $f_2$ is line $x_2$ s-a-b, where $x_1$ and $x_2$ are two signal lines in the circuit *C*. Fault variables *a* and *b* can assume any value 0 or 1. The primary input of *C* is a vector *X* that may or may not contain the two fault lines. We express the fault-free function using Shannon's expansion [4] as,

$$\begin{aligned} C(X, x_1, x_2) &= \bar{x}_1\bar{x}_2C(X, 0,0) + \bar{x}_1x_2C(X, 0,1) \\ &+ x_1\bar{x}_2C(X, 1,0) + x_1x_2C(X, 1,1) \end{aligned} \quad (5)$$

Therefore, the cofactors of *G(X, y)* are given by,

$$\begin{aligned} C_1 = C(X, a, x_2) &= \bar{a}\bar{x}_2C(X, 0,0) + \bar{a}x_2C(X, 0,1) \\ &+ a\bar{x}_2C(X, 1,0) + ax_2C(X, 1,1) \\ C_2 = C(X, x_1, b) &= \bar{x}_1\bar{b}C(X, 0,0) + \bar{x}_1bC(X, 0,1) \\ &+ x_1\bar{b}C(X, 1,0) + x_1bC(X, 1,1) \end{aligned} \quad (6)$$

Let us define following variables:

$$x_1' = \bar{y}a + yx_1 \quad \text{and} \quad x_2' = \bar{y}x_2 + yb \quad (7)$$

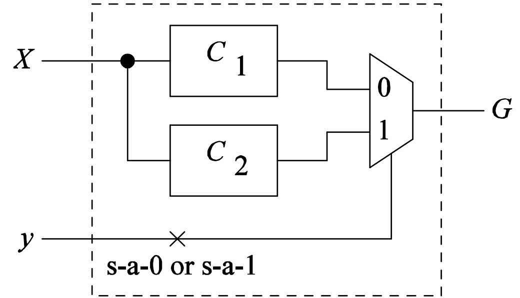


**Figure 3. An ATPG circuit for exclusive test.**

Using the rules of Boolean algebra, such as absorption and consensus theorems [18], we obtain [31]

$$\bar{x}_1'\bar{x}_2' = \bar{y}\bar{a}\bar{x}_2 + y\bar{x}_1\bar{b} \quad (8)$$

$$\bar{x}_1'x_2' = \bar{y}\bar{a}x_2 + y\bar{x}_1 b \quad (9)$$

$$x_1'\bar{x}_2' = \bar{y}a\bar{x}_2 + yx_1\bar{b} \quad (10)$$

$$x_1'x_2' = \bar{y}ax_2 + yx_1 b \quad (11)$$

First we substitute $C_1$ and $C_2$ from equations (5) into equation 2 and then make use of equations (7) through (10), to obtain [31]:

$$\begin{aligned} G(X,y) &= \bar{x}_1'\bar{x}_2'C(X,0,0) + \bar{x}_1'x_2'C(X,0,1) \\ &+ x_1'\bar{x}_2'C(X,1,0) + x_1'x_2'C(X,1,1) \\ &= C(X,x_1',x_2') \quad (12) \end{aligned}$$

where the last result follows from the Shannon's expansion of the original circuit function $C$, given by equation (5), in which new variables $x_1'$ and $x_2'$ defined in equations (7) replace $x_1$ and $x_2$, respectively.

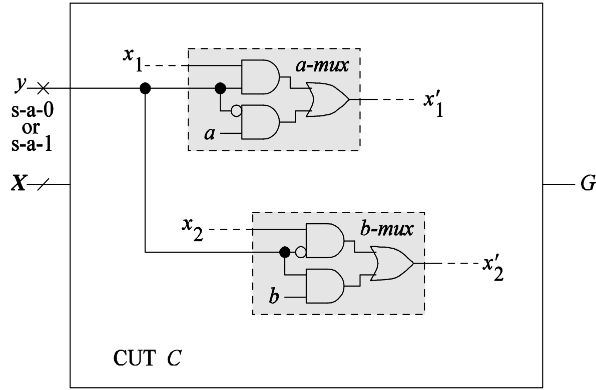


**Figure 4. ATPG circuit with multiplexers inserted in CUT such that a test for s-a-0 or s-a-1 fault on *y* is an exclusive test for faults $x_1$ s-a-*a* and $x_2$ s-a-*b*.**

The synthesized circuit for *G(X, y)*, shown in Figure 4, is obtained by inserting two multiplexers, a–mux and b–mux controlled by a new primary input *y*,in the original circuit *C(X)*. Veneris et al. [26] arrive at a similar construction though our derivation provides greater insight into the procedure. For any 0 or 1 value of variables *a* and *b* each multiplexers simplifies either to a single gate or a gate with an inverter. Consider the example circuit of Figure 5 from a previous paper [1]. We seek an exclusive test for two faults shown. The ATPG circuit for this problem is given in Figure 6. The logic shown with shading is obtained by simplifying the multiplexers of Figure 4 upon setting $a = 1$ and $b = 0$. The exclusive test found by a single fault ATPG is $a = 0$, $b = 1$, $c = 1$, $d = 0$. The signal values shown on lines are from five-valued D-algebra [4].

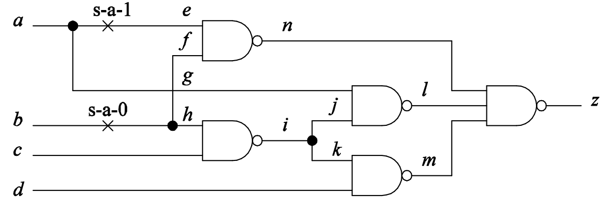


**Figure 5. A circuit with exclusive test required for *e* s-a-1 and *b* s-a-0** [1].

## 4 Generation of Exclusive Test

An exclusive test for two faults in a single output circuit must detect only one of those faults. In a multiple output circuit, that same condition must be satisfied at least on one output. Suppose, $C_1$ and $C_2$ blocks in Figure 2 each has n outputs. We will then have n two-input XOR gates such that the *ith* XOR gate receives *ith* outputs from $C_1$ and $C_2$. All XORs feed into an n-input OR gate whose output contains the single s-a-0 faults to be detected.

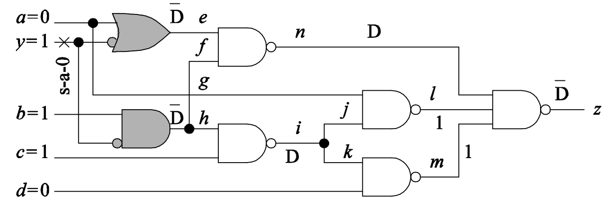


**Figure 6. ATPG circuit for the exclusive test problem of Figure 5.**

In general, an exclusive test for a multiple output circuit can detect both targeted faults as long as they are not being detected on exactly the same outputs. The ATPG circuit derived in the previous section remains valid for multiple outputs. Figure 7 shows the construction for two outputs.

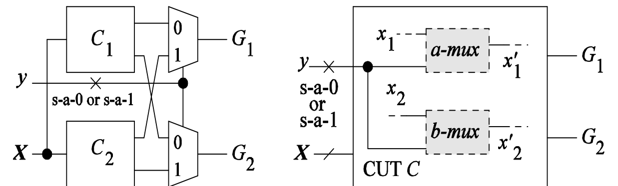


**Figure 7. Exclusive test ATPG circuit for a two-output CUT.**

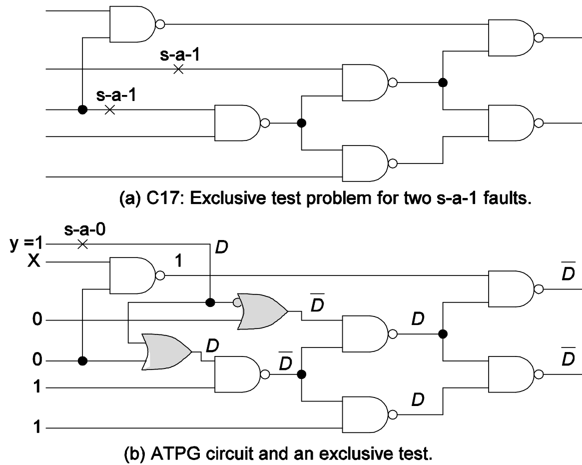


**Figure 8. Exclusive test example for multi-output circuit c17.**

Consider the problem of finding an exclusive test for two s-a-1 faults in the c17 circuit shown in Figure 8(a). The fault on signal *y* in the ATPG circuit of Figure 8(b) provides a test X0011. We can verify that this test detects both faults but at different outputs, hence it will distinguish between them.

## 5 Diagnostic Fault Simulator

Several diagnostic fault simulation methods have been presented [27, 29]. In one of the proposed method, faults are grouped into classes. Faults with identical output responses are put in the same class. Fault pairs consisting of faults from the same class are sent to an equivalence identification tool. If the fault pair is proved equivalent, one fault is dropped from the fault list. By concurrently performing diagnostic fault simulation and equivalence identification, the simulation time is greatly reduced. In our method a fault is dropped when it is distinguished from all other faults; this is achieved without fault equivalence checking, though we can also benefit from it. In [27], a diagnostic fault simulator is constructed for sequential circuits. Because of the possibility of the X state at a primary output in a sequential circuit, additional information has to be stored for diagnosis. For example, consider a sequential circuit with three primary outputs. For a certain input vector, suppose fault 1 has a response 1X0 and fault 2, X1X. These two faults are said to be *potentially distinguished*. However this is not the case for combinational or full scan circuits for which the simulation method described in this paper can be more memory and time efficient. In [9], a diagnostic test pattern generator (DATPG) for combinational circuits is presented. It aims at generating distinguishing test vectors for given fault pairs and no diagnostic fault simulation process is used. Such DATPG can greatly benefit from a fast simulation scheme.

### 5.1 Diagnostic Coverage Metric

To generate diagnostic tests we need a coverage criterion. This would be similar to the fault coverage ($FC$) used in conventional ATPG systems where fault detection is the objective. Since the core algorithm of previous sections generates distinguishing tests for fault pairs, our first inclination was to specify coverage as the fraction of distinguishable fault pairs. This has been called diagnostic resolution in several papers[6, 7, 27]. Consider an example. For $N$ faults, there will be $N(N-1)/2$ fault pairs. For a moderate size circuit, suppose $N$ = 10,000 then there are 49,995,000 fault pairs. If our tests do not distinguish 5,000 fault pairs, then the coverage would be (49,995,000 − 5,000)/49, 995,000 = 0.999, which turns out to be highly optimistic. There is additional problem of high complexity; for $N = 10^6$ the number of fault pairs is approximately half a billion. We, therefore, propose an alternative metric. For a set of vectors we group faults such that all faults within a group are not distinguishable from each other by those vectors, while each fault in a group is pair-wise distinguishable from all faults in every other group. This grouping is similar to equivalence collapsing except here grouping is conditional to the vectors. If we generate a new vector that detects a subset of faults in a group then that group is partitioned into two groups, one containing the detected subset and the other containing the rest. Suppose, we have sufficient vectors to distinguish between every fault pair, then there will be as many groups as faults and every group will have just one fault. Prior to test generation all faults are in a single group we will call $g_0$. As tests are generated, detected faults leave $g_0$ and start forming new groups, $g_1, g_2, . . . g_n$, where $n$ is the number of distinguishable fault groups. For perfect detection tests $g_0$ will be a null set and for perfect diagnostic tests, $n = N$, where $N$ is the total number of faults. We define *diagnostic coverage*, *DC*, as

$$DC = \frac{Number\ of\ detected\ fault\ groups}{Total\ number\ of\ faults} = \frac{n}{N} \quad (13)$$

Initially, without any tests, $DC = 0$, and when all faults are detected and pair-wise distinguished, $DC = 1$. Also, the numerator in equation (13) is the number of fault dictionary syndromes [4] and the reciprocal of $DC$ is the *diagnostic resolution* ($DR$) defined as the average size of fault group that cannot be further diagnosed [1]. Other metrics, diagnostic expectation [22, 30] that gives unnormalized and often large values and diagnostic power [6] that gives normalized but pessimistic values, need to be compared with $DC$ defined here. For

completeness of this discussion, detection fault coverage $(FC)$ is,

$$FC = \frac{Number\ of\ detected\ faults}{Total\ number\ of\ faults} = \frac{N - |g_0|}{N} \quad (14)$$

## 5.2 Diagnostic Fault Simulation Algorithm

We explain the simulation algorithm using a hypothetical example given in Figure 10. Suppose a circuit has eight faults $(N = 8)$, identified as $a$ through $h$. Assume that the circuit has two outputs. The grey shading, which identifies the undetected fault group $g_0$, indicates that all faults are undetected in the initial fault list. Also, fault coverage $(FC)$ and diagnostic coverage $(DC)$ are both initially 0. We assume that there are three vectors generated in detection phase and can detect all faults (Blocks 1 and 2 in Figure 9), thus having 100% $FC$. Later in the simulation process more vectors will be generated by diagnostic ATPG to improve diagnostic coverage $(DC)$.

The diagnostic phase begins with the three vectors supplied to Block 3. The first vector is simulated for all eight faults using a conventional fault simulator. We use a *full-response* simulator that gives us the fault detection information for each primary output $(PO)$. Suppose we find that the first vector detects $a$, $e$ and $g$. Also, faults $a$ and $e$ are detected only on the first output and $g$ is detected on both outputs. Thus, fault pairs $(a, g)$ and $(e, g)$ are distinguishable, while the pair $(a, e)$ is not distinguishable. The result is shown in the second list in Figure 10. The fault list is partitioned into three groups. The first two groups, $g_1$ and $g_2$, shown without shading contain detected faults. Group $g_0$ now has 5 faults. Each group contains faults that are not distinguished from others within that group, but are distinguished from those in other groups. Counting detected faults, the fault coverage is 3/8 and counting detected fault groups, the diagnostic coverage is 2/8 [31].

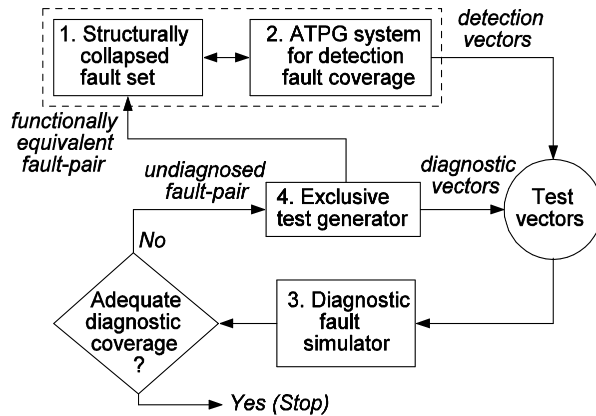


**Figure 9. Diagnostic test generation system.**

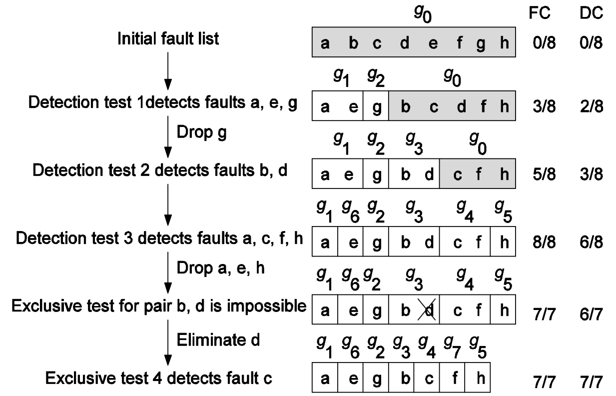


**Figure 10. Illustration of diagnostic fault simulation.**

Fault $g$, which is in a single fault group, is dropped from further simulation. This is similar to diagnostic fault dropping reported in other papers [27, 29]. Because this fault has been uniquely distinguished from all other faults, its distinguishability status will not change by other vectors. Note that pair-wise distinguishability provided by future vectors can only subdivide the groups and subdivision of a group with just one fault will be impossible. *The fact that faults can be dropped in diagnostic fault simulation is not always recognized.* However, fault dropping is possible here only because our interest is in diagnostic coverage and not in minimizing the vector set. Seven faults are now simulated for the second vector, which detects faults $b$ and $d$. Suppose, $b$ and $d$ are detected at the same set of outputs and hence are placed within same partition $g_3$. Thus, $FC = 5/8$ and $DC = 3/8$. No new fault can be dropped at this stage.

Vector 3 detects faults $a$, $c$, $f$ and $h$ increasing the fault coverage to 100%. Suppose $c$ and $f$ are detected at the same set of outputs and so are placed together in group $g_4$. Detection at different outputs distinguishes $h$ from these two and hence $h$ is placed in a separate group $g_5$. Also, noting that this test distinguishes between $a$ and $e$, group $g_1$ is split into $g_1$ and $g_6$. Now, $FC = 8/8 = 1.0$ and $DC = 6/8$. Faults in fault groups with single fault are dropped.

Having exhausted the detection vectors, we find that two pairs, $(b, d)$ and $(c, f)$, are not distinguished. We supply target fault pair $(b, d)$ to Block 4 in the ATPG system of Figure 9. Suppose we find that an exclusive test, i.e., a test that detects any one fault but not the other, is impossible thus indicating that two faults are equivalent. We remove one of these faults, say $d$, from $g_3$ and from the fault list as well. This does not change fault coverage since $FC$ = 7/7, but improves the diagnostic coverage to $DC$ = 6/7. All faults except $c$ and $f$ are now dropped from further simulation.

The only remaining fault pair $(c, f)$ is targeted and an exclusive test is found. Suppose fault $f$ is detected by this vector but $c$ is not detected Thus, $g_4$ is partitioned to create group $g_7$ with fault *f*. The new partitioning has just one fault per group, $FC$ = 7/7, and $DC$ = 7/7.

With fault dropping the simulation time for each vector is gradually reduced. The CPU time should have similar curve to that in Figure 13 (but fliped up side down).

## 5.3 Dictionary Construction

Fault dictionary is necessary in a cause-effect diagnosis. It facilitates faster diagnosis by comparing the observed behaviors with pre-computed signatures in the dictionary [24]. One common form of dictionary is the *full-response* (FR) dictionary, which stores all output responses of each faults for each test. But the problem is the size of a FR dictionary can grow prohibitively large, i.e., $(F \times V \times O)$ where $F$ is the number of faults, $V$ is number of vectors, and $O$ is number of primary outputs.

Much work has been done to reduce the size of the FR dictionary [5, 19, 20]. Here we assign integers to different output responses. Thus the largest integer needed to index all different syndromes in the worst case will be $minimum(2^n - 1,\ F \times V)$ where $n$ is number of primary outputs, $F$ is number of faults, and $V$ is number of vectors. However, it should be noted that faults in a same logic cone tend to produce identical output respones for a given vector set, so that the largest index is usually much smaller than $F \times V$.

| Faults | Output responses | | | |
|---|---|---|---|---|
| | t1 | t2 | t3 | t4 |
| a | 10 | 00 | 10 | X |
| b, d | 00 | 01 | 00 | X |
| c | 00 | 00 | 01 | 00 |
| e | 10 | 00 | 00 | X |
| f | 00 | 00 | 01 | 11 |
| g | 11 | X | X | X |
| h | 00 | 00 | 10 | X |

**Figure 11. FR Dictionary [31].**

The dictionary shown in Figure 11 is generated based on the example in Figure 10. Among the entries, X means the fault is already dropped and not simulated, 0 stands for pass (same as fault-free response), and 1 stands for fail. To reduce the dictionary size we assign integers to index different output responses. In this example, "10", "11", and "01" are indexed with 1, 2, 3, as shown in Figure 12. Although for small circuits the compression is not obvious, for larger ISCAS85 benchmark circuits the reduction can be as high as an order of magnitude.

| | t1 | t2 | t3 | t4 |
|---|---|---|---|---|
| a | 1 | 0 | 1 | X |
| b, d | 0 | 3 | 0 | X |
| c | 0 | 0 | 3 | 0 |
| e | 1 | 0 | 0 | X |
| f | 0 | 0 | 3 | 2 |
| g | 2 | X | X | X |
| h | 0 | 0 | 1 | X |

**Figure 12. Compressed dictionary [31].**

Because of fault dropping in our simulator there will be 'X' in the generated dictionary. This limits the use of fault dictionary to single stuck-at fault. For a real defect the faulty respones may have no match in the dictionary. To solve this problem we introduce a heuristic.

$$\frac{Hamming\ Distance}{O \times (V - X)} \leq threshold\ value \qquad (15)$$

Here hamming distance is calculated from observed response to the stored syndromes, ignoring 'X's. $O$ is the number of primary outputs, $V$ is number of vectors, and $X$ is number of 'X's for a fault in the dictionary. If the calculated result is smaller than a given threshold, the corresponding fault will be added to a candidate list. Then fault simulation without fault dropping will be performed on this list to obtain additional information to further narrow down upon a fault candidate.

# 6 Diagnostic ATPG System

Figure 9 shows the flowchart of a diagnostic ATPG system implemented in the Python programming language [21]. Main functions are Blocks 1 through 4. Blocks 1 and 2 form a conventional detection coverage ATPG system. In our system, these functions are provided by Atalanta [14] and Hope [15] acquired from Virginia Tech. Block 4 is an exclusive test generator program that implements the core algorithm of Sections 3 and 4. Internally, it also uses Atalanta for detecting a fault on line *y* in the ATPG circuit constructed for the given fault pair (Figure 4 and 7). Block 3 is a diagnostic fault simulator described before.

## 6.1 Redundant and Aborted Faults

In the ATPG system of Figure 9 when a single fault ATPG program is run it can produce three possible outcomes, (1) test found, (2) no test possible, or (3) run aborted due to CPU time or search limit in the program. In (1) a new detection or exclusive test is found. In (2), if it is detection test then the fault is redundant and is

removed from the fault list. If it is an exclusive test then the two faults are equivalent (perhaps functionally) and any one is removed from the fault list. In (3), for detection phase the fault remains in the set $g_0$ and results in less than 100% *FC* and *DC*. For an aborted exclusive test the target fault pair remains indistiguishable causing reduced *DC* and worse diagnostic resolution (*DR*) [1].

## 7 Results

We used the ATPG system of Figure 9 and Section 6. Internally, it employs the ATPG program Atalanta [14] and fault simulator Hope [15]. The circuit modeling for exclusive test and fault grouping for diagnostic fault simulation were implemented in the Python language [21]. The system runs on a PC based on Intel Core-2 duo 2.66GHz processor with 3GB memory. The results for c432 were as follows:

- Fault detection phase:
  - Number of structurally collapsed faults: 524
  - Number of detection vectors generated: 51
  - Faults: detected 520, aborted 3, redundant 1
  - Fault coverage, FC: 99.43%
- Diagnostic phase:
  - Initial DC of 51 vectors: 91.985%
  - Number of exclusive tests generated: 18
  - Number of undistinguished groups: 0
  - Largest size of undistinguished group: 1
  - Final diagnostic coverage DC: 100.%

Fault coverage (*FC*) and diagnostic coverage (*DC*) as functions of number of vectors are shown in Figure 13. First 51 vectors were generated in the fault detection phase, which identified only one of the known four redundant faults in this circuit. Diagnostic fault simulation computed the diagnostic coverage of 51 vectors as 91.985%. The diagnostic phase produced 18 exclusive tests while identifying 13 equivalent fault pairs. Diagnostic coverage of combined 69 vectors is 100%. No group has more than 1 fault. We also simulated a set of 69 random vectors and their coverages are shown in Figure 14. As expected, both fault coverage and diagnostic coverage are lower than those for algorithmic vectors. Results for ISCAS'85 circuits, shown in Table 1.

Table 1 indicates a dropping *DC* as circuit size increases. Once again, this is expected because of larger numbers of aborted pairs. Notice 453 aborted pairs for c1355. This circuit is functionally equivalent to c499, which has a large number of XOR gates. In c1355, each XOR gate is expanded as four NAND gates. This implementation of XOR function is known to have several functionally equivalent faults [2]. Its structurally collapsed set of 1,574 faults reduces to 950 faults when functional collapsing is used. If we use the set of 950 faults, the same 85 + 2 = 87 vectors of Table 1 will show a significantly higher *DC*. Thus, the advantage of functional fault collapsing, though only marginal in detection ATPG, can be significant in diagnostic test generation.

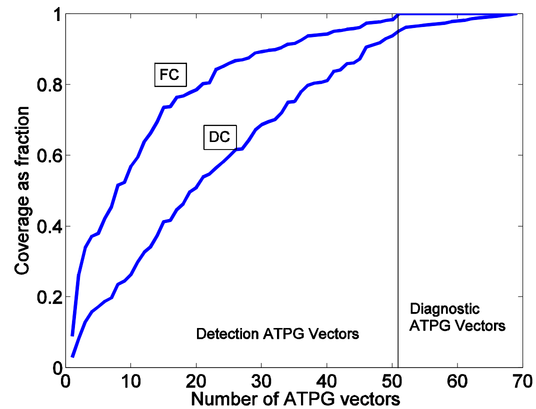


**Figure 13. Diagnostic fault simulation of c432 for 69 algorithmic vectors. *FC*: fault coverage, *DC*: diagnostic coverage [31].**

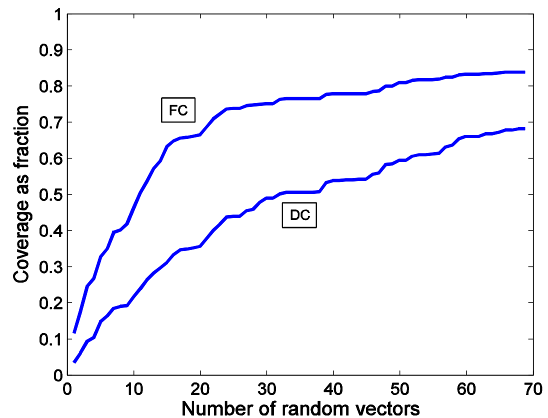


**Figure 14. Diagnostic fault simulation of c432 for 69 random vectors. *FC*: fault coverage, *DC*: diagnostic coverage [31].**

Similarly, the size of the undiagnosed fault group tends to increase for larger circuits. It is 11 for c2670. This is related to the lower *DC*, whose reciprocal is the diagnostic resolution (*DR*) [1]. *DR* > 1 indicates poor diagnosis; the ideal resolution *DR* = 1 [1] requires that each undistinguished fault group is no larger than 1.

The above discussion points to the need for improved fault collapsing and efficient redundancy identification algorithms. Much work on these has been reported, which we are exploring for suitable approaches. Interestingly, we find that most redundancies or

functional fault equivalences are localized within relatively small sub-circuits irrespective of how large the entire circuit is. Even though both problems have exponential complexities, this observation provides hope for finding efficient algorithms.

A comparison of two CPU time columns (columns 5 and 12) of Table 1 shows that diagnostic ATPG takes significantly more. This is because our implementation uses an existing ATPG program without changes. For detection ATPG circuit data structure is built once and then used repeatedly by every vector generation run. Even though the data structure building time is higher (can be ten times) than that of a single vector generation, the total CPU time is dominated by the combined vector generation times. However, in diagnostic ATPG we repeatedly modify the netlist, the ATPG program must rebuild the data structure prior to each vector generation run. An improved program will incrementally update the data structure instead of rebuilding it. That will bring the diagnostic ATPG time (column 13) in line with the detection ATPG time (column 5). While for diagnostic fault simulation, the CPU time is linearly dependent on each of the three variables, namely, number of gates, number of faults and number of vectors. The CPU times in Table 1 include the time of the conventional fault simulator Hope [15] and that of our Python program that partitions the fault list and computes *DC*. The overall increase in run times with increasing circuit size for diagnostic simulation shown in Table 1 is between $O(gates^2)$ and $O(gates^3)$, which is no different from what has been reported for conventional fault simulation [3, 6].

## 8 Conclusion

The core algorithms for exclusive test generation and diagnostic fault simulation should find effective use in the test generation systems of the future. Key features of these algorithms are: (1) exclusive test generation is no more complex than test generation for a single fault, and (2) diagnostic fault simulation has similar complexity as conventional simulation with fault dropping. The new definition of diagnostic coverage (*DC*) is no more complex than the conventional fault detection coverage and it directly relates to the diagnostic resolution [1], which is reciprocal of *DC*. These methods can be applied to other fault models [10, 16, 17] and to mixed fault models [28]. Future extensions of this work would be on generating compact diagnostic tests and organizing fault dictionaries of test syndromes. Because our fault simulation is done with fault dropping, the syndromes will contain 0, 1, and X (don't care). However, these don't cares do not reduce the diagnosability of a fault. Although, reordering or compaction of vectors will be affected. This needs further investigation. Our results show low *DC* on some benchmark circuits. For example in c1355 there are 453 aborted fault pairs, which are actually all equivalent pairs. This indicates that to obtain high *DC* for large circuits, improved heuristic algorithms for functional equivalence checking and redundancy identification will have to be implemented. One possible way is to exploit the fact that the distance between equivalent faults is generally small [12]. Thus an equivalence checking algorithm can be developed based on extracting a subcircuit that contains the fault pair.

***Acknowledgment*** – This research is supported in part by the National Science Foundation Grant CNS-0708962.

## References

[1] V. D. Agrawal, D. H. Baik, Y. C. Kim, and K. K. Saluja, "Exclusive Test and its Applications to Fault Diagnosis," in *Proc. 16th International Conf. VLSI Design*, Jan. 2003, pp. 143–148.

[2] V. D. Agrawal, A. V. S. S. Prasad, and M. V. Atre, "Fault Collapsing via Functional Dominance," in *Proc. International Test Conf.*, 2003, pp. 274–280.

[3] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, N. Tamarpalli, K.-H. Tsai, and J. Rajski, "Impact of Multiple-Detect Test Patterns on Product Quality," in *proc. International Test Conf.*, 2003, pp. 1031–1040.

[4] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Boston: Springer, 2000.

[5] B. Chess and T. Larrabee, "Creating Small Fault Dictionaries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 346-356, Mar. 1980.

[6] P. Camurati, A. Lioy, P. Prinetto, and M. S. Reorda, "A Diagnostic Test Pattern Generation Algorithm," in *Proc. International Test Conf.*, 1990, pp. 52–58.

[7] S.-C. Chen and J. M. Jou, "Diagnostic Fault Simulation for Synchronous Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 16, no. 3, pp. 299–308, Mar. 1997.

[8] P. Goel, "Test Generation Costs Analysis and Projections," in *Proc. 17th Design Automation Conf.*, 1980, pp. 77–84.

[9] T. Grüning, U. Mahlstedt, and H. Koopmeiners, "DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits," in *Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 194-197, Nov. 1991.

[10] Y. Higami, Y. Kurose, S. Ohno, H. Yamaoka, H. Takahashi, Y. Takamatsu, Y. Shimizu, and T. Aikyo, "Diagnostic Test Generation for Transition Faults Using a Stuck-at ATPG Tool," in *Proc. Int. Test Conf.*, 2009. Paper 16.3.

[11] Y. Higami, K. K. Saluja, H. Takahashi, S. Kobayashi, and Y. Takamatsu, "Compaction of Pass/Fail-based Diagnostic Test Vectors for Combinational and Sequential Circuits," in *Proc. ASPDAC*, 2006, pp. 75–80.

[12] I. Hartanto, V. Boppana, and W. K. Fuchs, "Diagnostic Fault Equivalence Identification Using Redundancy Information & Structural Analysis," in *Proc. International Test Conf.*, Oct. 1996, pp. 20-25.

[13] I. Hartanto, V. Boppana, J. H. Patel, and W. K. Fuchs, "Diagnostic Test Pattern Generation for Sequential Circuits," in *15th IEEE VLSI Test Symp.*, May 1997, pp. 196-202.

[14] H. K. Lee and D. S. Ha, On the Generation of Test Patterns for Combinational Circuits. Tech. Report 12-93, Dept. of Elec. Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.

[15] H. K. Lee and D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 9, pp. 1048–1058, Sept. 1996.

[16] Y.-C. Lin and K.-T. Cheng, "Multiple-Fault Diagnosis based on Single-Fault Activation and Single-Output Observation," in *Proc. Design, Automation and Test in Europe (DATE)*, 2006, pp. 424–429.

[17] Y.-C. Lin, F. Lu, and K.-T. Cheng, "Multiple-Fault Diagnosis based on Adaptive Diagnostic Test Pattern Generation," *IEEE Trans. on CAD*, vol. 26, no. 5, pp. 932–942, May 2007.

[18] V. P. Nelson, H. T. Nagle, B. D. Carroll, and J. D. Irwin, *Digital Logic Circuit Analysis & Design*. Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[19] D. Lavo and T. Larrabee, "Making Cause-Effect Cost Effective: Low-Resolution Fault Dictionaries," in *Proc. International Test Conf.*, 2001, pp. 278-286.

[20] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for Fault Location," in *Proc. Intl. Conf. Computer-Aided Design*, 1992, pp. 272-278.

[21] G. van Rossum and F. L. Drake, Jr., editors, Python Tutorial Release 2.6.3. docs@python.org: Python Software Foundation, Oct. 2009.

[22] P. G. Ryan, W. K. Fuchs, and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," in *Proc. Int. Conf. on Computer-Aided Design*, 1993, pp. 508–511.

[23] R. K. K. R. Sandireddy and V. D. Agrawal, "Diagnostic and Detection Fault Collapsing for Multiple Output Circuits," in *Proc. Design, Automation and Test in Europe (DATE'05)*, Mar. 2005, pp. 1014–1019.

[24] M. A. Shukoor and V. D. Agrawal, "A Two Phase Approach for Minimal Diagnostic Test Set Generation," in *Proc. 14th IEEE European Test Symp.*, May 2009, pp. 115-120.

[25] Y. Takamatsu, H. Takahashi, Y. Higami, T. Aikyo, and K. Yamazaki, "Fault Diagnosis on Multiple Fault Models by Using Pass/Fail Information," IEICE *Transactions on Information and Systems*, vol. E91-D, no. 3, pp. 675–682, 2008.

[26] A. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault equivalence and diagnostic test generation using ATPG," in *Proc. Int. Symp. Circuits and Systems*, 2004, pp. 221–224.

[27] 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 List," in *Proc. Design Automation Conf.*, pp. 133-138, June 1995.

[28] N. Yogi and V. D. Agrawal, "N-Model Tests for VLSI Circuits," in *Proc. 40th Southeastern Symp. System Theory*, 2008, pp. 242–246.

[29] X. Yu, M. E. Amyeen, S. Venkataraman, R. Guo, and I. Pomeranz, "Concurrent Execution of Diagnostic Fault Simulation and Equivalence Identification During Diagnostic Test Generation," in *Proc. 21st IEEE VLSI Test Symp.*, May 2003, pp. 351-356.

[30] X. Yu, J. Wu, and E. M. Rudnick, "Diagnostic Test Generation for Sequential Circuits," in *Proc. International Test Conf.*, 2000, pp. 226–234.

[31] Y. Zhang and V. D. Agrawal, "An Algorithm for Diagnostic Fault Simulation," in *Proc. 11th IEEE Latin-American Workshop,* March 2010.

**Table 1 Diagnostic Fault Simulation of ISCAS'85 benchmark circuits.**

| Circuit | Number of faults | Detection test Generation | | | | Diagnostic test Generation | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Detection vectors | *FC* % | CPU s* | *DC* % | Exclu. vectors | Abort pairs | Equv. pairs | Max. group size | *DC* % | CPU s* | CPU s** |
| c17 | 22 | 7 | 100.00 | 0.031 | 95.45 | 1 | 0 | 0 | 1 | 100.00 | 0.33 | 0.030 |
| c432 | 524 | 51 | 99.24 | 0.032 | 91.99 | 18 | 0 | 13 | 1 | 100.00 | 1.75 | 0.031 |
| c499 | 758 | 53 | 100.00 | 0.032 | 97.36 | 0 | 12 | 0 | 2 | 98.40 | 0.39 | 0.031 |
| c880 | 942 | 60 | 100.00 | 0.047 | 92.57 | 10 | 0 | 55 | 1 | 100.00 | 2.77 | 0.051 |
| c1355 | 1574 | 85 | 100.00 | 0.046 | 58.90 | 2 | 453 | 287 | 3 | 72.57 | 26.06 | 0.131 |
| c1908 | 1879 | 114 | 99.89 | 0.047 | 84.73 | 20 | 247 | 54 | 8 | 88.96 | 10.84 | 0.066 |
| c2670 | 2747 | 107 | 98.84 | 0.110 | 79.10 | 43 | 397 | 98 | 11 | 89.62 | 26.70 | 0.336 |
| c3540 | 3428 | 145 | 100.00 | 0.125 | 85.18 | 29 | 433 | 111 | 8 | 92.69 | 22.03 | 0.420 |
| c6288 | 7744 | 29 | 99.56 | 0.220 | 85.32 | 108 | 842 | 172 | 3 | 86.87 | 27.15 | 7.599 |
| c7552 | 7550 | 209 | 98.25 | 0.390 | 85.98 | 87 | 904 | 236 | 7 | 86.85 | 26.36 | 2.181 |

*Intel Core 2 Duo 2.66GHz, 3GB Ram. ** CPU time without rebuilding data structure