Fault Pattern Analysis Suggests Failure Modes

James Tan Swee Chuan - January 01, 1999

You can employ a fault-symptom-recognition technique to help determine the possible causes of in-circuit test failure. You can implement this technique using a relational database, which test operators can query to help identify bad devices.

The need for such a technique arises because a test operator can have trouble interpreting failure-ticket data to isolate a failed component. An in-circuit tester lists all the components whose measured values exceed specified limits, even though most of these components are not faulty; they only appear so because they connect directly or indirectly to the actual fault node.

Figure 1. A fault simulated by a 10-kV shunt from node 1 to ground resulted in an in-circuit tester identifying the three supposedly failed components listed in Table 1. In this example, the tester returned the value of 15.398k (for R34) instead of the expected 1k, possibly because of stored charges in capacitor C30. Despite the difficulty in explaining why a tester returns certain values, you can derive patterns from the values that can assist in board rework.

Although failure-ticket data seldom explicitly pinpoints one bad device or node, you can use the data to derive patterns that can help you locate similar faults in subsequent tests. Consider an example in which, for the circuit shown in Figure 1, a tester reports three failed components: R34, R35, and R74. These components’ nominal and measured values are shown in columns 2 through 5 of Table 1. The actual fault—a 10-kV shunt to ground from node 1—isn’t apparent from the Table 1 values and must be painstakingly located. Once located, however, you can correlate that fault with a fault pattern derived from the failure-ticket data to expedite fault detection on subsequent tests.
A three-step process derives the pattern:

1. For each component rejected by the tester, calculate the relative deviation $D$:

$$D = \frac{|V_M - V_N|}{V_N} \times \left(\frac{100\%}{T}\right) \text{ for } V_M > 0 \text{ and } T > 0$$

(Eq.1)

where $V_M$ is the component value measured by the tester, $V_N$ is the nominal component value, and $T$ is the component tolerance in percent.

2. Rank the components according to their relative deviation (see the rightmost column of Table 1), highest value ranked number 1.

3. Create the fault pattern, an ordered set of the component designators based on their relative-deviation ranking. The Table 1 data yields the following fault pattern:

Fault Pattern = \{R34, R35, R74\},

where R34 (with the highest relative deviation) is ranked first and R74, with the lowest relative deviation, is placed at the right most position.

The next step is to store the fact that this pattern has corresponded to a particular fault in a history file:

Fault Pattern = \{R34, R35, R74\} $\Rightarrow$ Actual Fault = “R35 node 1 shunted to ground.”

On subsequently encountering this pattern, the test operator can deduce a high probability that the actual fault is “R35 node 1 shunted to ground.”

### Practical Fault Pattern Recognition

In practice, fault patterns may vary from board to board even though the boards have the same fault. To illustrate this concept, I ran repeated tests, yielding these fault patterns:

P1=\{R34, R35, R74\}
P2=\{R34, R74, R35, C41\}
P3=\{R34, R35, R74, C41, R173\}

In pattern P1 and P2, R35 and R74 have swapped rank, a typical phenomenon in in-circuit testing. P3 maintains P1’s R34, R35, and R74 ranking but adds C41 and R173, whose measured values fell
slightly off the test limit.

Although P1, P2, and P3 are slightly different, all correspond to the same actual fault. The pattern-recognition technique should recognize that P3 is similar to our original fault pattern \{R34, R35, R74\} and hence may suggest that “R35 is shunted to ground.”

You can implement this idea using a relational database system. Consider Table 2, which stores three fault patterns. You can query this fault pattern table as follows:

• First, assume that a test yields the following pattern:

  P4 = \{R34, R35, R74\}

• Then, issue Query 1:

  SELECT * FROM PATTERN
  WHERE Rank1 = “R34”
  AND Rank2 = “R35”
  AND Rank3 = “R74”;

which will generate the result shown in Table 3.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Rank 1</th>
<th>Rank 2</th>
<th>Rank 3</th>
<th>Rank 4</th>
<th>Rank 5</th>
<th>Actual Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R34</td>
<td>R35</td>
<td>R74</td>
<td>—</td>
<td>—</td>
<td>R35 pin 1 shunted to ground</td>
</tr>
<tr>
<td>3</td>
<td>R34</td>
<td>R35</td>
<td>R74</td>
<td>C41</td>
<td>R173</td>
<td>R35 pin 1 shunted to ground</td>
</tr>
</tbody>
</table>

Table 3. Query Results

Here, we have an exact match for test No. 1, a partial match for test No. 3, and a mismatch for test No. 2 (and hence it is not shown in the query result set).

Assume that another test yields the following pattern:

P5 = \{R35, R34, R74\}.

Applying Query 1 generates no result, because there is no direct or partial match. We can modify the query, however, so the selection criteria are more flexible:

SELECT * FROM PATTERN
WHERE ((Rank1 = “R34”) OR (Rank1 = “R35”))
AND ((Rank2 = “R35”) OR(Rank2 = “R74”) OR(Rank2 = “R34”))
AND ((Rank3 = “R35”) OR(Rank3 = “R74”)).

This query, which includes the OR conditional operator, generates the Table 4 result, with fuzzy matches for tests 1, 2 and 3.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Rank 1</th>
<th>Rank 2</th>
<th>Rank 3</th>
<th>Rank 4</th>
<th>Rank 5</th>
<th>Actual Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R34</td>
<td>R35</td>
<td>R74</td>
<td>—</td>
<td>—</td>
<td>R35 pin 1 shunted to ground</td>
</tr>
<tr>
<td>2</td>
<td>R34</td>
<td>R74</td>
<td>R35</td>
<td>C41</td>
<td>—</td>
<td>R35 pin 1 shunted to ground</td>
</tr>
<tr>
<td>3</td>
<td>R34</td>
<td>R35</td>
<td>R74</td>
<td>C41</td>
<td>R173</td>
<td>R35 pin 1 shunted to ground</td>
</tr>
</tbody>
</table>

Table 4. Conditional-Query Results
Completing the System
This article introduces the concept of relational database implementation. A complete implementation would require additional steps:

• Establish a data link between the in-circuit tester and the system on which the database resides.

• Develop a procedure that converts failure-ticket data to the fault pattern.

• Develop a point-and-click graphical user interface to make it unnecessary for an operator to type structured-query-language statements to determine possible faults.

In addition, the operator should be aware that the fault diagnosis provided by the system is suggestive, not definitive. In fact, two distinct faults might produce identical fault patterns, requiring the test operator to locate the fault through trial and error or make a judgment based on close examination of detailed failure-ticket data. Furthermore, the relative-deviation ranking technique described here is applicable for analog components; digital components may be ranked according to the sequence in which they appear in the failure ticket.

This method provides knowledge retention, storing fault patterns and their associated actual faults in a centralized, shared database. This system would be particularly useful during production ramp-up, where yield is normally low because the manufacturing process is unstable and because test operators are unfamiliar with the product. T&MW

James Tan Swee Chuan is a test engineer at Singapore Technologies Electronics. He is currently investigating the use of neural networks to implement the fault-pattern analysis technique discussed in this article. He invites readers to send him comments at tansc@stee.com.sg.