A SYSTEMS APPROACH TO SPECIFYING BUILT-IN TESTS

John W. Sheppard and William R. Simpson

ARINC Research Corporation
2551 Riva Road
Annapolis, MD 21401

ABSTRACT

The need for specifying robust built-in test for systems is growing as systems become more complex. Further, detection-only BIT—the predominant form of BIT—is insufficient to meet the needs of system test on current and future systems; localization and even isolation is becoming essential. In response to this need, several computer-based analysis tools have become available that provide the ability to assess system testability and BIT effectiveness. Yet few if any formal methods exist for providing optimal BIT specifications. In this paper, we explore two BIT figures of merit—test point utilization and optimized resolution analysis. We will describe evaluation measures which we then apply to BIT specified using these two approaches and explain the resulting differences.

1. INTRODUCTION

Increased complexity of modern electronic systems has resulted in greater emphasis on specifying robust built-in test (BIT). Historically, BIT provided fault-detection capability with limited fault-localization. In broad terms, detection refers to the ability to determine through a set of tests or information sources whether a failure has occurred within the system. This has been the primary design criterion for BIT, with measures such as detection percent (DP), BIT coverage, and false alarm rate all being applied to determine the effectiveness of BIT[1]. Detection percent and BIT coverage frequently are used interchangeably but actually refer to the percent of failure modes (detection percent) or system components (BIT coverage) that will be detected when a failure occurs. False alarm rate attempts to predict the percent of alarms in a system that BIT will identify as faults when in fact no fault exists.

Studies have shown that it is not possible to predict false alarm rate for various reasons[2]. As a result, cannot duplicate (CND) rates are being used instead. The difficulty associated with predicting CND rate for BIT is that the results of BIT are generally verified at the next level of maintenance[3,4]. CNDs apply at the same level of maintenance. In this case, it may be more useful to attempt to predict retest OK (RTOK) rates which relate specifically to a change in maintenance level.

Recent trends in systems engineering are resulting in modern systems having maintenance architectures with fewer levels. For example, aircraft avionics maintenance may take place by sending units from the aircraft directly to the manufacturer. Consequently, it is becoming increasingly important to isolate to higher resolutions

(i.e., smaller replaceable parts) at the operational level of maintenance. At the same time, shrinking budgets will require this to be done with far greater automation.

The emphasis on developing detection-only BIT on current and future systems is no longer adequate; localization and even isolation is becoming essential. Localization concerns the ability to restrict the set of candidate failures causing the BIT indication. Obviously, some localization occurs in all BIT, but it has been a byproduct of the BIT development process rather than a design criterion. In the cases where localization has been specified in the BIT design, the industry has labelled these systems as *smart* BIT. Almost no BIT systems have been specified in which the goal is to provide fault isolation, i.e., the identification of the specific fault causing the BIT indication. Typically, isolation is defined to be the ability to localize to a point consistent with the current level of repair. In other words, localization should be sufficient for identifying a single unit to be repaired at the operational level (in the case of BIT).

In response to the need for improved BIT capabilities, several computer-based analysis tools have become available that provide the ability to assess system testability and BIT effectiveness. Two such tools are the Navy's Weapon System Testability Analyzer (WSTA) and ARINC's System Testability and Maintenance Program (STAMP)[5,6]. Both of these tools apply what the industry calls information flow models or dependency models to describe the diagnostic capability of a system, assess the effectiveness of a set of tests for that system, and provide optimized diagnostic strategies using the available test resources.

This paper discusses approaches to solving a difficult problem in system test, namely the specification and use of tests for BIT. It describes two specification algorithms in detail and demonstrates the capabilities of the algorithms on actual examples. It then assesses the advantages and disadvantages of the algorithms. Procedures and algorithms such as those described in this paper are a necessity for developing BIT on complex systems of today and tomorrow.

2. METRICS FOR BIT SPECIFICATION

In spite of the availability of more and better analysis tools, few if any formal methods exist for providing optimal BIT specifications. In response to this need, at least two approaches to specifying BIT have been developed. Both approaches are based on the assumption that BIT resources must be minimized, and they both provide methods for eliminating ''unneeded'' tests from the BIT specification. The first approach has become known as test point utilization (*TPU*). This approach is derived from examining the frequency of test and test point use in a fault tree to determine whether or not to include a specific test in BIT. The second approach is called optimized resolution analysis (*ORA*). This approach focuses on several requirements of BIT to provide complete detection and maintain maximum expected ambiguity resolution.

2.1 Test Point Utilization

In determining BIT for a system, we assume that we have developed a dependency or information flow model for that system which includes specifying a set of candidate tests to be used in BIT. Once the model has been defined, the test point utilization metric is used to determine which of the tests will be BIT as follows. First a diagnostic decision tree is developed with the candidate BIT tests using some optimization procedure

(e.g., information gain). The tree may include various cost weights, but the use of weights at this point should be considered with care. Once the tree has been generated, the number of times each test is used in an isolation sequence is counted, and the tests are sorted based on these counts. If only n tests can be used in BIT, then the n tests with the highest counts (i.e., the most frequently utilized tests in the tree) are selected.

More formally, we can compute the *TPU* for a test t_j (TPU_j) as follows. Let **S** be the set of isolation sequences in a fault tree $\Im$. Let **U** be the set of unique fault isolation conclusions in the tree. Each $u_i \in \mathbf{U}$ will appear as a leaf of $\Im$. Clearly $|\mathbf{S}| = |\mathbf{U}|$. Let $s_i \in \mathbf{S}$ be the i^{th} isolation sequence, and let it terminate by concluding u_i. Finally, if us_j is the set of isolation sequences in which t_j occurs, then $TPU_j = |us_j|$.

The advantage of this approach is its apparent simplicity to implement. Unfortunately, *TPU* provides the basis of an approach to specifying BIT that has the potential of unduly increasing the ambiguity between potential faults in a system. Worse, capricious elimination of tests such as in this approach could conceivably decrease the detection capability as well. This, of course, is unacceptable. In this paper, we will provide several examples demonstrating the problems associated with test point utilization. A more precise method for determining which tests to use as BIT must be developed that permits developing efficient BIT, minimizes ambiguity (or expected ambiguity), and maintains (if possible) complete fault detection.

2.2 Optimized Resolution Analysis

The optimized resolution analysis also uses the information flow model and proceeds from three assumptions:

- Since we are attempting to minimize BIT resources we need not consider tests providing redundant or excess information.
- Detection is still essential in BIT; therefore, specification of BIT tests must maintain maximum possible detection. Any unnecessary degradation of detection is unacceptable.
- Maximum isolation capability occurs when we have minimum ambiguity. This can be interpreted probabilistically as well.

In response to these three assumptions, the optimized resolution analysis method proceeds in three steps using the information flow model. First, an excess test analysis is performed to eliminate tests providing redundant or excess information. A redundant test is defined to be a test that provides exactly the same information as another single test. An excess test is defined to be a test that provides exactly the same information as a combination of two or more other tests. We will provide a detailed algorithm for determining redundant and excess tests.

The second step in the *ORA* algorithm is to generate a fault tree using all of the remaining BIT tests. This tree may be optimized according to whatever cost criteria are appropriate; however, BIT usually has uniform cost and time. This means the most often used criteria would be failure rate, thus emphasizing failure probability in deriving the decision tree. In addition, the tree should be generated in such a way that the fewest expected number of tests on the go-path will be generated. After the tree is

generated, the number of tests on the go-path is determined. If this number exceeds the number allowed, all other tests are eliminated from the tree, and tests are eliminated from the go-path starting at the end of the path until the number of tests remaining equals the allowance specified.

If there are sufficient tests available to construct a complete go-path but insufficient tests to produce a complete decision tree, then ambiguity must increase as a result of eliminating tests to meet the allowance. Following the third assumption, the *ORA* algorithm prunes the tree starting at the leaves, excluding the go-path from consideration[7,8]. The pruning algorithm maintains minimum expected ambiguity by examining the failure rates at the internal nodes of the trees and pruning the nodes with the lowest probability of failure.

More formally, as in the *TPU* calculation, we let $\Im$ be a fault tree generated to fault isolate a system using proposed BIT tests. We assume the fault tree has been generated using a model such as the information flow model described by Simpson and Sheppard[9]. Let **I** be the set of information sources available, **F** be the complete set of fault isolation conclusions (including ambiguous conclusions), and $\mathbf{D}^*$ be the closed dependency matrix for the model of the unit under test. For each test t_i, compute the expected information value of that test $E[\phi_i]$ as follows:

- Compute the information gained when t_i passes (g_i)

$$g_i = 1 - \sum_{j=1}^{|\mathbf{I}|} \alpha_{ij}$$

where

$$\alpha_{ij} = \begin{cases} 1 \; ; (\mathbf{D}^*_{ij} = 1) \wedge (i \neq j) \\ 0 \; ; \text{otherwise} \end{cases}$$

- Compute the information gained when t_i fails (b_i)

$$b_i = 1 - \sum_{j=1}^{|\mathbf{I}|} (\beta_{ij} \cdot \gamma_{ij} \cdot \delta_{ij})$$

where

$$\beta_{ij} = \begin{cases} 1 \; ; (\mathbf{D}_{ji} = 1) \wedge (i \neq j) \\ 0 \; ; \text{otherwise} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1 \; ; (\mathbf{D}^*_{ij} = 0) \wedge (i \neq j) \\ 0 \; ; \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1 \; ; (\gamma_{ij} = 1) \wedge \\ \quad (\mathbf{I} - \{t_j\} \Rightarrow |\mathbf{G}'| > |\mathbf{G}|) \\ 0 \; ; \text{otherwise} \end{cases}$$

where **G** is the set of ambiguity groups and **G** is the set of ambiguity groups after t_j is removed.

- Compute the information value of the test $\phi_i = \min \{g_i, b_i\}$.
- Compute the expected information value $E[\phi_i]$ using probabilities of failure for the conclusions upon which t_i depends.
- Rank the tests in nondecreasing order by $E[\phi_i]$.
- Processing the tests in order, if $|\mathbf{I}| - \{t_i\} \Rightarrow |\mathbf{G}'| > |\mathbf{G}|$, then $ORA_i = 0$.
- With the remaining $\mathbf{I}'$ (i.e., set of information sources with redundant and excess tests removed), generate $\Im$, applying the hypothesis directed search to generate the go-path with *No Fault* as the hypothesis[14].
- Let **S** be the set of isolation sequences.
- Let s_m^j be the m^{th} isolation sequence up to t_j (noninclusive).
- Let a_m^j be the set of fault isolations in ambiguity given s_m^j.

- Let $\lambda_{m}^{j} = \sum_{\forall f_k \cdot a_m^j} \lambda_{f_k}$

- $ORA_m^i = \lambda_m^i$ (Note: each internal node of $\Im$ has an *ORA* value.)

3. EXPERIMENTS

In order to evaluate the two BIT specification measures, experiments were run on two diagnostic models to determine the effects of following each measure in determining which tests to retain for BIT. The first model is a hypothetical model used in a series of articles describing a formal approach to integrated diagnostics. The model is derived from an anti-tank missile launcher circuit. The model consists of 20 possible tests and 26 fault isolation conclusions. The experiments considered the model given a set of failure rate data for the fault isolation conclusions[9-14].

The second model comes from a real analysis performed on the AN/SQS-53C Active Receive Beamformer for the U.S. Navy. The primary objective of the analysis was to assess system testability and evaluate a set of program monitoring/fault detection/fault location (PM/FD/FL) tests. The analysis procedure described in this paper was not available for this project, so we performed the analysis as a test case of the algorithm. The SQS-53 model has approximately 300 fault isolation conclusions, but only 18 PM/FD/FL tests. Also, no failure rate information was available, so we assumed uniform probability of failure in our analysis.

The experiments proceeded in four steps. First, fault trees were constructed for each model. Second, after the fault tree was generated, we computed the *ORA* and *TPU* measures and ranked the tests according the each criterion. Note that for *TPU*, detectability is not an issue, and this had a direct outcome on the comparison. Third, we sequentially eliminated one test at a time from the model. Fourth, after each test was eliminated, we computed three testability statistics to determine the impact of ordering BIT tests based on these criteria.

The three testability measures used to evaluate *TPU* and *ORA* were non-detection percent (*ND*), operational isolation (*OI*[1]), and isolation level (*IL*). Non-detection percent is the ratio of undetectable faults to the total number of faults and is computed as

$$ND = \frac{\left(\sum_{i=1}^{|\mathbf{F}|} \delta_i\right) - 1}{|\mathbf{F}|}$$

where

$$\delta_i = \begin{cases} 1 ; (SF_i = SF_{\text{no Fault}}) \wedge (f_i \in \mathbf{F}) \\ 0 ; \text{otherwise} \end{cases}$$

and SF_i is the fault signature of fault i. The first operational isolation measure (*OI*[1]) is the percentage of the time one is able to fault isolate to exactly one fault conclusion. This measure is computed as

$$OI[1] = \frac{\sum_{i=1}^{|\mathbf{F}|} w_i \gamma_i}{|\mathbf{F}|}$$

where

$$\gamma_i = \begin{cases} 1 ; \alpha_i \geq 1, \forall f_j \in \mathbf{F} \\ 0 ; \text{otherwise} \end{cases}$$

$$\alpha_i = \sum_{k=1}^{|\mathbf{BU}|} \beta_k$$

$$\beta_k = \begin{cases} 1 ; \exists f_j \in \mathbf{RU}_k \ni SF_j = SF_t \\ 0 ; \text{otherwise} \end{cases}$$

RU = the set of replaceable units (for our analysis, the set of individual conclusions). There is, in general, one value of *OI*, *OI*[*n*], for each of *n* levels of ambiguity.

Finally, isolation level is the ratio of the number of isolatable groups (i.e., ambiguity groups) to the total number of fault isolation conclusions and is computed as

$$IL = \frac{|\mathbf{UF}|}{|\mathbf{F}|}$$

where **UF** is the set of *unique* fault isolation conclusions.

4. RESULTS

Comparing the results of eliminating tests based on *ORA* and *TPU* indicate strong differences with respect to detection while isolation capability is comparable. For the hypothetical system, maximum detection was maintained throughout (Figure 1). This is because only one test is required to detect all detectable faults and the *ORA* does not recommend eliminating go-path tests until absolutely necessary. *TPU*, on the other hand, does not differentiate between a detection test and an isolation test; therefore, nondetection increases substantially as tests are removed. For the SQS-53, this conclusion is also supported, but there was not a single test to detect all failures (Figure 2). Consequently, the differences between *ORA* and *TPU* are not as pronounced.

Operational isolation tends to track fairly closely between *ORA* and *TPU* for the two systems, but a couple of interesting observations can be made. First, we should expect *OI*[1] to be identical for the two BIT measures when we have all of the tests available and when we have a minimum set of tests available. This is because the isolation potential is identical before deleting any tests and because *OI*[1] converges to 0.0. In both of the systems examined for this paper, *OI*[1] indeed converged to 0.0. Another interesting difference, however, becomes evident when examining the hypothetical model (Figure 3). This model includes failure rate information (the SQS-53 did not), and *OI*[1] includes weighting (e.g., failure probability) in its calculation. When the weights are not uniform, we would expect the *ORA* to yield better results, and this is demonstrated in the hypothetical system. The SQS-53, on the other hand, shows performance alternating between *ORA* and *TPU*, but they perform very closely (Figure 4).

Finally, the isolation level measure shows almost no difference between the *ORA* and the *TPU*, but the difference is significant. (Note that *IL* has the same convergence properties as *OI*[1].) Both the hypothetical system (Figure 5) and the SQS-53 (Figure 6) show *ORA* with a slight advantage, but (with two exceptions on the hypothetical system), *ORA* is consistently equal to or better than *TPU*. Thus in all three cases, using *ORA* to determine the BIT tests to use provides a better result than *TPU*.

5. DISCUSSION

The advantages of the *ORA* algorithm over the *TPU* algorithm should be evident. First, detection is paramount to effective BIT. *ORA* ensures maximum possible detection is maintained by focusing on using tests to minimize ambiguity with detection receiving the highest priority. *TPU*, on the other hand, has no control over ambiguity groups that may be performed which in turn has the potential of decreasing detection ability of BIT. Second, *ORA* is concerned with

minimizing the impact of eliminating BIT tests on ambiguity. *TPU*, on the other hand, is more interested in maximizing the use of available tests at the expense of ambiguity. Third, the *ORA* procedure is well suited to incorporating with other optimization procedures so that the types of ambiguity to be minimized and the types of test resources to be utilized can be taken into account. To some extent, *TPU* can incorporate this information as well, but it does not. Fourth, *ORA* (and to some extent *TPU*) can incorporate multiple failure diagnostics.

Once the BIT tests have been specified, BIT will be used in a system to detect, localize, and ideally isolate faults as they occur. Historically, BIT has been designed using a table-lookup approach where a BIT signature is matched against entries in a fault dictionary to determine the fault detected. Approaches based on the information flow model can be used to construct decision procedures using the available BIT tests. In fact, these procedures (as indicated above) form an integral part of the BIT recommendation process.

The premise behind the diagnostic process is that table lookup operations are more expensive than they need to be. Without an architecture such as a content-addressable memory in which entries can be found in constant time, some sequential analysis of the BIT readings is required to make a diagnosis. Further, in many cases, the BIT values are determined in a sequential fashion rather than in parallel. Diagnostic algorithms exist that optimize the sequential query of BIT values for performing diagnosis[13]. These algorithms use information as a figure of merit for determining BIT attributes to evaluate, and the information figure of merit can be weighted by appropriate cost criteria. In addition, mechanisms for rapid inference exist that minimize the effects of sequential processing and permits parallel inference to take place. These algorithms can be implemented in hardware in a straight forward way.

6. CONCLUSIONS

In this paper, we discussed and compared two algorithms for determining an optimal set of built-in tests for a system. The approaches rely on the ability to analyze a fault tree generated in one of several ways. The test point utilization metric is used from fault trees and does not require a particular form of tree to be generated; most systems that use *TPU* do not apply any special constraints for generating the tree. Considerable improvement can be achieved with *TPU* if a few changes are made in its use. First, since detection is paramount for reliable BIT, *TPU* should prefer tests on the go-path rather than treating all tests equally. Second, trees can be generated that optimize the use of tests based on several criteria. The most effective criterion for optimization of BIT when an isolation capability is required is failure probability (frequently provided in terms of failure rate). Failure probability weighted trees will result in improved performance for *TPU* since tests examining higher probability failures will appear closer to the root of the tree and thereby have higher *TPU* values.

Aside from the approaches to improving *TPU* for determining BIT tests, *TPU* has problems as demonstrated in this paper. One potential problem not discussed arises after the set of tests has been determined. If we develop a tree using all of the tests and then eliminate several tests based on *TPU*, using the resulting set of tests to generate the tree may lead to a tree that is not consistent with the original tree. Ideally, the tree resulting after eliminating undesirable tests should be

the subtree induced by the set of tests retained. Unfortunately, the *TPU* may preclude this in the event it recommends the elimination of a test that is the parent of a test that has been retained. This can only happen if a test appears more than once in the tree, and we encountered this in the trees generated for our two examples.

Because the *ORA* is calculated from the leaves of the tree upward toward the root and the *ORA* of any internal node of the tree is strictly greater than the *ORA* of any of its children, most tree development algorithms will develop subtrees induced by the subset of tests retained from applying the *ORA*. This is because no test will be recommended for deletion if any of its descendants are still in the tree. This represents a clear advantage of *ORA* based test sets over *TPU* based test sets.

In addition to the potential improvements to *TPU* and the difficulty associated with generating subtrees, it is clear that the *ORA* analysis led to a better set of tests to be retained for BIT than *TPU* did. Further, other measures can be considered for comparison that were not included in this paper due to space limitations. For example, the *OI*[*n*] measures identify the impact of eliminating tests on the ability to isolate to *n* or fewer units. Related to this measure is the maximum size of the ambiguity groups that result. Other measures that may be considered for BIT include the tolerance to false alarms and susceptibility to false failure indications. These will be considering in future work on BIT specification.

ACKNOWLEDGEMENTS

The authors would like to thank several people for their assistance in the preparation of this paper. First, thanks go to Michael Seldes who indicated an interest in evaluating the performance of test sets for meeting special requirements. We also thank Jerry Hadfield, Brian Pickerall, and Broady Cash who constantly bring interesting problems to our attention. Finally, we wish to thank Sheryl Sieracki and Elizabeth Reed for their comments on an early draft of this paper.

REFERENCES

1. Coppola, Anthony, *A Design Guide for Built-in Test (BIT)*, RADC-TR-78-224, Rome Air Development Center, April 1979.
2. Simpson, W. R., and J. W. Sheppard, ‘‘Analysis of False Alarms During System Design,’’ *Proceedings of the National Aerospace Electronics Conference*, Dayton, Ohio, May 1992.
3. Hughes Aircraft Company, *Analysis of Built-in Test (BIT) False Alarm Conditions*, RADC-TR-81-220, Rome Air Development Center, August 1981.
4. Malcom, J. G., ‘‘BIT False Alarms: An Important Factor in Operational Readiness,’’ *Proceedings of the Reliability and Maintainability Symposium*, 1982, p. 206.
5. Franco, J. R., ‘‘Experiences Gained Using the Navy’s IDSS Weapon System Testability Analyzer,’’ *Proceedings of AUTOTESTCON ’88*, Minneapolis, Minnesota, September 1988.
6. Johnson, F., and C. R. Unkle, ‘‘The System Testability and Maintenance Program (STAMP): A Testability Assessment Tool for Aerospace Systems,’’ *Proceedings of the AIAA/NASA Symposium on the Maintainability of Aerospace Systems*, Anaheim, California, July 1989.
7. Quinlan, J. R., ‘‘Simplifying Decision Trees,’’ *Journal of Man-Machine Studies*, Vol. 27, 1987, pp. 221-234.

8. Mingers, J., ‘‘An Empirical Comparison of Pruning Methods for Decision Tree Induction,’’ *Machine Learning*, Vol. 3, 1989, pp. 227-243.
9. Simpson, W. R., and J. W. Sheppard, ‘‘System Complexity and Integrated Diagnostics,’’ *IEEE Design and Test of Computers*, Vol. 8, No. 3, September 1991, pp. 16-30.
10. Sheppard, J. W., and W. R. Simpson, ‘‘A Mathematical Model for Integrated Diagnostics,’’ *IEEE Design and Test of Computers*, Vol. 8, No. 4, December 1991, pp. 25-38.
11. Simpson, W. R., and J. W., Sheppard, ‘‘System Testability Assessment for Integrated Diagnostics,’’ *IEEE Design and Test of Computers*, Vol. 9, No. 1, March 1992, pp. 40-54.
12. Sheppard, J. W., and W. R. Simpson, ‘‘Applying Testability Analysis for Integrated Diagnostics,’’ *IEEE Design and Test of Computers*, Vol. 9, No. 3, September 1992, pp. 65-78.
13. Simpson, W. R., and J. W. Sheppard, ‘‘Fault Isolation in an Integrated Diagnostics Environment, *IEEE Design and Test of Computers*, Vol. 10, No. 1, March 1993, pp. 52-66.
14. Sheppard, J. W., and W. R. Simpson, ‘‘Performing Effective Fault Isolation in Integrated Diagnostics,’’ to appear *IEEE Design and Test of Computers*, Vol. 10, No. 2, June 1993.

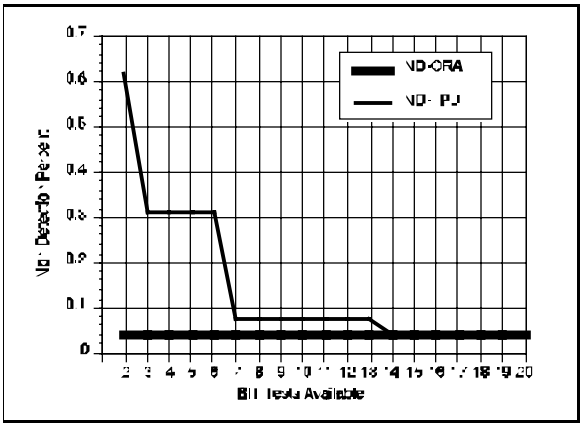

Figure 1. Nondetection Percent for Hypothetical System.

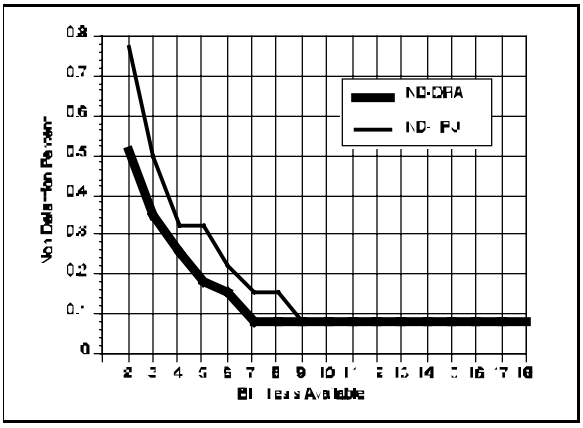

Figure 2. Nondetection Percent for SQS-53 Beamformer.

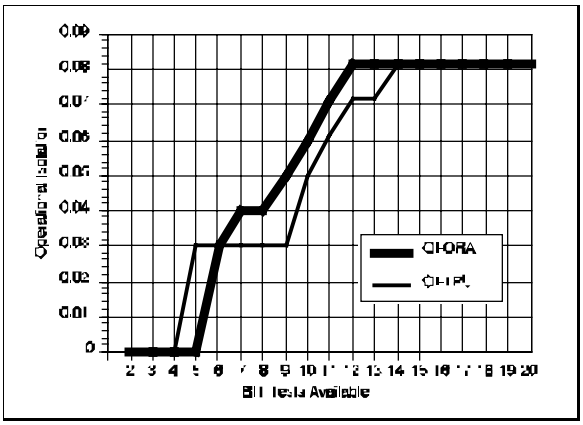

Figure 3. Operational Isolation for Hypothetical System.

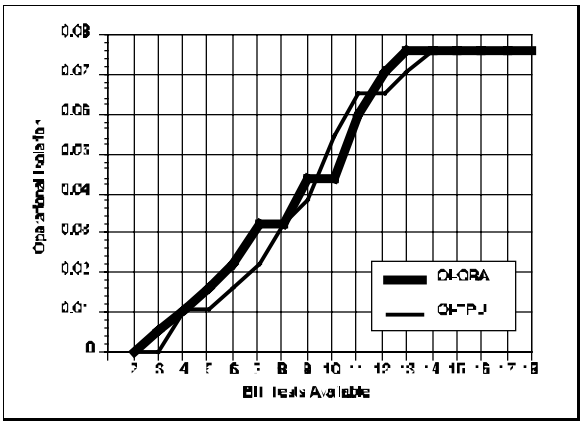

Figure 4. Operational Isolation for SQS-53 Beamformer.

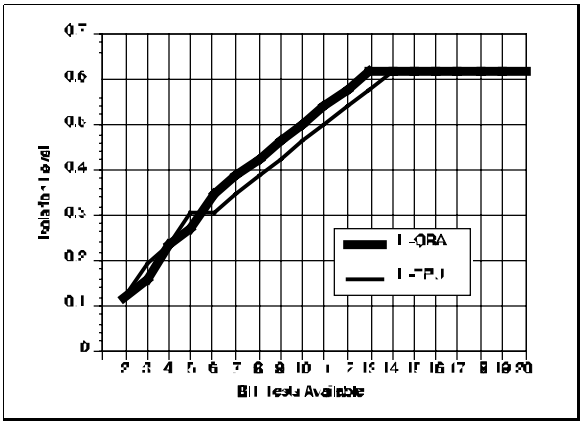

Figure 5. Isolation Level for Hypothetical System.

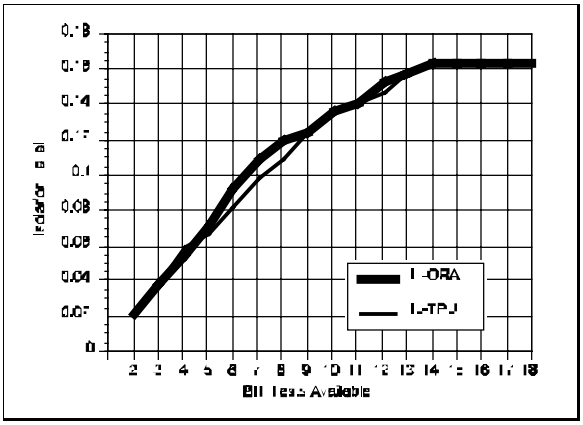

Figure 6. Isolation Level for SQS-53 Beamformer.