Practical Application of Automated Fault Diagnosis at the Chip and Board Levels

Mario Maccanelli  Andy Halliday  Bill Bell  Don Steiss  Kenneth M. Butler

Texas Instruments Instruments
P.O. Box 655303
Dallas, TX 75265

Abstract

As the sizes of electronic products grow larger, the process of diagnosing failed components becomes increasingly complex. The problem is compounded by the fact that there exists no unified system with which to diagnose problems at all levels of the product design: integrated circuit (IC), printed circuit board (PCB), and system.

In this paper we present the results of an industrial experiment with techniques for automating the diagnosis process. We have developed a prototype automated fault diagnosis (AFD) system which can input a fault dictionary from either of two different commercial ATPG systems along with results from test pattern application and produce a list of candidate defect sites within a given circuit. We ran our prototype against simulated single and multiple stuck-at faults in a portion of a commercial floating point unit and at the PCB level using a special test PCB. Our results have been encouraging in that we have obtained fairly accurate diagnoses with relatively low coverage stuck-at fault sets and in the presence of simulated non-classical defects. We have thus demonstrated that it is possible to produce a uniform methodology for AFD at the IC and PCB levels.

1 Introduction

The tremendous competition in the electronics marketplace is causing manufacturers to focus on speeding up product cycle times in order to gain and maintain market share. Each mass produced integrated circuit (IC), printed circuit board (PCB), or system goes through at least two phases of quality as the production line is "ramped up".

In the early phases of production, nearly all products are characterized by large scale or "catastrophic" failures. Typically, these failures are caused by oversights or problems in the manufacturing process which are easy to locate and correct due to the dramatic affect they have on the products. Gradually, though, products mature to the point in which failures are characterized by small, localized defects. The causes of these defects can be designed-in flaws, improper measurement during test, contamination from outside sources, and other similar phenomena. If, during this second and perhaps final phase of manufacturing, these localized defects occur repeatedly, the quality of the shipped goods may suffer. Clearly, then it is important to be able to quickly isolate and eliminate the causes of such defects. However, with today's ICs containing many millions of individual transistors and PCBs containing hundreds of ICs, the problem of isolating such small defects is often quite difficult simply due to the size of the product.
In order to solve these problems in the IC domain, researchers have proposed techniques for automated fault diagnosis (AFD) [1]-[8]. Typically, a database is created which records as much information as is feasible about which test patterns detect which modeled faults in the circuit. This database is commonly referred to as a fault dictionary. Generation, storage, and use of the fault dictionary are the central issues in rapid and accurate AFD.

While all of the techniques cited above have demonstrated successful results, it is not clear what the fault coverage requirements are to be able to use them. Some impose certain design constraints that must be met for the technique to be of use. Furthermore, it is desirable, at least for a vertically integrated manufacturer, to be able to use the same system to diagnose both ICs as well as PCBs.

In this paper we describe a prototype system for both IC and PCB level AFD. We discuss the algorithms used in the system and their general nature. The same basic system has been applied to both ICs and PCBs using two different mechanisms to collect the failure data. It has been successful at pinpointing fault locations even in the presence of non-ideal defects and low fault coverage test sets. We hope apply the system to actual failed silicon and implement a path to feedback the information to failure analysis so that fewer “unresolved” diagnoses occur in the future.

2 Objectives of AFD

It should be noted that the objective of fault diagnosis is very different at the IC level versus the PCB and system levels. At the PCB and system levels the objective of fault diagnosis is focused on repairing the fault. The diagnostic process searches for pin faults (opens and shorts) and must be performed very quickly (minutes). It is key that the diagnostic process locate either the exact fault (pin) or a point very close to the exact fault.

At the IC level, AFD is focused on improving the manufacturing process or design thereby preventing the fault from occurring in the future. There are many types of faults that can occur as shown in Table 2. The process has a longer term focus (days) of locating the fault. Therefore the objective of fault diagnosis is to significantly reduce the die area a technician must search to locate the fault. For example, with current equipment, a practical area in which to look for a fault is approximately 200 to 500 microns square. If a VLSI IC is 225mm square (15mm by 15mm) then diagnosis must reduce the relevant die area by a factor of between 500,000 and 1,000,000.

3 Requirements for AFD Use

The use of automated fault diagnosis imposes several requirements on the design environment if it is to be successful. There must be access to a tool which can simulate the candidate test patterns and log as detailed information as is possible about the modeled faults (typically stuck faults) in the circuit. This information should not only record that a fault has been detected by a pattern, but which pattern(s) detect it, and, if possible, where the detections occurred. If the AFD system is to be used at multiple levels of design, then the format for the failing information must be standardized, or multiple formats must be supported by the system.

The fault coverage of the patterns is of paramount importance as well. It may have to be very high to ensure that the fault is identified with great accuracy. Authors have described board level examples with 96% to 100% pin fault coverage requirements [9]. At the IC level, a lower fault coverage may be acceptable if the goal is only to isolate to a point near the actual defect. We will describe IC level experimental result with a fault coverage level of only 56%.

Other factors which affect the required level of fault coverage include:

- The accuracy required to identify the fault
- The acceptable time to visually verify the fault location.
- The distribution of the faults across the circuit or system.
4 AFD and Design Style

The style of design may also be affected. While AFD is applicable to random groups of logic, its effectiveness can be increased through the use of structured design. Typically, the more regular a design is, the more amenable it is to be diagnosed by automated means. This is exemplified by semiconductor RAMs and ROMs, where a failing vector can often be used to locate the exact point at which the defect has occurred. Table 4 summarizes many of the design characteristics which make an IC more or less amenable to diagnosis by automatic means.

<table>
<thead>
<tr>
<th>Item</th>
<th>More AFD Compatible</th>
<th>Less AFD Compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clocking style</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Test methodology</td>
<td>Scan</td>
<td>Ad hoc, Non-scan</td>
</tr>
<tr>
<td>Gate count</td>
<td>&gt;20K gates</td>
<td>&lt;20K gates</td>
</tr>
<tr>
<td>Pattern source</td>
<td>ATPG</td>
<td>Ad Hoc/Functional</td>
</tr>
<tr>
<td>Digital/Analog</td>
<td>Mostly digital</td>
<td>Analog</td>
</tr>
<tr>
<td>Max module size</td>
<td>&lt;10K gates</td>
<td>&gt;20K gates</td>
</tr>
<tr>
<td>Fault grade</td>
<td>&gt;70%</td>
<td>&lt;50%</td>
</tr>
<tr>
<td>Production volume</td>
<td>High volume</td>
<td>Lower volume</td>
</tr>
<tr>
<td></td>
<td>More expensive</td>
<td>Less expensive</td>
</tr>
<tr>
<td>Cost</td>
<td>More expensive</td>
<td>Less expensive</td>
</tr>
<tr>
<td></td>
<td>(greater benefits)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Design characteristics for AFD compatibility and non-AFD compatibility

5 Techniques for Diagnosis

A number of techniques for AFD have been described in the literature, most of them applicable at the IC level. These techniques are generally similar, and focus on methods to minimize the amount of fault simulator data that must be stored and scoring techniques to optimize the matching process. Richman et al. [3] and Ryan et al. [10] presented a technique for scoring possible detects when diagnosing sequential circuits. In the latter paper, the authors also proposed a method in which the test patterns are partitioned so that each partition covers only a subset of the possible circuit faults. When multiple “subtests” are failed by particular circuit, the relevant portions of circuitry are intersected so that a much smaller portion of the fault list must be fault simulated, resulting in faster simulations and less data to manage. Wacikauski et al. identified a set of criteria which must be satisfied in order for a candidate fault to be selected for fault simulation after the tester data has been gathered [6]. These criteria relate to specific information about the manner in which the circuit failed. These and other similar techniques are applicable to the general problem of AFD and could easily be implemented in our system as well. We will describe the algorithmic details of the system in Section 8.

6 Generality of AFD Approaches

Until the late 1980’s, automated fault diagnosis at the IC, PCB, and system levels could be considered independent requirements. This is no longer true due to an increase in the complexity
of all these systems. A key question is, "Are there common approaches that can be shared at all these levels of design?"

In order to facilitate the overall diagnostic process, a standard method to control and observe nodes at all levels is required. The JTAG or IEEE 1149.1 standard is emerging as the methodology of choice. This standard relies on the concept of boundary scan to observe and control device pins via a serial scan chain. The standard can easily be extended to control other test and diagnostic capabilities built into each IC. Figure 1 illustrates the concept of using JTAG to perform AFD at the IC, PCB, and system levels.

![Figure 1: Using automated fault diagnosis at all levels of design.](image)

7 Benefits of Automated Fault Diagnosis

There are a number of benefits to be realized with a fully functional automated fault diagnosis system. At the IC level, AFD can speed the "ramp up" process for a new semiconductor product. Once the causes of all of the catastrophic failures have been found and fixed, AFD can be utilized to locate isolated failures which can lead to diagnosing process problems, defects in the photomasks, clean room contamination, or other similar problems. AFD might also aid circuit timing analysis in that the defects may manifest themselves in a manner similar to static logic faults. The removal of the source of all of these errors can help bring a product to market more rapidly. At the PCB and system levels, AFD can not only be used to diagnose manufacturing and part handling problems, but it can be used in the field to facilitate easier technician troubleshooting and repair as mentioned in Section 2.

There are also a number of benefits to unifying the AFD approach across all levels of the system. These include:

- It is often difficult to isolate an IC failure from a board interconnect failure. It is beneficial to have common methods so that the fault isolation process can be iteratively applied until there is a high confidence that the fault has been found.
- High density boards are generally very delicate and only a limited number of rework cycles may be performed. It is critical that only failing ICs be removed.
- It is easy to damage an IC upon removal from a board, so it is desirable to locate the failed part precisely before removing any parts.
- With parallel and fault tolerant systems, it is more difficult to locate the failing component. It is better to be able to begin at the system level and traverse down the hierarchy.
- Once the system has been developed at one level, it is economically advantageous to use it other levels rather than develop separate systems.

Table 2 presents a pareto ranking of failure mechanisms of failed devices [11]. Based on experience estimates have been made on how well AFD would perform on finding faults for these failure mechanisms. "Min/Max % for AFD" columns indicate the minimum and maximum percent of devices that we would expect AFD to help with. "Overall Min/Max %" is the product of "% Failed" and "Min/Max % for AFD". AFD would have the most benefit for the electrical overstress/ESD and unresolved categories. The range of benefit varies considerably (3 to 1), and it is anticipated that to obtain the maximum benefit would require that AFD be considered as part of the overall design process.

<table>
<thead>
<tr>
<th>Failure Mechanism</th>
<th>% Failed</th>
<th>Min % for AFD</th>
<th>Max % for AFD</th>
<th>Overall Min %</th>
<th>Overall Max %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical Overstress/ESD</td>
<td>19.9</td>
<td>25</td>
<td>75</td>
<td>4.97</td>
<td>14.93</td>
</tr>
<tr>
<td>Unresolved</td>
<td>15.9</td>
<td>25</td>
<td>75</td>
<td>3.97</td>
<td>11.90</td>
</tr>
<tr>
<td>Gold Ball Bond Fail at Ball Bond</td>
<td>9.0</td>
<td>25</td>
<td>75</td>
<td>1.6</td>
<td>4.5</td>
</tr>
<tr>
<td>Not Verified</td>
<td>6.0</td>
<td>25</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gold Bond at Stitch-wire</td>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shear Stress-chip Surface</td>
<td>3.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrosion-chip, Metallia/Assembly</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dielec. Fail, Poly-Met, Met-Met</td>
<td>3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oxide Defect</td>
<td>2.9</td>
<td>40</td>
<td>60</td>
<td>1.16</td>
<td>1.74</td>
</tr>
<tr>
<td>Visible Contamination</td>
<td>2.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metal Short, Metal Open</td>
<td>2.6</td>
<td>10</td>
<td>25</td>
<td>.36</td>
<td>.65</td>
</tr>
<tr>
<td>Latch-up Likely</td>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misprocessed-Bond Related</td>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip Damage, Cracks, Scratches</td>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misprogrammed</td>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oxide Instability</td>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design-chip</td>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diffusion Defect</td>
<td>1.5</td>
<td>25</td>
<td>75</td>
<td>.375</td>
<td>1.13</td>
</tr>
<tr>
<td>Final Test Escape</td>
<td>1.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact Failure</td>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bond Failure, Non-Gold</td>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protective Coating Defect</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembly-Other</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polysilicon/Silicide</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Contamination</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td>100</td>
<td>12.2</td>
<td>34.86</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Pareto ranking of failure mechanisms of failed devices (total number of failed devices = 3400).
8 Algorithmic description

The first phase in the process of creating a fault dictionary is to identify the target faults for
the device under test (DUT). Acceptable target faults are a function of the fault types that are
comprehended by the fault simulator or automatic test pattern generator (ATPG). Most fault
simulators and ATPGs consider only stuck-at faults on the inputs or output pins at the gate and
sometimes the transistor level.

The amount of processing and disk space required for the creation of a fault dictionary is a
function of the number of target faults. Fault collapsing ([12], [13]) is usually performed to reduce
the number of faults that must be included in the fault dictionary. As the number of faults that
collapse together increases, the "granularity" of the fault isolation decreases. In other words, as
the cardinality of the fault class (the faults represented by only one fault in the dictionary) goes
up, the size of the area to which one may isolate the defect also increases.

The fault simulation process is accomplished by using the test patterns and a modified copy
of the circuit description to produce the faulted patterns. The circuit modifications for the fault
simulation are obtained by incorporating one fault class at a time into the circuit. The faulted
patterns are compared against the expected patterns. If the faulted patterns for a fault class
match the expected patterns then the fault has not been detected by the test patterns, and additional test
patterns are required.

The iterative process of test pattern creation, good circuit simulation and fault simulation con-
tinues until either the number of faults detected have reached the fault detection goal, or the time
allocated has expired.

Storing the faulted patterns in the fault dictionary is one approach that can be taken. Alterna-
tively, one may choose to store only the differences between the faulted patterns and the expected
patterns. Using this abbreviated format will reduce the time that is required to perform the AFD.

Since both the expected patterns and the faulted patterns are of the same fixed width, each
position in the patterns can be assigned a unique number. The faulted patterns can then be
represented as a list of pattern positions, each position indicating a difference between the expected
pattern and the faulted pattern.

Fault detection uses the previously identified test pattern set to validate the DUT. If the output
patterns from the DUT match the expected patterns, no faults have been detected, and consequently
no fault isolation may be performed. If the DUT patterns do not match then a fault has been
detected.

The same process that is used to convert the fault class patterns into a list of differences is
also used to convert the DUT patterns into a list of differences from the expected patterns. The
DUT patterns are compared against the patterns of each fault class. A score can be calculated to
express how precisely the fault class differences and the DUT differences match. A fault callout list
is obtained by sorting the fault classes by their corresponding scores, the lowest scores identifying
the the most likely faults.

The size of the fault dictionary is often of great concern. Fault collapsing helps reduce the size of
the fault dictionary. Another approach that can also be used is to limit the number of differences
that are stored for each fault class. This approach will also decrease the amount of time needed
during fault isolation. Care must be taken, however, in reducing the number of differences to
retain. Too small a number could render the fault dictionary useless. A good approximation to the
optimal number of differences to retain can be determined by comparing the fault class differences
and noting how many differences are required to uniquely identify each fault class. If the differences
for two or more fault classes are identical, then either additional patterns are required to isolate
between the fault classes, or the fault classes could be collapsed.

Using our technique for fault dictionary storage, the approximate size of the fault dictionary can
be calculated by the following formula:

\[ FD_{\text{size}} = FC_{\text{test}} + (FC_{\text{max}} \cdot \Delta_{\text{max}}) + ((\text{Vectorcount} \cdot \text{Vectorwidth}) / \text{BITs/\text{WORD}}) \]
where $FD_{dict}$ is the size of the fault dictionary in bytes, $FC_{text}$ is the number of bytes needed to store the textual description of the fault classes, $FC_{max}$ is the number of fault classes for this fault dictionary, $delta_{limit}$ is the limit on the number of differences retained for any fault class, $Vectorcount$ is the number of input patterns, $Vectorwidth$ is the number of output signals, and $BITsinWORD$ is the number of bits in a machine word of the computer.

9 Experimental Method

The system has been tested at both the IC and PCB levels. Each of the experimental procedures and results will be described below.

9.1 Chip Level Experiments

We ran an experiment with the AFD system to isolate faults at the IC level. The initial phase of the experiment was to use the fault dictionary to isolate faults inserted using a fault simulator to simulate the faults and generate the faulted circuit responses. The second phase of the experiment will be to use the fault dictionary to isolate faults in actual silicon, and should be completed later this year.

The circuit that was selected for the experiment contains 4005 transistors, or approximately 1000 gates. A scan path can be used to isolate the test circuit from the remainder of the part. The scan path controlled 99 input pins to the circuit, and observed 76 output pins. TESTSCAN, a product of Cadence Corporation, was used to create the input patterns, as well as a fault dictionary. The test patterns created by TESTSCAN had a fault coverage of 58% at the transistor level.

A different fault simulator, Zycad, was used to simulate the silicon for the experiment. The input patterns were able to detect faults for four of the five test cases.

Fault isolation was performed for the four test cases that were able to detect the fault. More than one fault callout was given for each of the test cases. Each correct callout was within one gate of the actual fault that was inserted. The incorrect callouts were all within five gates of the inserted fault.

9.2 Board Level Experiments

We also conducted an experiment designed to determine the feasibility of using ASSETTm in conjunction with fault dictionary software to perform PCB level fault detection and isolation. The ASSETTm system would be responsible for applying the input patterns and collecting the DUT patterns. The AFD system would then perform fault isolation using the DUT patterns.

The circuit that was selected for the experiment contained a mixture of JTAG compliant and noncompliant parts. The circuit was designed with six "fault switches" in order to simulate faults in hardware. The fault simulator and ATPG that were selected for this experiment was the INTELLIGEN/CADAT toolset from Racal-Redac.

Of the six faults implemented on the test board, one fault was a corruption of the IEEE 1149.1 data bus, one was a bridging fault, one was an intermittent fault, and the remaining were either stuck-at-1 or stuck-at-0 faults. The AFD system was able to identify the IEEE 1149.1 data bus fault, but was unable to isolate the fault. The bridging fault was identified and isolated to within two gates. When the intermittent fault was present while ASSETTm applied the patterns, the fault dictionary software was able to isolate the fault to within one gate. The fault dictionary software isolated the remaining faults in an average of two gates.
10 Conclusions

We demonstrated the generality of AFD techniques for the IC and PCB levels [9]. At the IC level, a fault dictionary was created based on a relatively low fault coverage test pattern set. Remarkably, the resolution of the fault dictionary was still fine enough that the AFD system was able to produce diagnoses at or very near the fault site in every trial.

At the PCB level, even when simulating non-classical faults such as bridging and intermittent faults in the presence of a stuck-at-only fault dictionary, our AFD system was successful at diagnosing defects to within close range of the actual fault site. We hope to extend this work to diagnosing actual silicon in the near future to further prove the feasibility of the concept.

11 References


