AD-A241 621

RL-TR-91-180
Final Technical Report
August 1991

ANALYSIS AND DEMONSTRATION OF DIAGNOSTIC PERFORMANCE IN MODERN ELECTRONIC SYSTEMS

ALPHATECH Inc.

Jerold L. Weiss, James C. Deckert, Kevin B. Kelly (GE-ASD)

DTIC
ELECTE
OCT 08 1991
S D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

91-12544

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-91-180 has been reviewed and is approved for publication.

APPROVED: Roy F. Stratton

ROY F. STRATTON
Project Engineer

FOR THE COMMANDER: John J. Bart

JOHN J. BART
Technical Director of Reliability & Compatibility

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (ERSR) Griffiss AFB, NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Final Sep 88 - May 90

4. TITLE AND SUBTITLE
ANALYSIS AND DEMONSTRATION OF DIAGNOSTIC PERFORMANCE IN MODERN ELECTRONIC SYSTEMS

5. FUNDING NUMBERS
C - F30602-88-C-0068
PE - 62702F
PR - 2338
TA - 02
WU - 3P

6. AUTHOR(S)
Jerold L. Weiss, James C. Deckert,
Kevin B. Kelly (GE-ASD)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ALPHATECH Inc.
50 Mall Road
Burlington MA 01803-4537

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-494

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (ERSR)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-91-180

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Roy F. Stratton/ERSR/(315) 330-4205

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report presents methods and procedures for validation and demonstration of diagnostic system performance for use as part of the Government's acceptance test procedures for electronic systems having contractual diagnostic requirements. These methods provide for examining the adequacy of the diagnostics architecture, establishing bounds on fraction of faults detected (FFD) and fraction of faults isolated (FFI) (the only two figures of merit considered here), and choosing faults to be inserted in the demonstration, but fall short of confirming exact values for FFD and FFI. The current state-of-the-art does not support an exact determination of FFD and FFI for new equipment, especially that containing higher density VLSI and VHSIC electronic devices, because of the inability to determine the failure rates for the various failure modes, and the difficulty in inserting a truly random sample of faults.

The effort also reports data on failure modes for a selection of modern electronic components (Table 4-2).

Pages 1-15 discuss the contents of the various sections of the report.

NOTE: Rome Laboratory/RL (formerly Rome Air Development Center/RADC)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Diagnostics, Acceptance Testing, Validation, Demonstrations	212
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39.18
298-102

EXECUTIVE SUMMARY

The objective of this effort is the development of methodologies and procedures for validation and demonstration of the performance of diagnostic systems associated with electronic circuit boards containing VLSI/VHSIC devices, and line-replaceable units consisting of many such boards. All types of diagnostic capabilities are considered. This includes on-equipment built-in test (including monitoring, initiated, power-up test, etc.), organizational-level test systems (such as portable maintenance aids), and test programs for depot-level automatic test equipment. These procedures are intended for use as part of the Government's acceptance test procedures for electronic systems having contractual diagnostic requirements; the diagnostic capability may be either internal or external.

For this effort we are interested in two performance measures (or figures of merit): fraction of faults detected (FFD) and fraction of faults isolated (FFI), where FFD refers to the fraction of total faults, which occur over the *useful lifetime* of the system, that are *detected* by a specified *means*, and FFI refers to the fraction of lifetime faults that are correctly *isolated* to ambiguity groups of specified *size* by specified *means*. Diagnostic system characterization for useful lifetime faults is chosen not only because of the absence of a logical alternative, but also because such real-world behavior affects a myriad of operational and logistics performance measures, such as readiness level, usage rate of spare parts, average down time of the equipment, unnecessary down time due to false alarms, mean time between depot maintenance actions due to lack of organization-level fault isolation, etc.

Relating FFD and FFI to useful lifetime faults dictates that fielded-equipment failure rates must be considered. As we determined through an extensive literature survey as part of this effort, the accurate characterization of physical faults over a system's useful lifetime presents difficulties for modern electronic systems. A single VLSI chip contains of the order of 100,000 transistors, each of which may have any of a number of physical failure modes (e.g., shorts, opens, new (parasitic) device, degraded device) depending upon the layout of the circuit and the size of the physical defect. The functional manifestation of these physical faults includes the ubiquitous stuck-at condition (which, when occurring on internal nodes, causes incorrect behavior for only some input stimuli), high- and low-impedance states, increased circuit response delays, additional system states, and others. The accuracy with which FFD and FFI can be predicted by either

analysis or demonstration depends upon the ability to determine the relative frequency with which the various faults can/will occur. Unfortunately, empirical data and prediction standards such as MIL-STD-217E provide failure rates only for devices (and higher levels of indenture) as a whole. Likewise, insufficient empirical data exist for the relatively few systems containing VLSI/VHSIC components fielded today to provide precise failure rate information.

As a result of our study, we feel that the combined effects of the sheer number of potential failure modes and the uncertainties associated with the lifetime rates for those modes preclude, in general, the *reliable, accurate* validation of the diagnostic capability of a system composed of boards containing VLSI/VHSIC components. This is especially true if validation is desired through only a live demonstration involving the insertion of a series of (physical or software) faults, since a significant number of failure modes are precluded from insertion due to topological and other practical constraints, and the total number of insertions is constrained by the limited demonstration time available.

Because we determined that reliable, accurate testability validation cannot be accomplished for modern electronic equipment by live demonstration alone, we have developed an alternative two-phase methodology in the spirit of MIL-STD-2165. The first phase consists of a diagnostic test-effectiveness analysis (supported by simulation and engineering judgement) that is used to predict performance, in the most precise quantitative manner possible, as a means to uncover any diagnostic system design deficiencies, determine critical assumptions, etc. Our set-theoretic approach to test effectiveness analysis does not require the (often erroneous) assumption of diagnostic test outcome independence. The second phase consists of a demonstration, in a factory environment, that verifies that the functional requirements, consistent with the assumptions and conclusions of the test effectiveness analysis, have been met. In essence, this two-phase approach employs as many fault insertion demonstrations as are practical, and uses simulation and engineering judgement as backup means for providing the Government (during the analysis phase) with quantitative assurances of the quality of the diagnostic capability being procured. It should be emphasized, however, that we still must rely on engineering judgement in several areas, including the definition of fault modes for a given component, the definition of diagnostic coverage figures for those component modes, the assignment of portions of the overall component failure rate (computed using MIL-HDBK-217 or similar means, or obtained directly through use of increasingly common manufacturer-warranted figures) to those fault modes, and the bounds or confidence levels for these quantities. Page 1-15 discusses the contents of the report by chapters.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

A	address (on PDIU board drawings)
AD	address and data (on PDIU board drawings)
A/D	analog-to-digital
ASIC	application-specific integrated circuit
ATE	automatic test equipment
BIT	built-in test
BITE	built-in test equipment
CAD	computer-aided design
CID	Commander's Integrated Display for the Army M1A2 tank
CM	continuous monitoring tests of the PDIU
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
D-level	depot-level
DC	direct current
DCU	display and control unit of the PDIU
DP	demonstration plan
DTACK	data acknowledgement
DVM	digital volt meter
EEPROM	electrically erasable PROM
EOS	electrical overstress
EPROM	erasable PROM
ESD	electrostatic discharge
FFD	fraction of faults detected
FFI	fraction of faults isolated
FMEA	failure mode and effects analysis
FMECA	failure mode effects and criticality analysis
FPMH	faults per million hours
GaAs	gallium arsenide
GE-ASD	General Electric Automated Systems Department
glue logic	AND/OR/NOR gates (on PDIU board drawings)
IC	integrated circuit

ICE	in-circuit emulator
RU	replaceable unit (line or shop)
MOS	metal oxide semiconductor
MUX	multiplexer
O-level	organization-level
OD	on-demand/on-event tests of the PDIU
PAL	programmable array logic
PCT	programmable counter/timer
PDIU	Prognostic Diagnostic Interface Unit, a modern Army electronic system
PLA	programmable logic array
PMA	portable maintenance aid
PROM	programmable read-only memory
PUCT	power-up confidence test of the PDIU
RAC	Reliability Analysis Center, PO Box 4700, Rome, NY, 13440
RADC	Rome Air Development Center (an Air Force laboratory), renamed Rome Laboratory
RAM	random access memory
RC	resistor-capacitor
ROM	read-only memory
R/R	remove/replace
RTE	run-time environment
RTI	remote terminal interface
RTOK	re-test OK
RU	replaceable unit (line or shop)
s/a	stuck-at
SP	signal protection
SRAM	static RAM
STE/ICE	Simplified Test Equipment/Internal Combustion Engines, a modern Army electronic system
STE/ICE-R	STE/ICE-Reprogrammable, a modern Army electronic system
TPS	test program set
TQM	total quality management
TTL	transistor-transistor logic
UUT	unit under test
V	volt
Vcc	supply voltage (+5V for the PDIU)
VHSIC	very high-speed integrated circuit

VLSI	very large-scale integration
Xceiver	transceiver (on PDIU board drawings)

SECTION 1

INTRODUCTION

1.1 OVERVIEW

The objective of this effort is the development of methodologies and procedures for validation and demonstration of the performance of the diagnostic systems associated with electronic circuit boards containing VLSI/VHSIC devices, and replaceable units (RUs) consisting of many such boards. All types of diagnostic capabilities are considered. This includes on-equipment built-in test (BIT) (including monitoring, initiated, power-up test, etc.), organizational-level (O-level) test systems (such as portable maintenance aids (PMAs)), and test programs for depot-level (D-level) automatic test equipment (ATE). These procedures are intended for use as part of the Government's acceptance test procedures for electronic systems having contractual diagnostic requirements; the diagnostic capability may be either internal or external.

Before proceeding further, it is important that we clarify our use of the words verification and validation in this report (Hoover and Perry, 1989). *Verification* of a diagnostic capability is the determination that the diagnostic capability *as built* properly implements the diagnostic capability *as designed* (i.e., "did we build the product right?"). *Validation*, on the other hand, is the determination that the diagnostic capability *as designed* satisfies the *stipulated diagnostic requirements*, such as $x \pm y$ percent FFD with z percent confidence (i.e., "did we build the right product?").

For this effort, two performance measures (or figures of merit) are of interest: fraction of faults detected (FFD) and fraction of faults isolated (FFI). Herein, FFD refers to the fraction of total faults, which occur over the *useful lifetime* of the system, that are *detected* by a specified *means*. The fault universe for FFD may be all faults, including built-in test equipment (BITE) failures, or it may be only primary equipment failures, only mission-critical failures, or any other

specific subset of failures. FFI refers to the fraction of lifetime faults that are correctly *isolated* to ambiguity groups of specified *size* by specified *means*. FFI may also refer to any specified fault universe (most often seen is the set of detected faults).

Diagnostic system characterization for useful lifetime faults is chosen not only because of the absence of a logical alternative, but also because such real-world behavior affects a myriad of operational and logistics performance measures, such as readiness level, usage rate of spare parts, average down time of the equipment, unnecessary down time due to false alarms, mean time between depot maintenance actions due to lack of organization-level fault isolation, etc.)

Both measures require specification of a means of detection or isolation. This is important since it defines the diagnostic outcome to be verified. For example, setting a fault-status bit in a specific RAM location may be one means of isolation for a power-up test. The performance of this means of isolation, however, is not the same as the display of a fault-detect message on a PMA. This is because some faults may correctly set the fault-status bit, but may also prevent its display. (Note–such a detect-but-not-report error occurred in the demonstration discussed in subsection A.3). For FFI, the ambiguity group size of interest must also be specified. Faults that are either incorrectly isolated (isolated to an ambiguity group not containing the fault) or correctly isolated to groups of larger than the specified size are not included in FFI. Usually ambiguity group size is specified as one replaceable unit (e.g., a board at O-level or a device at D-level).

Because both FFD and FFI refer to useful lifetime faults, fielded-equipment failure rates must be considered. The accurate characterization of physical faults over a system's useful lifetime presents difficulties for modern electronic systems, in which a single VLSI chip contains of the order of 100,000 transistors, each of which may have any of a number of physical failure modes (e.g., shorts, opens, new (parasitic) device, degraded device) depending upon the layout of the circuit and the size of the physical defect (Maly, 1987). The functional manifestation of these physical faults includes the ubiquitous stuck-at (s/a) condition (which, when occurring on internal nodes, causes incorrect behavior for only some input stimuli), high- and low-impedance states, increased circuit response delays, additional system states, and others (Abraham and Fuchs,

1986). The accuracy with which FFD and FFI can be predicted by either analysis or demonstration depends upon the ability to determine the relative frequency with which the various faults can/will occur. Unfortunately, empirical data and prediction standards such as MIL-HDBK-217 provide failure rates only for devices (and higher levels of indenture) as a whole. Likewise, insufficient empirical data exist for the relatively few systems containing VLSI/VHSIC components fielded today to provide precise failure rate information.

As the above discussion suggests, we feel that the combined effects of the sheer number of potential failure modes and the uncertainties associated with the lifetime rates for those modes preclude, in general, the *reliable*, *accurate* validation of the diagnostic capability of a system composed of boards containing VLSI/VHSIC components. This is especially true if validation is desired through only a live demonstration involving the insertion of a series of (physical or software) faults, since a significant number of failure modes are precluded from insertion due to topological and other practical constraints, and the total number of insertions is constrained by the limited demonstration time available.

Because reliable, accurate testability validation cannot be accomplished for modern electronic equipment by live demonstration alone, we have developed a two-phase methodology in the spirit of MIL-STD-2165. The first phase consists of a diagnostic test-effectiveness analysis (supported by simulation and engineering judgement) that is used to predict performance, in the most precise quantitative manner possible, as a means to uncover any diagnostic system design deficiencies, determine critical assumptions, etc. Our set-theoretic approach to test effectiveness analysis does not require the (often erroneous) assumption of diagnostic test outcome independence. The second phase consists of a demonstration, in a factory environment, that verifies that the functional requirements, consistent with the assumptions and conclusions of the test effectiveness analysis, have been met. In essence, this two-phased approach employs as many fault insertion demonstrations as are practical, and uses simulation and engineering judgement as backup means for providing the Government (during the analysis phase) with quantitative assurances of the quality of the diagnostic capability being procured. It should be emphasized, however, that we

must rely on engineering judgement in several areas, including the definition of fault modes for a given component, the definition of diagnostic coverage figures for those component modes, the assignment of portions of the overall component failure rate (computed using MIL-HDBK-217 or similar means, or obtained directly through use of increasingly common manufacturer-warranted figures) to those fault modes, and the bounds or confidence levels for these quantities.

1.2 PROBLEM DEFINITION

In order to understand the diagnostic validation and demonstration problem more thoroughly, it is useful to define FFD and FFI more precisely. Formally, FFD and FFI are defined by

$$FFD = \frac{\sum_{i=1}^{N} a_i \lambda_i}{\lambda_T}$$

$$FFI = \frac{\sum_{i=1}^{N} b_i \lambda_i}{\lambda_T}$$

where a_i is equal to 1 if fault i is detected by the specified means and zero otherwise, b_i equals 1 if fault i is correctly isolated to an ambiguity group of the specified size by the specified means and zero otherwise, N is the total number of faults being considered, λ_i is the failure rate of fault i, and λ_T is the failure rate of the entire system. (Note that λ_T is not necessarily equal to the sum of all λ_is since we usually cannot enumerate all of the faults in the fault universe).

To validate whether or not a diagnostic capability meets desired levels of FFD and FFI, the quantities λ_i, λ_T, a_i, and b_i must be known or estimated. Under *ideal* circumstances, all of the possible faults of interest can be listed and their failure rates, λ_i, determined (say through use of MIL-HDBK-217 plus engineering judgement). The total failure rate, λ_T, can be determined by adding the λ_is (since, ideally, complete enumeration of faults is possible). Then a_i and b_i can be

determined by inserting each of the faults in the list into the system and observing if they are detected and correctly isolated by the specified means.

Obviously, there are many practical problems associated with using this ideal method for modern electronic systems containing VLSI/VHSIC devices. A brief discussion of these problems is given below.

Problem 1: Too Many Faults

Even in moderately complex systems, the number of potential faults can be very, very large. To insert each fault for demonstration purposes is too time consuming to be practical. This problem is adequately addressed by the concept of random sampling and appears in MIL-STD-417A and MIL-STD-2077. Of course, the accuracy of the knowledge of the relative frequencies of the dominant fault modes directly impacts the reliability of the sampling results.

Problem 2: Inaccessible Faults

As devices get more complex and more dense, a greater percentage of fault mechanisms occur *within* devices and are not strictly accessible for demonstration purposes (i.e., one can not insert a stuck-at fault on a node within a VLSI device without the addition of fault-injection circuitry or special software). As a result, the assumption of many verification methods that faults can be selected randomly (according to failure rate or not) is violated.

The obvious alternative is to determine how each internal fault mechanism manifests itself at the device output and then insert the fault "effect" at the device boundary, if possible. As discussed in subsection 2.2, this concept is rather limited for a variety of reasons.

The physical insertion of a fault's effect at the device boundary is also not without its limitations, such as:

- lack of physical access (e.g., component not socketed),
- potential damage to other components, resulting in the insertion of multiple (unknown) faults, and
- requirement for special support equipment that may not be available.

Problem 3: Inaccurate Failure Rates

Failure rate prediction is a lengthy subject unto itself (see, for example, USDoD (1986), USAF (1988), Coit et al. (1984), Rossi (1986), Wong et al. (1987), Zins and Smith (1987), and Stockman and Rash (1985)) and will not be discussed in detail here. In general, there are several reasons for possible inaccuracies. First, only recently have prediction methods been developed for VLSI/VHSIC technology (Denson and Brusius, 1989). Since there are relatively few data on field failures of such devices, verification of these methods is not yet possible. Secondly, such prediction methods provide failure rates for *devices*, not the failure *mechanisms* (or *effects*) that must be considered for validation of diagnostic capabilities. Finally, even predicted failure rates of devices may be inaccurate with respect to field failures since these rates are sometimes governed by design characteristics that are not captured by MIL-HDBK-217 (e.g., use of the wrong connector in environmentally-stressing applications). This inherent, irreducible inaccuracy of failure rate estimates is a fact of life that must be recognized, and accommodated, in any practical, useful methodology.

Problem 4: Unknown Fault Modes

While the single stuck-at logical fault model has been widely used for developing test patterns and assessing fault coverage for digital devices, field experience (e.g., see Meth and Musson (1983)) suggests that this model does not come close to characterizing the faults seen in the field. From physical analyses such as Galiay et al. (1980) and Abraham and Fuchs (1986), it is clear that the single stuck-at fault does not cover all possible fault mechanisms. For example, multiple faults are not uncommon, high/low impedance states may occur, circuit delays can be induced, and even extra states can be introduced (i.e., a combinational logic circuit can be turned into a sequential circuit). Knowing which of these effects are the important ones to consider (i.e., which have the highest failure rates) is virtually impossible to predict before the system is fielded. This is true even if detailed fault mode information were known for component elements in other applications since actual fault behavior is very dependent on the specific topology of the subcomponents relative to heat sources, magnetic sources, and the like.

Problem 5: Internal Diagnostic Capabilities

One of the unique features of VHSIC technology is the incorporation of BIT functions within the chip itself. These functions include both monitoring BIT functions and externally initiated BIT. For such devices, it no longer makes sense to talk about inserting faults or emulating faulty device behavior for demonstration purposes. For verification of systems containing such devices, the main questions are: can the system in use properly interpret the results of on-chip BIT, what are the performance characteristics of the on-chip BIT, and how do the BIT performance measures relate to the overall diagnostic system performance measures? Of course, vendor- or Government-supplied on-chip BIT performance characteristics may be used when available.

1.3 SUMMARY OF RESULTS

1.3.1 Overview of the Effort

This effort consisted of four major tasks:

1. Survey and Analyze Current Methods,
2. Develop New Methods (as required),
3. Develop Practical Procedures to Apply Selected Methods, and
4. Demonstration.

The methods being surveyed and developed in Task 2 included:

a. Demonstration, Validation, and Verification Methods,

b. Methods of Fault Characterization,

c. Fault Simulation Methods, and

d. Fault Insertion Methods

Demonstration, validation, and verification methods (a) refer to *practical* procedures for accomplishing testability verification. These procedures may include specific methods of fault characterization, simulation and/or insertion or they may leave open the specific selection of methods. The demonstration task (Task 4) was intended to illustrate the resulting approach by applying it to an existing system. The following is a summary of the results of these tasks.

1.3.2 Summary

TASK 1 SURVEY AND ANALYZE CURRENT AND PROPOSED METHODS

An extensive database of open literature, DoD, Air Force, and industrial sources was developed. The database includes references to, and summaries of, over 130 sources in the following areas:

- Demonstration/Validation Techniques
- Fault Insertion Methods
- Fault Simulation and Fault Grading (Analysis) Methods
- Fault Characterization

Section 2 provides an analysis of this data. Statistical demonstration procedures (e.g., MIL-STD-471A and 2077) are prevalent in current acceptance procedures. Our study has concluded, however, that such procedures are misapplied to the testability validation problem. This is because: a) failure rates are not known precisely (including the fact that not all fault modes are known), and b) selection of faults for demonstration cannot be done completely randomly since many types of faults cannot be physically inserted (large classes of typical faults can not be selected). The effect of (a) is direct. Large errors in failure rate prediction (which frequently occur) result in an incorrect sampling distribution and hence an incorrect prediction of diagnostic performance (FFD or FFI) from sampled fault insertions. The effect of (b) is indirect, but has a potentially serious effect. Since large classes of faults can not be practically inserted, the sample fault insertions will omit these faults entirely. While each individual fault that is not sampled may have a small probability of occurrence, it is quite possible that the sum of the non-insertable faults dominate the failure rate of the equipment. Thus, even if insertion of all insertable faults were accomplished and the diagnostic capability correctly responded in each cases (100 percent correct insertions), there may still be a majority of faults to which the diagnostic capability responds incorrectly. Furthermore, the argument that there is some correlation between the detectability (or isolability) of insertable faults and non-insertable faults (and, hence, insertions give information about detection/isolation of noninsertable faults) has not been adequately established. (One can, in

fact, easily devise a diagnostic system that has 100 percent coverage of insertable faults and less than 50 percent coverage of actual faults by assuming that only stuck-at faults at a device output are insertable and using a limited number of test patterns.)

Some current standards do not rely on statistics-based decisions for acceptance or recognize the limitations of such an approach, and incorporate other tasks that provide the Government with more assurance that a diagnostic capability will work well. The US Army TPS Procedures Manual, for example, suggests an acceptance procedure that effectively provides for more Government involvement in the TPS integration process. It requires correct responses of the diagnostic capability for essentially all fault insertions. Any failure results in a rework and a penalty insertion (i.e., an additional insertion is imposed on the contractor as a penalty). In the US Air Force MATE TPS Verification Guidelines, acceptance requires not only passing statistical insertion success criteria, but also acceptance of a TPS integration log book. This log contains the results of fault insertions and the rework that was done to correct any diagnostic errors. The Army procedures do not require failure rate analysis and are therefore unable to provide any assurances that the important failure modes (those that occur frequently) are addressed in the acceptance test. Failure rates, obtained through MIL-HDBK-217 and an FMEA, are applied in the statistical acceptance criteria in the MATE TPS Verification Guidelines.

MIL-STD-2165 provides another combination of statistical procedures with other tasks for testability verification. MIL-STD-2165 relies primarily on MIL-STD-471A procedures for acceptance demonstrations, but recognizes the limitations of this and suggests that additional demonstrations be done to validate the assumptions made in a previously performed test-effectiveness analysis.

Finally, none of the procedures provide any guidance on product maturation and support. Such activities are becoming increasingly important (General Dynamics, 1987) especially in light of the Total Quality Management Approach that is currently being institutionalized by DoD.

TASK 2 IDENTIFY AND DEVELOP NEW METHODS, INCLUDING FAULT SIMULATION AND INSERTION

Overview

As discussed above and in Sections 2 and 3, accurate, reliable demonstration, in a factory environment, that specific FFD and FFI requirements have been met (i.e., diagnostic performance validation) is not generally possible for modern digital electronic equipment. That is, predicted values of FFD and FFI that are obtained during fault insertion testing may bear little resemblance to the actual values of these quantities when the diagnostic capability is fielded and tested over the operational lifetimes of many devices. In general, this is because at the time of Government acceptance (to which this effort applies) it is not possible to define accurately the expected fielded-system faults and their rates, and because many relevant faults cannot be practically inserted. This fact has led to our development of a two-stage approach in the spirit of MIL-STD-2165, in which: 1) a demonstration is used to *verify* that functional design requirements have been met (using fault insertion and emulation techniques) and 2) a diagnostic test-effectiveness analysis is used to predict performance and discover any design deficiencies. The test-effectiveness analysis can be used subsequently, if desired, in the development of a maturation plan for the diagnostic capability. The maturation process is an ongoing one that conceptually begins with the test-effectiveness analysis, and whose purpose is, in part, to *validate* that the diagnostic performance requirements (specified levels and confidences for FFD and FFI) are being (or will be) met. In essence, this two-phase approach employs as many fault insertion demonstrations as are practical (in terms of time, support equipment required, and risk of damage to the demonstration unit) and uses simulation and engineering analysis and judgement as backup means for providing the Government with assurances of the quality of the diagnostic capability being procured. This approach is discussed in Section 3, with details on the analysis methods given in Section 8. These analysis methods lend themselves to both direct and indirect analysis, the latter being where failure effects are linked from one level of indenture to another. It is envisioned that these methods be employed by starting at the device level, where simulations are normally used to determine fault coverage of test patterns.

Simulation and Insertion

There are many considerations that govern the ability to perform fault insertion demonstrations. We have developed several guidelines to aid in the process of defining a meaningful fault insertion demonstration. These are discussed in Section 6. The more important guidelines are summarized below:

- Use an engineering prototype with socketed chips if possible. This will permit lead-clipping fault insertions without damaging a production unit, and facilitates a variety of other insertion procedures as well.
- Shorting of input and output pins (to each other and to supply voltages and ground) can be used to simulate stuck-at faults. This must only be done on devices that can handle the resulting currents without catastrophic failure.
- A break-out box is a hardware device, which may or may not be remotely programmable, that creates the actual signals that would be produced by a faulty piece of equipment. Usually, break-out boxes are limited to board-level fault emulation or higher. The use of a break-out box for chips is technically feasible, and has even been employed for testability verification in Hummel (1988).
- Fault emulation (software fault insertion) should be employed when no other means of demonstrating a particular fault class can be feasibly obtained. This method usually entails modification of test software elements to force certain tests to fail. For example, an incorrect checksum can be stored in a PROM to force a test outcome that results from PROM failures to occur. Fault emulation, therefore, provides a means of ensuring that certain functional requirements of the diagnostic capability have been met and that the correct diagnostic outcome is obtained when specific test outcomes are created. Fault emulation is analogous to forced branching of a TPS during TPS integration.

We recommend that simulations be employed only as part of the test-effectiveness analysis task, if possible, and not as part of the demonstration procedure. In using simulations for analysis, it must be recognized that a fault simulation can be inaccurate for a variety of reasons (including the fact that it may ignore important classes of faults). Simulations should be used only for analysis, and not for demonstrations, because experimental validation of a simulation's fidelity (in terms of FFD and FFI prediction) in a factory environment is not practical. This fact is due, again, to the limitations of fault insertion to validate the simulation.

More detail on fault simulation and fault insertion techniques is provided in Sections 5 and 6, respectively.

Field Failure Characterization

The objective of one subtask, discussed in Section 4, was to determine if the manifestations of typical field failures could be characterized at a high enough level (generally at the device, board, or unit output) so that simple simulation processes or simple insertion methods could be used in testability verification. Unfortunately, while much data have been collected about failure rates of devices, boards, and units, there are few relevant data available that describe the typical fault manifestations (at the device, board, or unit level) that appear in field testing situations. The data available in the literature tend to be inconsistent from one program or report to the next and are therefore useful only to point out the types of failure modes that occur. No detailed analyses as to the average number of pins affected, or other high-level characterizations, are available in the literature.

In general, detailed post-mortem analyses of failed parts are not performed. The parts are either discarded or, for particularly high failure rate components, they are often returned to the manufacturer, who then will perform analyses to determine the cause of the failure and the corrective actions needed. As one might suspect in a competitive industry, information gleaned from these analyses tends to be proprietary. In this investigation, we contacted the Reliability Analysis Center (RAC), Raytheon, and various divisions of General Electric. No information could be found at Raytheon, although if the desired information had been collected, we were told that it would likely be considered proprietary. Fortunately, as we originally proposed, we were able to examine maintenance data for the Simplified Test Equipment-Internal Combustion Engines (STE-ICE) and the STE/ICE-Reprogrammable (STE/ICE-R), portable diagnostic and prognostic tools built by GE-ASD for the Army (see subsection 4.3). Because GE-ASD is the depot for STE/ICE and STE/ICE-R, we feel these data are representative of the best maintenance information available in general on fielded military electronic equipment. These data covered the top ten component failures for two periods: 1989, and the first two months of 1990. The faults were classified into ten high-level modes. For those components whose failure rates were significantly higher than predicted (via MIL-HDBK-217E parts count method):

- a larger than expected number of 8K x 8 EE PROMs failed, with the cause traced to bad workmanship, fault mode unknown,
- the vast majority of IC failure modes are high/low value or gain, or stuck high/low, and
- intermittent faults appear to be the dominant failure mode for switches.

The RAC compiles a database of device faults. The database breaks down device faults into very broad classes. For some of these classes, typical failure manifestations can be postulated, although no data are available for determining if such manifestations actually occur. For VLSI devices, for example, most failures result in "improper output." This suggests internal faults (since the other classes in the RAC report are shorts and opens on inputs and outputs, and output latching). Internal faults generally result in incorrect input-output mappings for subsets of device excitations. There is no way to narrow the failure effects any further.

A short VLSI-device failure mode and effects analysis (FMEA) performed at the RAC provided relative failure frequencies for different failure modes of different device classes (Stockman and Rash, 1985). The modes included opens, stuck-ats, and excessive current on input and output pins (because these are the modes that could be observed). It was indicated, however, that these modes accounted for no more than about 80 percent of the total fault universe for the devices. No data are available on the other 20 percent because these faults are the ones in which the particular fault manifestation could not be identified. From these limited data the conclusion that we can draw is that stuck-at fault simulation/insertion on input and output pins probably account for at least 20 percent of IC failure modes.

Finally, the RAC Field-Failure Return Program provided us with preliminary data on causes of field failures in electronic equipment (Green, 1987 and 1988; and Green and Denson, 1988). Based on analysis of 63 returned units, they found that field failures include fabrication or assembly defects (22 percent), electrostatic discharge (ESD) induced effects (40 percent), re-test OK (RTOK) (20 percent), and non microcircuit (2 percent). Fabrication defects result in too many different fault effects for modeling or insertion by a few simple techniques. ESD induced failures, however, may be frequently modeled by shorts and opens at device inputs and outputs. These

types of failures can be more readily inserted and simulated and, therefore, should be considered more strongly in a validation effort. Section 5 discusses these data in more detail.

We have used this information to help guide our fault insertions/emulations in the example Demonstration Plan given in subsection A.3. Specifically, we perform a significant number of insertions/emulations on EEPROMs, and insert/emulate a significant number of shorts, opens, and stuck-ats. Guidelines for choosing the number of insertions and emulations are given in Section 7.

TASK 3 DEFINE PRACTICAL PROCEDURES TO APPLY SELECTED METHODS.

In Sections 7 and 9 we provide detailed guidelines for the preparation of a demonstration plan and a test-effectiveness analysis report. The demonstration plan provides a recommended approach that includes specifications of how failures will be chosen and who will choose them. The accept/reject criteria are quantitative but not statistically precise, since factory validation of performance is not possible; MIL-STD-471 is therefore not applicable. Section 8 provides procedures that can be used for test-effectiveness analysis even when significant test overlay structures (where several tests exercise the same device) are employed or when significant amounts of test-coverage overlap exist in the diagnostic capability (i.e., the test matrix is "full"). The Appendix contains an illustrative application of these methods to a modern military electronic unit. This serves as an example of how the selected methods can be applied.

TASK 4 DEMONSTRATION

The Prognostic Diagnostic Interface Unit (PDIU), a modern Army electronic unit developed by GE-ASD to perform diagnostics and limited prognostics for the M109E5 Improved Howitzer, was selected as the demonstration unit a modern Army electronic unit developed by GE-ASD to perform diagnostics and limited prognostics for the M109E5 Improved Howitzer. The Appendix gives the details of how our approach was applied to the PDIU's self-test feature. The use of simulations in the test-effectiveness analysis and the hierarchical nature of the analysis could not be demonstrated since simulation models were not available for any of the PDIU ICs or boards, and since the PDIU self-test feature included no elaborate subtesting categories.

1.4 OUTLINE OF THE REMAINDER OF THE REPORT

In Section 2 we present a review of current and proposed methods for testability verification/validation. In Section 3 we present our approach for demonstration and validation of diagnostic capability for modern military electronic systems. We argue that validation of diagnostic performance through factory demonstration alone is not possible. We propose instead two separate procedures: 1) test-effectiveness analysis to predict diagnostic performance, and 2) diagnostic design verification to demonstrate that the diagnostic capability performs as designed through factory fault insertion. In Section 4 we discuss the difficult task of characterizing field failures, and also present some data on actual field failures. To calculate FFD and FFI we need actual failure rates, but these are not and can not be known for a new system. MIL-HDBK-217 and similar methods predict failure rates for devices. Using the device failure rates and engineering judgement we can estimate the failure rate for a given fault class. The data given in Section 4 may assist in such partitioning of device failure rates among the various classes. In Section 5 we discuss fault simulation and conclude that it is most appropriately used to support the test-effectiveness analysis, but that the existence of an appropriate simulation for any given diagnostic capability cannot be generally assumed. In Section 6 we present guidelines for performing fault insertions during a test-effectiveness demonstration. We present an outline for a test-effectiveness demonstration plan in Section 7, and present an illustration of such a plan in subsection A.3. In Section 8 we present two approaches to test-effectiveness analysis, one a probabilistic approach that assumes the independence of test outcomes, and the other a set-theoretic approach in which independence need not be assumed. In Section 9 we present guidelines for the preparation of a test-effectiveness analysis report, describing the application of the techniques of Section 8, and present an illustration of such a report in subsection A.2. We present a summary and suggestions for future work in Section 10. Finally, in the Appendix we illustrate our approach through its application to the self-test feature of the Prognostic Diagnostic Interface Unit (PDIU), a modern Army electronic unit developed by GE-ASD to perform diagnostics and limited prognostics for the M109E5 Improved Howitzer.

SECTION 2

ASSESSMENT OF CURRENT VERIFICATION METHODS

2.1 REVIEW OF ESTABLISHED DEMONSTRATION/VALIDATION PRACTICES

In this subsection we review several existing standards that have been suggested for use in testability verification. While other standards may be appropriate, the ones discussed below represent a good cross-section of tri-service standards for quality assurance and acceptance testing for diagnostic systems.

MIL-STD-471A

This is a DoD standard for verification of general maintainability. It was originally intended for verification and demonstration of maintenance task times. The verification methodology requires a stratification of *failure events* by failure rate. Failure events are then randomly selected according to this stratification, and the corresponding maintenance tasks are executed. The time it takes to perform these tasks is recorded and statistical decision criteria are applied to the results. In 1978 an addendum was provided to extend this procedure to verification of fault isolation and testability attributes. In this addendum, instead of simply selecting fault events and executing the corresponding maintenance task, *actual faults* must be selected and inserted so that the diagnostic outcomes of interest (fault detection, isolation, etc.) can be observed (actually, one need only establish that these outcomes would occur; fault insertion appears to be the most reliable manner to do this). Again success is determined by simple approximate statistical decision criteria that are based on estimates of diagnostic performance obtained from the number of successful fault insertions.

The problem with using this method for verification of modern diagnostic capabilities is the fact that the actual faults in equipment (unlike the fault events which may be used to drive a maintenance task-time verification) can not be randomly sampled. This is because many actual

faults can not be inserted (their insertion is destructive, they occur inside sealed devices, or insertion procedures are not cost-effective (e.g., delays are required)). If whole classes of faults are left out for this reason, the success probability calculations that govern the decision criteria can not possibly be accurate, and acceptance of inadequate diagnostic capabilities (or rejection of adequate systems) may result. Furthermore, the procedure provides the Government with no qualitative assurances that the diagnostic capability works properly. For example, there is no requirement to exercise all aspects of the diagnostic capability (e.g., all branches in a TPS).

MIL-STD-2077

This is a Navy standard for TPS development that includes sections on verification. The method is similar to MIL-STD-471A in that it requires insertion of actual faults (sampled according to failure rate distributions) and observation of the response of the diagnostic capability (in this case a TPS running on some ATE). Thus, the inability to randomly sample due to inaccessibility or other reasons is still a problem for this standard. The major difference in testability verification procedures between this standard and 471A is the way in which the number of faults is selected and the statistical decision criteria that are applied.

US Army TPS Procedures Manual (USA-PMTPS, 1987)

This is a recent Army manual that includes product acceptance test procedures for TPSs. The acceptance of a TPS centers on its proper operation under normal conditions (go-chain demonstration) and under fault-inserted conditions. No statistical criteria are set for acceptable fault insertion performance. Rather, the contractor is responsible for fixing any diagnostic errors that occur during acceptance test. The fault insertions are also divided into announced (selected by the contractor) and unannounced (selected by the Government) faults. Only one diagnostic error is allowed for announced faults. For unannounced faults, a fault insertion penalty is applied for each diagnostic error and the TPS fails acceptance testing if a certain number of fault insertions in a row result in diagnostic errors, whether they are fixed or not. The acceptance test plan must also ensure that all branches of the TPS are demonstrated. Throughout the manual, design documentation,

formal design reviews, and problem reporting and tracking are heavily stressed. This continuous Government involvement provides a higher level of confidence in the TPS than can normally be provided by acceptance testing alone.

Since the procedure does not utilize statistical decision criteria for acceptance, the result is not unduly influenced by the inability to sample randomly or by unknown failure rates as is the case in MIL-STD-471 and 2077. However, the procedure gives no indication whatsoever of the ability of the diagnostic capability to meet its performance goals (FFD or FFI). Also, by not considering failure rates at all, this procedure may inadvertently ignore testing the diagnostic capability against the most important failures.

MIL-STD-2165

This is a DoD standard for "testability programs." While the standard focuses on testability issues in the development of major electronic equipment, the standard can be used as a guide for the procurement of any diagnostic capability. Tasks in this standard span the entire life-cycle of the diagnostic capability including program planning, requirements analysis, design (preliminary and detailed), effectiveness analysis ("testability prediction"), demonstration, and product support. Two tasks within this procedure are relevant to the testability verification effort: Task 203 (Testability Detail Design and Analysis) and Task 301 (Testability Inputs to Maintainability Demonstration). As part of Task 203, the standard requires an analysis of the design in terms of diagnostic performance measures (analysis methods are given in Appendix A of MIL-STD-2165). Task 301 requires a demonstration of the diagnostic capability (during a broader maintainability demonstration if necessary). While the task requirements reference MIL-STDs 471 and 2077 for demonstration planning, the guidelines in Appendix A recognize that such demonstrations are of limited value in predicting diagnostic performance measures. Therefore, a recommendation is made to perform experiments during the demonstration that are capable of validating the assumptions made in Task 203. If necessary, Task 203 should be repeated to get a better prediction of diagnostic performance.

The reliance on the statistical decision criteria of either MIL-STD-471 or 2077 for acceptance has all of the drawbacks discussed in the review of those standards. The guidelines do recognize this and propose a strategy for using the demonstration to get a more accurate prediction of diagnostic performance, upon which an acceptance decision could be made. The standard does not give any guidelines on what types of demonstration experiments should be done to validate the analysis nor does it say how to fold a repeat analysis into the decision criteria. Furthermore, in many situations, preparation of the initial analysis report may be beyond the scope of the effort, and repeating this task is almost never done due to limited project scope. In broad terms, this concept appears reasonable, given sufficient scope in a project. However, when one considers the type of information that would need to be gathered at the demonstration for analysis validation, the concept falls short. The major errors in performing an analysis of diagnostic performance are the uncertainties associated with failure modes and failure rates, and for modern electronics, the uncertainty associated with the effectiveness of individual test techniques with respect to actual component fault modes (e.g., it's hard to quantitatively predict the percentage of memory faults covered by a checkerboard test (Siewiorek and Swarz, 1982)). None of these uncertainties can be resolved very well in a factory demonstration. Nevertheless, the performance of both an analysis task *and* a demonstration task, and the idea of trying to validate the analysis in some way, are all important elements of the testability verification process. In our proposed approach (Section 3), these concepts are central. Both analysis and demonstration tasks are recommended. New analysis methods are derived to accommodate modern electronics and diagnostic capabilities, and validation of the analysis is relegated to the support phase, where the uncertainties of the analysis may be resolved by longer term observation of the diagnostic capability in operation.

MATE Users Guide (USAF, 1985)

This is an Air Force standard that includes procedures for TPS integration (Section 5) and Acceptance Test (Sections 6 and 7) and Effectiveness and Comprehensiveness Analysis (Sections 8 and 9). The TPS integration phase includes testing of the go-chain, software testing of all TPS

branches (forced branching), and fault insertion testing as well as other tests (such as ATE compatibility, timing, etc.). In this phase, the TPS and UUT must be comprehensively exercised. The selection of faults for insertion should account for predicted failure rates, not repeat faults in repetitive circuitry, be as comprehensive as is feasible, include misalignment-type failure modes (i.e., inconsistent parameter adjustment within a unit), and exercise as much of the TPS as possible. Guidelines for selecting the number of fault insertions to be done are given. All steps in the procedure are logged and an analysis of the results is performed At this stage, change proposals may be made as the result of integration, and these proposals will be approved or disapproved, and if necessary executed, prior to acceptance testing.

At acceptance testing (Section 6), fault insertions are selected largely according to failure rate (guidelines are given in Section 7). Random sampling and selection by convenience is discouraged; however, reliance on statistical decision criteria for accept/reject decisions is made. Detailed figures are given that essentially make the calculations needed in MIL-STD-471 and 2077 easier.

The test effectiveness analysis procedure (Section 8) is similar to MIL-STD-2165 in that it simply assesses effectiveness by assuming that no testing errors occur. Failure rates of all failure modes are to be computed. The TPS call-outs are then used to assess figures of merit such as FFD and FFI, which of course assumes that we can know whether or not a particular test will pass or fail for each failure mode specified (reasons why this is not known very well are discussed in Section 8 of this report). This assumption is recognized in the procedure. It is recommended that the analysis be used to evaluated TPS "applicability" (i.e., is the TPS capable of achieving the required diagnostic performance levels) and not to predict performance (i.e., will it actually achieve these levels). Other analyses (Section 9) include assessment of TPS comprehensiveness.

The reliance on statistical decisions for acceptance testing suffers from all of the drawbacks discussed under MIL-STD-471. Again, this type of methodology relies on the assumption that statistical sampling can estimate diagnostic performance characteristics to within some confidence level, and that given a desired confidence level, we can decide if the test results imply satisfactory

performance. Since random sampling is impossible in modern equipment and, to a lesser extent, since failure rates can not be accurately known, the statistical decision criteria can be quite inaccurate. The integration phase guidelines and the performance of a TPS applicability analysis, however, can give the Government substantial assurances that the TPS is constructed properly. As will be seen in Section 3 of this report, our approach brings many of the integration tests to the acceptance test. New analysis methods are provided that are more appropriate to prediction of diagnostic performance. However, the inclusion of a test applicability analysis is also included as an option.

2.2 OTHER SUGGESTED APPROACHES

Many suggestions have been made to handle the drawbacks associated with accurate statistical validation of diagnostic performance. In this subsection, some of these suggestions are reviewed. Some of the suggestions have appeared in the open literature, while others have been developed in the course of this effort. The suggestions include methods to deal with unknown failure rates for individual modes of components and to deal with fault insertion problems.

While these approaches may address one or more of the problems we have discussed, none address all problems simultaneously, nor does any combination of these approaches. This is because each approach takes the view that direct validation of diagnostic performance by factory demonstration is possible. We shall argue in Section 3 that this is not the case, and devise an approach that views testability validation as an ongoing process that is supported by acceptance test demonstrations, but should start during the design phase with test effectiveness analyses and continue throughout maturation of the diagnostic capability.

APPROXIMATION OF FAILURE MODE RATES. This method addresses the problem of devising failure rates for different modes of a component when only a single rate is available for the component itself (e.g., via MIL-STD-217). It suggests that failure mode rates be approximated by dividing the failure rate of the device into equal parts. This implies, however, that accurate device failure rates are known, that we can determine the total number of failure modes of a particular device, and that there are no "bad actors."

FUNCTIONAL DEFINITION OF FAILURE MODES. This method addresses the problem of knowing all failure modes. It suggests that failure be defined as the inability to perform each function of a device or board or RU. There are many drawbacks to this approach. First, and most important, is the fact that for modern digital equipment, FFD (and FFI) is determined by whether or not the test patterns appearing at the inputs of a device (or board, or RU) are extensive enough to cause a physical fault to be manifested as some functional failure. The insertion of functional failures can, by definition, emulate only those physical failures that are in fact manifested. As a result, this approach virtually guarantees that 100 percent of the inserted faults will be detected (or isolated in accordance with the required ambiguity specifications) independent of the true percentage of successful results that would be obtained in the field. Other problems include the inability to determine failure rates accurately for each functional failure mode, the difficulty of determining the various different manifestations of functional failure for each function, and the inability to insert or emulate the failure manifestations at the accessible points in the system.

BEHAVIORAL DEFINITION OF FAILURE MODES. This method addresses the problem of inaccessibility of faults for insertion. It suggests that faults be grouped according to their behavioral effect at accessible points in the system. Fault insertion then amounts to finding ways of emulating this behavior at the accessible points. There are several problems with this method. First, and most important, is the difficulty of determining the behavioral effects of the various failure mechanisms. Such determination is called for in many existing standards through use of an FMEA. However, FMEAs become impractical when circuit size is large, and this is the trend in modern equipment. Development of standard failure effects for standard circuits and devices would help this process in many cases; however, the increasing use of application-specific ICs (ASICs) implies that an FMEA is needed at least to assess the effect of cell faults at the ASIC pin level and higher. Also, standard failure effects for standard device or cell functions may be an elusive goal since failure effects depend more strongly on device topology than does the functional behavior of the device. Also important is the major issue raised in the functional definition of

failures, namely the inability to consider faults that do not cause observable effects *for the set of test pattern inputs that are applied during testing*. In addition, it is not always the case that behavioral effects at accessible nodes can be practically inserted (e.g., the effects of some faults may be delays, which can be difficult to emulate). Also, the failure rates associated with these behavioral-level faults is still difficult to determine (especially considering the fact that there may not be a one-to-one mapping between actual physical faults and the behavioral fault classes). These problems notwithstanding, the insertion or emulation of faults at the accessible points in a circuit remains the best way to demonstrate diagnostic capability (even though formal validation of performance measures such as FFD and FFI cannot be accomplished in this way alone).

SECTION 3

PROPOSED APPROACH TO DEMONSTRATION AND VALIDATION OF DIAGNOSTIC CAPABILITY

3.1 MOTIVATION

From the discussions in subsections 1.3 and 2.2, it can be seen that most of the problems with existing methods revolve around the inability to define accurately and to insert the relevant faults (either fault mechanisms or fault effects) in a system and the inability to determine reliably their rates of occurrence. From extensive literature review and discussions with diagnostic system designers, it has been determined that these difficulties can not be surmounted by any practical means. That is, reliable validation of diagnostic performance can not be accomplished through direct demonstration alone because the key elements (the important faults and their rates) can not be known with great enough accuracy, and because of the inability to perform truly random insertions. Nevertheless, diagnostic demonstrations are a vital tool in the broader process of assuring the quality of diagnostic capabilities. The approach discussed below recognizes the limitations of direct demonstration and attempts to satisfy the broader objective of diagnostic quality assurance through a combination of direct demonstration and qualitative and quantitative design analyses. The view is taken that acceptance testing is the bridge between the design and maturation phases of diagnostic development. As such, it is important to engage in activities that determine that the diagnostic capability was constructed properly and to predict its performance in the most precise quantitative manner possible. This latter process must recognize that reliable lifetime failure rate and failure mode data are in general not available at the time of acceptance testing. Therefore, it concentrates on an analysis of the effectiveness of the diagnostic design using the *data that are available at the time.* Ideally, this analysis can provide insight into potential problem areas, such as critical failure modes relative to the overall diagnostic requirements or

goals, which can be addressed as the electronic equipment matures and more data become available.

The narrow view of the testability validation process, and the objective of this effort, is to ensure that the government procures only equipment with diagnostic capabilities that, with high likelihood, meet their specified performance requirements (FFD and FFI for example). In broader terms, however, the validation process is only a part of a larger diagnostic development and maturation process whose overall objective is to ensure that Government-procured weapon systems have high quality diagnostic capabilities. In recent years, a Total Quality Management (TQM) approach (Lochner, 1989; Gruska and Heaphy, 1989) has evolved to ensure that quality is designed into and continually improved upon throughout the life of a product. The proposed approach discussed in this section follows the TQM principle of continuous improvement. The approach does not attempt to *guarantee* that specific measures of diagnostic performance will be met when the diagnostic capability is fielded (which, as we have discussed, can not be done accurately). Rather, it attempts to:

- *verify* that the diagnostic capability correctly implements its documented functional design specifications (modified, where necessary, by the results of an independent test effectiveness analysis), and
- start the diagnostic performance *validation* process, which can also be used as the first step in the entire system's overall maturation process, if desired.

As discussed in subsection 1.1, we have defined verification and validation in precise terms. The objective of verification is the determination that a diagnostic system's implementation is faithful to the documented functional design specifications (e.g., test A will detect fault mode x of component y in z seconds). The objective of validation, on the other hand, is the determination that the functional design is actually capable of meeting the performance requirements (e.g., FFD= $x \pm y$ %) that have been levied upon the product.

For most systems, including diagnostic systems, *verification* can be accomplished by demonstrating that the product correctly performs each of its intended functions. Additionally, some systems can also be *validated* by demonstration testing (for example, equipment reliability

can be validated by accelerated life-testing, and maintenance task times can be validated by measuring task performance times under various conditions). Modern diagnostic capabilities, however, are not amenable to direct validation by demonstration testing.

As discussed in subsection 1.2, the main problems with validation of diagnostic systems by demonstration testing are: 1) the limited ability to perform fault insertions, 2) the inability to anticipate all failure modes and 3) the inaccuracy of the prediction of failure rates for fielded failure modes. Once a diagnostic capability is fielded, however, actual faults occur with measurable failure rates. If the appropriate performance monitoring, data collection, and feedback activities are established before field operation (or before operational testing) this actual experience can be used to continually validate, reevaluate, and "mature" the diagnostic performance. To establish such activities, one must analyze the system under consideration to determine what data should be collected and what should be done with it. Thus, the first step in a maturation process might be called *validation analysis*. The objectives of such an analysis are a quantitative and qualitative assessment of the diagnostic capability's performance abilities. The quantitative assessment is designed to predict *approximate* diagnostic performance measures (such as FFD and FFI). These predictions may then serve as benchmarks for comparison of actual field performance and may also provide a means for assessing the approximate performance impact of potential design changes. The qualitative assessment is designed to document the detailed diagnostic capabilities and limitations. This type of qualitative assessment should stimulate consideration of potential changes for improved performance, and provide a systematic means for uncovering design flaws that might limit performance.

3.2 OVERVIEW OF APPROACH

The proposed approach to ensuring high-quality diagnostic capabilities follows the above discussion and therefore requires two separate activities: a validation analysis and a verification demonstration. In the following discussion, an overview of the requirements of each of these activities is given.

3.2.1 Requirements for Diagnostic Demonstrations—Verification

The objective of system verification is to demonstrate that the documented functional requirements of the system of interest have been properly implemented. The main functional requirement of all diagnostic capabilities is that a predetermined set of indicators of system operational status (e.g., Go, No-Go, various fault codes for isolation, etc.) be correctly produced under a variety of system malfunction states (including the no-failure state). Thus, to verify a diagnostic capability, we must be able to cause the equipment to behave as it would in the various system operational states and we must demonstrate that the correct diagnostic outcome is produced as a result. Therefore, the verification process is a (physical or software) fault insertion demonstration.

Now, because we are treating the fault insertion demonstration as a verification process, the selection of faults and the percentage of those that are successfully demonstrated are not determined by probabilistic analysis of a random sampling process (as is the case with many current methods). Rather, we first require that fault insertions be conducted so that every diagnostic outcome is produced at least once (e.g., all paths in a TPS must be tested). Secondly, we require that as many as possible of the system malfunction states for which the diagnostic capability was designed be created or emulated in some way. This is not to imply that all faults must be inserted. Faults may be aggregated into classes of similar types, and then at least one of each type considered in the demonstration if possible. However, it is important that particularly troublesome or questionable capabilities emerging from the validation analysis be thoroughly demonstrated. If physical insertion of a fault can be done, it should be done. Otherwise, some means of emulating faulty behavior should be devised.

For example, consider a simple microprocessor-based system with CPU, RAM, and bus components. A power up test might involve testing of each CPU instruction, the ability to read and write to RAM, and the ability of the CPU to communicate over the bus. There are two diagnostic outcomes, PASS and FAIL. To demonstrate performance of the diagnostic capability, we must demonstrate that the PASS indication (whatever it might be) is produced when the system is in a no-

fail state, and that the FAIL indication is produced under a variety of system failure states. Since the power-up test checks the CPU instruction set, failures that result in each of the instructions producing incorrect results should be either inserted or emulated (e.g., an incorrect result could be used in the diagnostic program). Similarly, bus failures should be demonstrated (perhaps by shorting or opening pins on various chips) and memory faults should be demonstrated (perhaps by backdriving memory chip pins if possible). The guideline for determining what types of faults must be inserted or emulated should be the document that describes the types of faults for which the diagnostic capability should correctly respond. If for example, the contractor claims that CPU timing faults can be detected, then such errors should be demonstrated as part of this process (if possible).

Since the point of the verification process is to demonstrate that functional requirements are met, success can only be obtained with 100 percent successful demonstration results. Thus, failure to correctly produce the correct diagnostic outcome under any of the fault insertion or emulation conditions will require ultimately that the diagnostic capability be redesigned so that the correct outcome is obtained. This should occur before the Government accepts the product or, alternatively, a plan for mitigating an error may be established. The decision as to which event takes place depends on the potential impact of the error, which should be approximately determinable by reference to the validation analysis (see subsection 3.2.2). At a minimum, a plan for monitoring performance and establishing the impact of any errors encountered during the demonstration must be established prior to the demonstration.

Finally, the demonstration activities must ensure that the verification process is both efficient and fair. General industry practices such as those in USA-PMTPS (1987) form the basis for the proposed activities (see Section 7 for more detailed guidelines). The basic elements of these practices include:

- Successful demonstration of operation for no failures,
- A list of demonstration faults chosen by the contractor for demonstration,
- A list of demonstration faults from which the Government may choose,
- Eventual rectification of all problems uncovered in the demonstration, and

- A penalty system that requires the contractor to do more demonstration work for poor demonstration results.

The way in which these practices are implemented varies considerably. This is because most procedures are contractor established as part of an acceptance test plan. Thus, the Government need only ensure that all of the basic elements listed above are present in the demonstration plan prepared by the contractor. (The procedures detailed in Section 7 may therefore be considered as a suggested approach that embodies these basic elements).

3.2.2 Requirements for Diagnostic Performance Analyses—Validation

The intent of validation is to determine if the functional design of a system is capable of achieving the desired system-level performance measures. At the acceptance test phase for diagnostic systems, this can only be accomplished in an approximate manner through analysis. This is because the relevant faults and failure rates are unknown at this stage and because random sampling of faults is not possible when many potentially relevant faults are inaccessible. Since acceptance test validation should be the first step in the maturation process, it is important that the analysis not only produce approximate predictions of diagnostic performance, but also provide some insight into the system design so that design flaws may be uncovered and engineering changes for improved diagnostic performance may be found.

In Section 8 we present a fault accountability theory that can be used to satisfy the performance prediction requirement of validation analysis. This theory was constructed so that the process of collecting diagnostic design information to produce these predictions would hopefully yield enough insight into the design to uncover design logic flaws. In Section 9, specific suggestions for a validation analysis document are given.

The proposed validation analysis methodology centers on a fault accountability analysis. In this analysis, the primary system is decomposed into separate "fault classes" and the diagnostic capability is decomposed into individual "tests." Fault classes can be thought of as disjoint sets of physical faults, and they should be constructed from a physical (not functional) decomposition of the system. Fault classes may may or may not include diagnostic hardware depending on the

desired definition of the performance measures. Tests may include individual steps in a test program, operation of individual BITE functions or groups of test program steps and/or BITE hardware operation. Each test has two or more outcomes (usually two: pass or fail). The analysis method is constructed so that more accuracy is obtained when more detailed decomposition of both the primary and diagnostic capability is performed. (Subsection A.2 contains an illustrative fault accountability analysis.)

The information that must be gathered for the analysis is an accounting of the faults that cause the various outcomes of each test (actually all but one outcome must be considered); hence the name fault accountability. As in Siewiorek and Swarz (1982), this accounting can be qualitative or quantitative. A qualitative accounting simply lists the fault types that can cause the various test outcomes to occur. Quantitative accounting is usually based on a single-fault assumption and is an attempt to specify the fraction of faults from each class that could cause each test outcome to occur. Quantitative accountability can usually only be approximated. Good approximations can be obtained if simulations can be used. (For example, a single test for our purposes might consist of application of a large set of test patterns to a combinational logic chip and the comparison of the actual output with stored output results; the set of patterns can be analyzed for fraction of stuck-at (s/a) fault coverage quite readily by a variety of simulation and fault grading (i.e., fault-coverage estimation) techniques (Agrawal and Seth, 1988).) In the case where a test is actually a previous accepted diagnostic capability (e.g., an entire ASIC on-chip BIT), the results of the validation analyses performed for that capability may be used directly. In most situations, however, engineering judgement must be used to estimate quantitative accountability (e.g., the percentage of all faults covered by tests that are designed only for stuck-at fault coverage). *It is the necessity of relying on engineering judgement that makes the analysis approximate.*

The proposed methodology for validation analysis includes both qualitative and quantitative accountability documentation. The quantitative accountability figures will allow calculation of overall FFD and FFI, once the decision logic for the various diagnostic outcomes is specified. If qualitative accountability includes both faults that are "covered" and faults that are not covered by

each test, then it can provide backup for engineering assessments of quantitative accountability. Also, the exercise of determining qualitative accountability at this level of detail should provide diagnostics engineers with insights into potential logical flaws in their design or potential improvements to it. A side benefit of the quantitative analysis is that it provides a database from which fault isolation algorithms can be devised.

From the quantitative accountability information, overall FFD and FFI can be calculated only after specification of the decision logic used to generate each of the various diagnostic outcomes. The decision logic describes the relationship between the test outcomes and the overall diagnostic outcomes. Section 8 provides more details on how the decision logic and quantitative accountability can be used to determine overall FFD and FFI. The analytic results provide bounds on FFD and FFI that account for the possibilities of overlapping test coverage of unknown extent and nondisjoint fault classes with unknown common faults. The tightness of the bounds depends on the depth to which the diagnostic capability and the fault classes are decomposed. In the limit where all tests either cover all faults in a fault class or don't cover any faults in a class (i.e., either 100 percent or 0 percent of faults in a class can cause each test outcome, respectively) and where all fault classes are disjoint, FFD and FFI can be computed precisely (if not accurately due to the approximations of failure rates).

Finally, we note that the fault universe of interest depends on the particular definitions of the measures FFD and FFI. For example, if FFD and FFI for mission critical faults are of interest, then the fault universe must be determined by a fault modes effects and criticality analysis (FMECA).

3.3 SUMMARY

The view has been taken that testability validation is a part of a broader development and maturation process intended to ensure that high-quality diagnostic capabilities are procured by the Government. When the demonstration of the diagnostic capability is conducted, the development process ends and the maturation process starts. The proposed approach is two-part: it consists of a *validation* analysis, which, ideally, is the first phase of the maturation process that follows, and a *verification* demonstration, which is the final phase in the development process. Validation

analysis is intended to give a coarse prediction of diagnostic performance and to provide insight into the design for uncovering logical design flaws, critical assumptions, and potential improvements. Verification is intended to demonstrate that the functional design requirements of a diagnostic capability have been implemented correctly.

It must be understood that some systems must work correctly at acceptance either because their failure is life threatening, or because there can be no changes after acceptance due to lack of funds. In the first case, maturation is limited and in the second case, consideration of maturation is useless. In any case, maturation is no cure for an unacceptable product.

SECTION 4

CHARACTERIZATION OF FIELD-LEVEL FAULTS IN ELECTRONIC EQUIPMENT

In this section, we discuss different ways of characterizing faults in electronic equipment and provide an overview of published literature on the likelihood and modeling of various faults. The focus is on boards containing digital devices. This section is motivated by the following testability verification concerns:

- In order to insert meaningful faults in electronic equipment, we must know the effects of field-level faults at the accessible points in the circuit, since fault insertions can be accomplished only at these locations.
- The testability verification process is most relevant when the failures that are most likely to occur in fielded systems are considered.
- When analyzing a test system for detection and isolation effectiveness, guidelines that tell the analyst the types of failures that are "covered" by different testing schemes are needed. Faults must be aggregated into classes in order to make these guidelines usable.
- Modeling of faults is necessary for determining how to insert faults for testability demonstrations and for simulating faults when analyzing test-system effectiveness.

In the following, we present an overview of recent literature on physical fault mechanisms and their effects. The discussion focus is primarily on circuit boards containing digital VLSI devices and on failures that can occur in the field. We also discuss in subsection 4.3 results of an examination of actual field faults from the STE/ICE and STE/ICE-R, two generations of diagnostic and prognostic O-level tools developed for the Army by GE-ASD.

Field failures consist of: 1) manufacturing failures that are not detected during production screening and hence are present when the equipment is delivered, and 2) failures that are induced during equipment operation. Induced failures, in general, are caused by environmental and electrical stress, equipment wear-out, and human-related problems.

4.1 FAILURES IN ELECTRONIC EQUIPMENT

System faults are composed of faults that cause incorrect functional operation of the system elements (e.g., boards), faults in the interconnections between these elements, and faults in connectors (the interface between a board and the board-interconnection system).

Board-level faults can be broken down into board-interconnect faults, failures of discrete components (resistors, capacitors, etc.), internal chip failures, and connector failures (the interface between a component or a chip and the component-interconnect system).

Circuit boards and systems vary greatly in terms of their component composition and functional capability. As a result, it is not really possible to make meaningful general statements about system or circuit board failure modes.

To see that this is the case consider the study of Air Force avionics in Wong et al. (1987). In this study, integrated circuits *appeared* to contribute significantly (20 percent - 40 percent) to causes of equipment removals while only 1 percent- 2 percent are due to connectors. In contrast, the study in Zins and Smith (1987) determined that IC components were involved in less than 2 percent of all avionics failures, while over 60 percent were due to connectors and cabling. The discrepancies in reports such as these are due to the fact that the characteristics of the boards being studied are drastically different. In more precise terms, neither study accounted for the "independent" variables that can affect the failure distributions. For example, Wong et al. note that magnetics accounted for very few failures because the boards contained very few magnetic components. Thus, it is evident that their results do not extrapolate to boards that are composed of larger portions of magnetic components. The distribution of component types, environmental factors, screening levels, and design issues are all independent variables that can affect the field failure distribution of modern electronics (Stockman and Rash, 1985). Studies that adequately account for such variables could not be located for this effort. (The closest is Rossi (1986), where failure rates of various integrated circuits are tabulated as a function of environmental and screening factors. Failure modes are not considered).

Anecdotal evidence (Anderson, 1989; Zins and Smith, 1987; Wong et al. 1987) suggests that connector faults can be a major cause of high RTOK rates. This is because many diagnostic capabilities do not consider faulty connectors or interconnects as relevant fault modes. As a result, the connector or interconnect fault is correctly detected by most diagnostic capabilities, but incorrectly isolated to a board or chip that is functional. When the board or chip is tested at the next level of maintenance, it is found to be operational, resulting in an RTOK.

4.2 FAILURES IN DIGITAL VLSI DEVICES

Since the use of digital VLSI devices in modern electronic equipment is increasing, it is logical to assume that the majority of system and board failures are either due to digital VLSI device faults or connections and interconnects. Since digital VLSI will play an important role in future electronic systems, this section provides a brief overview of VLSI device failure mechanisms and their effects. This area has been studied extensively and we present only the salient features here.

4.2.1 Physical Fault Mechanisms in VLSI Devices

We consider VLSI devices constructed in bipolar, MOS, and GaAs technologies. Bipolar technology has been used in older VLSI devices. CMOS is presently the most prevalent technology, allowing much greater chip densities. GaAs is now being used to attain higher speeds in VLSI and VHSIC devices.

Recent descriptions of failure mechanisms are given in Mangir (1984), Wadsack (1978), Maly (1987), Ghate (1982), Green (1987 and 1988), and Green and Denson (1988). In Maly (1987), electron micrographs of damaged ICs obtained from fabrication testing showed damaged metal lines due to spot defects (contamination and unexposed and scratched photoresist). Most of these faults result in catastrophic failures that are very likely to be detected during initial screening of a device, and hence, are unlikely to show up in fielded units. *Metal corrosion*, while rare according to Maly, is the only mechanism cited that can not be screened, and thus shows up during

fielded operation. Therefore, metal corrosion may be a likely field fault mechanism. Metal corrosion causes open circuits and increased resistive paths within a VLSI device.

Electromigration is another defect mechanism that can arise during field operation (Ghate, 1982; Ho, 1982). Electromigration is the transport of metal due to an impressed DC electric field. Metal lines on VLSI devices can be shorted due to electromigration. Electromigration can severely limit circuit densities since, for a given migration growth rate, closer lines will be shorted together in less time. Electromigration is affected by current density (high currents and/or thin lines are susceptible) and temperature. New alloy films are being developed that are less likely to experience electromigration. No evidence could be found to determine the relative importance of electromigration-induced faults in fielded IC units.

In Mangir (1984) these and other fault mechanisms are discussed. For fielded devices the important mechanisms (not discussed above) include random changes of state due to exposure to alpha particles and other *ionizing radiation*, shorts between metalizations due to oxide breakdown from *static discharges*, transistor threshold shifts due to *hot electrons*, shorts between diffusion regions due to *charge spreading*, and line coupling and charge loss (memory devices) due to *degradation of insulator quality* from extended use. Temperature is also cited as a cause of VLSI device failures, but specific physical effects on the device are not given. All of the mechanisms presented above result in permanent faults except a portion of faults due to radiation, which result in transient errors (Anderson and Binari, 1983), and some modes that result in "soft memory failures" (Sie et al., 1977).

According to Denson and Brusius (1989), the dominant failure modes in VHSIC devices include dielectric breakdown, mechanisms that result in latch-up effects, electrical overstress and package related problems (functional failures, input or output leakages, wire bond failures, etc.).

The relative importance of VLSI/VHSIC failure mechanisms in fielded units is now being studied in the RAC Field Failure Return Program (Green, 1987 and 1988; Green and Denson, 1987, Stockman and Rash, 1985). A preliminary analysis suggests that about 20 percent of the identifiable faults are due to IC design or fabrication errors (i.e., defects that were present on

delivery, but passed screening tests) and about 40 percent are caused by electrical overstress (EOS). EOS and electrostatic discharge (ESD) faults were evidenced by broken bond wires, and cracked dies and other obvious defects. Note that the percentages given above are for those failures that *could be identified*; no indication of the numbers of failures that couldn't be identified has been given.

Finally, Table 4-1 shows a VLSI fault mode distribution where the percentages are, again, given for those faults that were identifiable. The source data for this table was Stockman and Rash (1985), while the form of the data is due to Denson (1989).

TABLE 4-1. IC FAILURE MODE DISTRIBUTION SUMMARY

MOS DIGITAL	%	INTERFACE	%
Input Open	33	Output stuck low	59
Output Open	33	Input Open	16
Excessive Input Current	10	Output Open	16
Supply Open	10	Supply Open	10
Output Stuck High	7		
Output Stuck Low	7		
BIPOLAR DIGITAL	**%**	**MOS MEMORY**	**%**
Output Stuck High	23	Data Loss	79
Output Stuck Low	22	Dynamic Characteristics	8
Input Open	17	Input Open	6
Output Open	16	Output Open	6
LINEAR	**%**	**BIPOLAR MEMORY**	**%**
Input Offset Voltage (out of spec)	63	Dynamic Characteristics	79
Output Stuck Low	37	Data Loss	21

4.2.2 Models of the Effects of VLSI Device Faults

The above subsection discussed several physical causes of VLSI device faults. In this subsection, we discuss the effects of these faults. The effects of the physical fault mechanisms can be described at various levels. Ultimately, we are interested in describing fault effects at device pins since these are the only observable/accessible points on the chip. However, it is instructive to review models for lower levels in order to understand the basis for the higher-level characterizations. In the remainder of this subsection, we consider fault models at the transistor level and at the gate level. We then discuss fault effects at the chip level for various device types.

In Timoc et al. (1983) and Galiay et al. (1980), transistor-level fault models consist mostly of shorts and opens between connections to a transistor. Both shorts and opens are described generically as increased and decreased resistances between transistor connections. (Note that failures of metal interconnection lines can also be modeled as shorts (between adjacent lines) and opens.) In TTL bipolar transistors, excessive emitter-collector current is also possible. In MOS technology, opens are sometimes incomplete, causing RC coupling to a transistor gate resulting in slower rise times. Also, oxide-affecting faults can cause transistor threshold changes. From an electrical point of view, these transistor faults can result in many different types of behavior as follows.

TTL Bipolar Circuits:

- Shorts between gate output transistors can result in wired-AND behavior (Timoc et al., 1983).
- An open fanout stem or branch results in s/a 1 behavior (Timoc et al., 1983).
- Excessive collector-emitter leakage may result in multiple s/a behavior depending on the gate topology (Timoc et al., 1983), while other leakage faults have indeterminate behavior that can only be detected by changes in the supply current (Malaiya and Su, 1983).

MOS Technology:

- Shorts between outputs of gates may be modeled by any of 4 different types of asymmetrical wired-logic functions depending on the details of the gate topology (i.e., what other gates are connected to the gates that are shorted) (Timoc et al., 1983).
- Incomplete opens can be modeled as a delay (Timoc et al., 1983; Abraham and Fuchs, 1986).
- Threshold changes, open gates, and some short modes can be modeled as stuck-at 1 or 0 faults since they prevent the transistor from being turned on or keep it turned on (Timoc et al., 1983; Galiay et al., 1980). Some s/a faults occur only after some delay from turn-on (Case, 1976).
- Some open transistor failures cause line capacitance that results in sequential behavior (called a stuck-open in (Wadsack, 1978)) (Timoc et al. 1983; Galiay et al., 1980; Agrawal, 1988).
- Breakdown of insulators can result in line coupling (Abraham and Fuchs, 1986), especially at higher circuit densities.

In Cunningham et al. (1987) GaAs technology faults are considered. Fault mechanisms are similar to MOS fault mechanisms. Instead of characterizing the faults at the transistor-level, however, simulations of interconnected transistors making up different gate types are done with SPICE (Nagel, 1981). The SPICE fault models include transistor threshold changes and decreased and increased resistance values between transistor connections. For example, they conclude that inverter faults can be modeled as: a) stuck-at, b) asymmetric and symmetric delays (rise times and fall times may be different), or c) indeterminate voltage levels. They do not consider how these fault modes may change when gates are connected to each other.

At the gate level, the most common failure mode is the stuck-at zero or one fault model. As its name implies, a stuck-at fault consists of a gate input or output being permanently set to logic voltage levels of zero or one. In test analysis, the single stuck-at fault model is frequently used. However, it is widely known that the single stuck-at fault model falls short of characterizing all faults (Mangir, 1984; Abraham and Fuchs, 1986; Galiay et al. 1980; Maly, 1987; Cunningham et al. 1987; Agrawal, 1988; Breuer et al. 1983a; Fujiwara , 1985). Furthermore, not all s/a faults at the gate level can physically occur (Maly, 1987). If one includes multiple stuck-at faults many more physical faults are represented (Breuer et al. 1983a; Maly, 1987). Some modeling schemes also include timing information, thereby allowing delay faults to be modeled (e.g., see Hayes (1985), Carter et al. (1987), Devadas (1989)). Unfortunately, there are many faults that can not be generically described at the gate-level. These are mostly caused by shorts that change the desired function of a gate, or can cause a change in the function of a group of gates (Cunningham et al., 1987; Galiay et al., 1980; Mangir, 1984). Since the effect of faults depends on the layout of the device and the specific gate topology, it is difficult to decompose these faults any further (except for the wired-logic behavior described above). In fact, it is even possible for a combinational circuit to become sequential due to certain fault modes (Galiay et al., 1980).

At the pin level of a chip, some general statements about fault effects on VLSI devices are now made. For certain regular structures such as programmable logic arrays and memories, more detailed fault models are discussed.

In all ICs, pins may be open due to broken bond wires. In one survey of RAC data (Stockman and Rash, 1985), between about 30 percent and 60 percent of classifiable failures were classified as input or output opens.

Stuck-at faults on output lines, of course, cause stuck-at failures at the corresponding pin. Stuck at faults on internal lines, however, will not result in such consistent behavior. In combinational logic circuits, all we can say in general, is that for a given stuck-at fault, there is an input for which the output is incorrect. Notice that many inputs may result in correct outputs even though a fault is present (this is what makes test vector generation and achievement of high fault coverage for VLSI devices difficult). This is also the case for failures (non stuck-at) that cause the function of the circuit to change (such as the wired-logic behavior of certain shorts discussed above). Failures that turn combinational circuits into sequential ones result in incorrect outputs for only certain inputs as well, but the occurrence of an incorrect output now depends on the sequence of inputs. As a result, a single input pattern may result in either correct or incorrect output patterns depending on the sequence of inputs previously applied. This is also true for stuck at faults in sequential circuits. Delays at the gate level result in response delays of the entire circuit, and finally, leakage faults can result in excess supply currents (Malaiya and Su, 1983) as seen from the supply pins.

Memory Devices

Memory devices (RAM, ROM, PROM, etc.) are highly regular structures that perform very well defined functions. As a result, we can be more specific about their modes of failure. In Agrawal (1988), memory faults are decomposed into parametric and functional faults. Parametric faults include low output levels and inadequate fan-out driving capabilities (due to sense-amp problems), slow access times, and loss of data. Functional faults include one or more memory cells being stuck at one or zero, coupling between memory cells and the inability to store or retrieve certain patterns (pattern sensitive faults). In Micro Control Company (1981) memory faults are also described. Pattern sensitive faults are discussed in detail in Franklin et al. (1989). Frequently the sensitivities are caused by certain cells that are affected by the values in their neighboring cells

(in the same row or in the same column). Soft failures in dynamic RAMs that result in intermittent data changes are discussed in Sie et al. (1977).

In Stockman and Rash (1985) there is evidence that 80 percent of the identifiable faults in bipolar memories result in dynamic characteristic errors (e.g., access time reduction), whereas for MOS memory devices 80 percent of the identifiable faults result in data loss.

Programmable Logic Arrays (PLAs)

PLAs are also very regular devices that can be thought of as a matrix of lines with transistor connections existing between some of the lines to define the desired input/output combinational logic. In Agrawal (1988), three fault types are considered: 1) stuck-at one or zero on the inputs, input inverters, product lines or outputs, 2) crosspoint faults in which a single transistor is added or missing, and 3) bridging faults in which shorts between adjacent lines occur, resulting in wired-logic behavior (type of wired-logic depends on the technology of the PLA). The effect of all of these faults is either a stuck output or, more likely, a change in the circuit function being implemented. The gate-level s/a fault model is sometimes used to predict the effects of PLA faults, but this is frequently inadequate for crosspoint and bridging faults (Chang and Abraham, 1987). In Chang and Abraham (1987) a "personality matrix" is used to more accurately predict the faults on PLAs and ROMs. Note that crosspoint failures can occur in the field since every crosspoint has a device, whether or not it is used. In Fujiwara (1985) it is claimed that most crosspoint faults can be modeled as stuck-at faults in the two-level AND-OR logic implemented by the PLA.

Microprocessors

While the microprocessor is far from a regular structure, it does have a well-defined function. According to Agrawal (1988) some success has been obtained in decomposing microprocessor faults into five classes: 1) register address decoding, 2) instruction decoding, 3) data storage, 4) data transfer, and 5) data manipulation.

4.3 EXAMINATION OF ACTUAL FIELDED-EQUIPMENT FAULTS

In this subsection we review a study of actual field faults for the STE/ICE and STE/ICE-R, two generations of diagnostic and prognostic O-level tools developed for the Army by GE-ASD. As the depot repair facility for STE/ICE and STE/ICE-R, GE-ASD maintains for the Army a system to track the testing of field failure returns.

Whenever a failed unit (UUT) is tested, the results of the test are manually entered by a test technician on a UUT Test Result form, and subsequently entered into a computer database in a coded/abbreviated manner. (A blank form is shown in Fig. 4-1, and an example of a filled-out form is shown in Fig. 4-2.) Since the content and accuracy of the database are dependent on the tester who enters the data, the result forms are used primarily as a vehicle to provide information for rework and repair of the UUT (e.g., which components are to be replaced). Also, detailed failure data (i.e., which test procedure detected the failure, blocks 12 - 14) is not typically entered. Furthermore, even if it were entered, that test procedure may consist of hundreds of individual tests, and the tester may not have access to or knowledge of the actual test performing the diagnostics. Once detected, the fault is then isolated to an ambiguity group, which typically contains many components, and the entire ambiguity group is replaced. Consequently many good parts may be replaced. The end result is that the UUT Test Result database contains failure data that represents both failed and unfailed components, with at best high-level information about the failure mode. Because of its position as the depot for STE/ICE and STE/ICE-R, GE-ASD receives the highest quality maintenance information available for those systems. Therefore, we may infer that the type of information available to GE-ASD and reviewed here is representative of the most detailed maintenance information available in general on fielded military electronic equipment.

Table 4-2 shows maintenance data for the top ten STE/ICE and STE/ICE-R component failures for 1989; Table 4-3 shows identical information for the first two months of 1990. (Note: these tables do not include data for components that were screened for workmanship errors; a large number of one type of 8K × 8 EE PROM fit into that category, failure mode unknown.) In Tables 4-2 the predicted failure numbers were obtained using the MIL-HDBK-217E parts-count

UUT TEST RESULT No.

UNIT IDENTIFICATION

XACT	SHOP ORDER	PART NUMBER	LOT	ASSY.	JOB	SYSTEM	WORK CTR.

FOR UNITS TESTED

DATE	SERIAL NUMBER	RESULTS - CIRCLE ONE
		AC WK OS TP EL TE PL RE UN

DETAILED DATA

ASSOCIATED NO.	PARA. NO.	STEP NO.	READING / DISCREPANCY

REF. DES.	PART NO.	S/N OR LOT CODE	TESTER CLOCK NO.

AE 1594 (11-42)

COPY 1

Figure 4-1. Blank UUT Test Result Form (courtesy GE-ASD)

UUT TEST RESULT No.

UNIT IDENTIFICATION

XACT	SHOP ORDER	PART NUMBER	LOT	ASSY.	JOB	SYSTEM	WORK CTR.
	WH101-501	896AS889		PC	U/I		730

FOR UNITS TESTED

DATE	SERIAL NUMBER	RESULTS - CIRCLE ONE
7/20/90	0200	AC (WK) OS TP EL TE PL RE UN

DETAILED DATA

ASSOCIATED NO	PARA. NO.	STEP NO.	READING / DISCREPANCY
			3/ R80 THRU R95 wrong Parts

REF. DES.	PART NO.	S/N OR LOT CODE	TESTER CLOCK NO.
R80 - R95	M5542K05B2H00P		11819

AE 1594 (11-42)

COPY 2

Figure 4-2. Example of Filled-In Form (courtesy GE-ASD)

method. The ten failure mode values in Table 4-2 represent the granularity of maintenance failure-mode data available at the depot. The part numbers shown in the table have been assigned by the Army for the STE/ICE and STE/ICE-R programs. The type of technology employed (TTL bipolar or CMOS) is indicated for the IC components. The assessment that an actual failure rate was significantly higher than predicted was made by GE-ASD.

TABLE 4-2. TOP TEN COMPONENT FAILURES FOR 1989 AND 1990 (2 MOS.)

					FAILURE MODE								
COMPONENT	PART #	TECHNOLOGY	ACTUAL FAILURES	PREDICTED FAILURES	0	1	2	3	4	5	6	7	8	9
1989														
ANALOG MUX	12303710	CMOS	193	209	25	121					43	4		
DIFF 4 CHANNEL	12258802	CMOS	*98	1	81	17								
RESISTOR NETWORK	12303724	–	96	50	4	43			1		48			
RESISTOR	RCR07G164JR	–	*75	0	1	74								
QUAD OP AMP	12322423	TTL	*52	12	15	33					4			
HEX D FLIP-FLOP	12258836	CMOS	*49	13							49			
12 BIT A/D CONVERTER	12258843	TTL	*46	10	4	7					35			
UV ERASABLE PROM	12310061	TTL	40	415			1		1		38			
ATTENUATOR HYBRID	12258862	–	*39	1	26	10				3				
ATENUATOR NETWORK	12303728	–	39	16	15						19			
1990 (2 mos.)														
SWITCH-DIP	M53504/01-027	TTL	*77	1	1					9			65	2
ANALOG MUX	12303710	CMOS	50	35	16	31					3			
KEYBOARD 4x5 MATRIX	12315257	–	39	33		1			4		33		1	
DIODE	JAN1N963B	–	*31	1	27	4								
OCTAL LATCH/DRIVER	12315213	TTL	*23	4	1	3					19			
INSTRUM AMP	12258842	TTL	22	14	8	11					3			
ATTENUATOR NETWORK	12303728	–	21	187	5	15								1
CAPACITOR	M83421/01-1117M	–	*21	0	19	2								
INSTRUM AMP	12303715	TTL	18	29	10	7					1			
CAPACITOR	M39014/01-1299	–	*17	7	15	2								

FAILURE MODE LEGEND

0 Low Value or Low Gain
1 High Value or High Gain
2 Wrong Part, Missing Part
3 Installed Wrong or Labelled Wrong
4 Shorted
5 Open
6 Won't Switch, Stuck High or Low
7 Overstressed, Cracked, Broken, Bent
8 Intermittent
9 Oscillates, Noisy, Spikes

* = significantly more failures than predicted

Examination of Table 4-2 indicates that no single component had significantly higher than predicted failures in both time periods, and that eleven components had significantly higher than predicted failures over the two time periods. Six integrated circuits were in this category, as were a resistor, an attenuator hybrid (resistor network), two capacitors, and a diode. The vast majority of the IC failure modes fall into categories 0, 1, 6, which are high/low value or gain and stuck high/low. These data suggest that stuck-at fault models represent a significant number of fielded faults, but that high and low gain faults are other important modes. In addition, intermittent faults

(category 8) appear to be the dominant failure mode for switches. The distribution of actual failure modes, such as shown in Table 4-2, can aid engineering judgement in determining failure mode rates from device failure rates calculated using MIL-HDBK-217 or a similar source.

4.4 SUMMARY: IMPORTANT FAILURE EFFECTS TO CONSIDER IN MODERN ELECTRONIC EQUIPMENT

In summary, it appears from the literature that a small number of failure mechanisms and generic effects should be considered when designing, analyzing, or verifying a diagnostic capability.

At the physical fault-mechanism level, it appears that corrosion, electromigration, electrical overstress, and electrostatic discharge are the dominant mechanisms that cause fielded electronic equipment to fail. Also important for MOS technology are time-dependent dielectric breakdown and hot electron degradation. The effects of these mechanisms on the behavior of the transistors, connectors, and interconnects that make up a device are primarily shorts and opens on transistor connections, shorts between metal lines, open metal lines, and changes to transistor threshold levels.

At the gate level, these mechanisms translate into stuck-at lines, changes in gate function (not representable by stuck-at faults), changes in circuit state dimension (e.g., a combinational circuit can become sequential), and increased rise and fall times (delays).

At the chip level, these failures result in some type of incorrect output for some subset of inputs and response delays. The incorrect output type of fault may be dependent on the sequence of inputs and includes stuck-at and stuck-open behavior of a device's output as well as other type of incorrect outputs.

When a device has a regular structure, more can be said about the effects of physical fault mechanisms at the device boundary. Memories exhibit several specific faulty behaviors (Agrawal, 1988) though there is evidence (Stockman and Rash, 1985) that data loss is the primary field-experienced failure. PLAs also exhibit special types of functional failures as discussed in Chang and Abraham (1987). In addition, grouping faults according to the device functions that are

affected can be useful for test design and qualitative coverage assessment (does a test detect a particular function failure?), but is useless for quantitative coverage analysis (what percentage of actual fault mechanisms are detected by the test method?).

Finally, the information gleaned from our examination of STE/ICE and STE/ICE-R data, which we feel is representative of the most detailed maintenance information available in general on fielded military electronic equipment, is the following:

- a larger than expected number of 8K × 8 EE PROMs failed, with the cause traced to bad workmanship, fault mode unknown,
- the vast majority of IC failure modes are high/low value or gain or stuck high/low, and
- intermittent faults appear to be the dominant failure mode for switches.

As discussed in subsection A.3, we have utilized the above information in our selection of faults to be inserted and emulated in the demonstration by emphasizing stuck-at faults, shorts, and opens, and a significant number of faults on the EEPROMs. We also note that intermittent faults on any device type will continue to be a challenge to diagnostics designers and verifiers alike.

SECTION 5

THE ROLE OF FAULT SIMULATION

One of the main goals of this effort was to determine the roles of fault simulation and fault insertion in the verification of diagnostic capability. In Section 3, we argued that validation of diagnostic performance measures through factory fault insertion alone was not possible, and consequently proposed two separate procedures: 1) diagnostic design verification–to demonstrate that a diagnostic capability can detect the faults it is supposed to detect through factory fault insertion, and 2) test-effectiveness analysis–for approximate validation (or really prediction) of diagnostic performance measures.

As a result of this separation, the roles of fault insertion and fault simulation become very clear. Fault insertion is used as part of the demonstration procedure. It is used solely to demonstrate that the diagnostic capability has been implemented in accordance with its design requirements. For example, if in the design of the diagnostic capability it is determined that a particular type of fault in a particular component should be detected, then this fault should be inserted during the demonstration (or emulated if fault insertion is not possible, see Section 6). Fault insertions can't be used to predict diagnostic measures like FFD and FFI because failure rates and failure modes are uncertain at this stage in the development process and because random selection of real faults for insertion or emulation is not generally feasible (e.g., due to accessibility limitations and damage concerns).

Simulations potentially provide more visibility into the fault detection and isolation performance of a diagnostic capability because more faults can be feasibly considered. However, simulation techniques vary widely in terms of their utility for testability verification, and the existence of a suitable simulation for any given device, including requisite failure modes and diagnostics, is usually problematic. In addition, the problem of accurate prediction of failure rates

makes any simulation-based prediction of FFD or FFI only approximate. Finally, the costs and complexities associated with developing and verifying a simulation may be too high to be of practical value considering the potential inaccuracies of their results. Thus, we have concluded that simulations can not be relied upon for testability verification. If available, their role is to support the test-effectiveness analysis task by providing approximate values for the coverage of different board fault modes by different test-elements within the diagnostic capability.

In the remainder of this section, we provide an overview of simulation techniques and briefly discuss their role in support of the test-effectiveness analysis task. In addition to simulation-based fault coverage methods, we also discuss some methods based on statistical design analyses (e.g., STAFAN (Jain and Agrawal, 1988)). Methods for organizing and performing the test-effectiveness analysis task are discussed in detail in Section 8.

5.1 FAULT SIMULATION TECHNIQUES

In general there are two ways fault coverage of a diagnostic capability can be determined; Simulation and Design-Analysis. Simulation refers to software that *replicates circuit behavior* (at some level) in both faulty and nonfaulty modes of operation. The results of simulations are used to "grade" diagnostic systems by comparing faulty and nonfaulty simulation results (for FFD-type analysis) and by comparing faulty simulation results (for FFI-type analysis). Design-Analysis refers to direct prediction of diagnostic capabilities without explicit replication of circuit behavior in faulty and nonfaulty modes. Design-Analysis uses a description of the circuit (as does simulation) to determine if faults are inherently detectable or isolable. Descriptions for design-analysis may be the same as for simulation (as in fault-grading methods such as in Agrawal (1988)), or drastically simplified descriptions of the diagnostic capability itself (such as the fault accountability tables employed in Section 8 of this report). This section discusses simulation techniques. Subsection 5.2 provides an overview of design analysis techniques that utilize logical models of the circuit under consideration. Section 8 considers design methods that utilize higher-level descriptions of the diagnostic capability.

Simulation techniques can be characterized by three features: 1) modeling technique, 2) sampling technique, and 3) extent of simulation. To be useful, the simulation "engine" should be independently verified (e.g., using a commercial simulator usually guarantees that the simulation engine performs correctly, while the validity of customized simulation programs are more difficult to establish).

The modeling technique determines what type of simulation engine is employed. There are three generally accepted modeling techniques:

- Device models. Modeling of all lumped elements such as transistors, resistors, etc. is accomplished through equations describing each device's V-I characteristics. Voltage and current responses are then determined by the circuit network's topology, e.g., SPICE (Nagel, 1981),
- Switch models (applies to digital circuits only). Each transistor is modeled as a switch that is either on or off or in transition. The latter allows timing information to be derived. Logic levels vs. time are determined by the state of each switch and the circuit topology),
- Logical models. Only the logical relationship between gate, cell, or chip inputs and outputs are represented. When logical models are used to represent groups of gates (cells) or an entire chip it is referred to as a register-transfer or functional level model. Logic levels at each clock cycle are determined by the interconnection of logical models, e.g., LASAR (Thomas, 1971).

In each of these methods, fault simulation is accomplished in a different way. With device models, new voltage-current characteristics for faulty devices must be defined (e.g., see Weiss et al. (1989)). For switch models, stuck-at logic levels and delay-type faults are simulated by modification of gate outputs. For logical models, only stuck-at faults *at the interfaces between cell models* are usually simulated by modifying cell-output logic levels. Note that when a cell consists of many gates, this type of fault simulation omits many possible fault modes (see Section 4).

5.1.1 Gate-Level Logical Models

The most common simulations in use for digital circuits use logical gate models. In gate-level logical modeling, the state (0,1) of every node in the gate-level circuit network is determined for both nonfaulty and single s/a fault modes under a set of test patterns. If there is a difference at the output between nonfaulty and a single faulty simulation for any test pattern, the fault is said to

be detected. Highly efficient ways of simulating faults at the gate level have been developed. Today, the use of concurrent, table-driven fault simulation techniques and the use of hardware accelerators are most common and very effective (e.g., see Agrawal and Seth (1988)).

Even though efficient gate-level simulations exist, exhaustive fault simulation (i.e., all faults are inserted and all test vectors are simulated for each fault) for large circuits is often not possible (note that if all test vectors are applied, all faults of interest are guaranteed to effect the device output for at least one vector. If no output data compression is used in the test technique, then this implies 100 percent coverage and simulation is not necessary). This is true despite techniques such as fault dropping (when you are determining FFD, you needn't simulate any fault that has already been determined to be detectable). Because exhaustive fault simulation is often not possible, sampling techniques are used. The most common methods use standard sampling equations (for both finite and infinite populations) to give confidence measures for FFD (e.g., see Agrawal and Seth (1988) p. 241). Typically only one to two thousand faults need to be inserted to get high confidence using these methods. However, one should note that the confidence intervals depend strongly on the ability to select faults at random (and thereby generate independent observations). If, for example, there are many missed failures that all lie in one region of the circuit, then accurate confidence intervals will not be obtained by sampling in any systematic way according to the layout. This suggests that, where possible, potentially hard to detect faults be determined by other means, and then simulation used to determine coverage of these faults. While most sampling methods only deal with sampling of faults (and applying all test vectors), one could easily consider sampling the test vectors for simulation (e.g., simulating all faults and some vectors, or just some of both) as well. Much has been written about the use of gate-level logical models and fault simulation for determining coverage of testing schemes. The latest Air Force policy can be found in Debaney (1989)

5.1.2 Register-Transfer-Level Logical Models

Commercially available simulation packages have focused primarily on the gate-level logical model for use in finding test vectors for the detection of faults in VLSI chips at the chip foundry. The use of these packages for designing board-test schemes has been limited in the past, but is increasing. This has required simulation vendors to develop register-level logical models for commonly used digital cells. In most packages (e.g., LASAR) these models represent normal, unfailed behavior only. Fault simulation is, therefore, limited to stuck-at faults at the interfaces between register-level models only. Furthermore, current register-level models are of limited use in simulating the different modes of devices. While each mode of a device can be simulated in many packages, the switching of modes in response to external stimuli is frequently not simulated. Furthermore, register-level device libraries are, today, limited in scope and device vendors are not always willing to provide such models for use in board test development.

Despite these limitations, the concept of board simulation using register-level logical models of devices is a useful one for design and evaluation of diagnostic capabilities. For example, one suggestion involves using such simulations for evaluation of the diagnostic performance measures with which this report is concerned (FFD and FFI). The idea here is to simulate the operation of a board during operation of the diagnostic capability. If diagnostic circuitry is located on the board, this would be modeled and simulated as well. Similarly, if the diagnostic capability includes a self test program that is run by an on-board microprocessor, simulation of the execution of that program would be done. Stuck-at faults on lines connecting register-transfer-level models would then be inserted into the simulation to determine if the diagnostic capability does in fact detect the fault as it operates. Essentially, this idea amounts to simulation of the operation of the diagnostic capability in addition to simulation of the hardware responses of the unit under test. Of course, such an approach would not be capable of evaluating detection or isolation accuracy for all failure modes since many failures can't be represented by the fault simulation method discussed above (see Section 4). However, since the faults that are

simulated in the above manner are a subset of all faults, this approach would give upper bounds to FFD and FFI.

5.2 DESIGN-ANALYSIS TECHNIQUES

Most established/automated methods of design-analysis apply to gate-level logic models. They are largely based on probabilistic analysis in which it is assumed that test vectors are chosen at random. This could be the case, for example, if test vectors are generated automatically using a linear feedback shift register (on-line or off-line). The methods then try to determine the probability that a s/a fault on a particular line will be detected by a given number of test vectors. These probabilities are then added across all failure modes to give an estimate of test coverage. There are usually several approximations that must be employed to make the calculations feasible, and as a result, less accurate. For example, many methods compute two figures called controllability (the probability that a line can be set to 0 or 1) and observability (the probability that a path to the output is sensitized to a line set at 0 or 1) (Agrawal and Seth, 1988). Assuming independence, detectability is the product of these two numbers. Unfortunately, independence is not valid in many cases because the same underlying circuitry can determine both controllability and observability. Methods for determining controllability also sometimes use the assumption that input lines to a gate are independent. This is violated whenever there is reconvergent fanout. Research in this field is currently active. Most new methods try to provide more accurate results at the expense of somewhat more computation. All analysis methods are much faster than fault simulation, however. Examples of commercial versions include STAFAN (Jain and Agrawal, 1988), PREDICT (Seth et al., 1988), and COP (Brglez, 1988). Other ad-hoc methods for determining lower bounds on test coverage include the toggle test in which one simulates the nonfaulty circuit and determines which lines are never toggled to both 0 and 1. Stuck-at 0 and stuck-at 1 can not both be detected on such lines. This provides an upper limit to test coverage for a set of test vectors.

SECTION 6

GUIDELINES FOR PERFORMING FAULT INSERTIONS

6.1 INTRODUCTION

This section provides guidelines for demonstrating the fault detection and fault isolation ability of the diagnostic capability of the unit under test (UUT) via physical and software fault insertions (subsections 6.2 and 6.3, respectively).

The term fault refers to the condition when one or more functions of the UUT fails to meet its performance specification. A diagnostic error is defined as the condition when the diagnostic system fails to detect (or isolate, if required) *and* report a fault, or when it produces a false alarm (e.g., an error message is generated during a no-fault condition). If the UUT is part of a larger system, there must be clear, unambiguous definitions/partitions of the diagnostic responsibility of the UUT. This is necessary for the determination of diagnostic errors.

Each fault inserted via the techniques in subsections 6.2 and 6.3 has the following attributes when it is applied to a particular UUT:

1. Ease of Insertion and Ease of Removal
2. Risk of Permanent Damage (to the affected component and its neighboring components)
3. Cost (e.g., for special equipment)
   a. to insert
   b. to remove
   c. to replace, if damaged
   d. to monitor (for verification of the fault insertion and its subsequent removal)
4. Significance—is the fault representative of an actual/real fault with significant probability of occurring? (For example, a resistor has two fault modes, but typically whenever it experiences a short, it quickly heats up and becomes an open)
5. Uniqueness—is the fault well-defined and isolated, or does it permeate into other parts of the circuit, as with bus faults?

Each of these factors must be considered and weighed for each candidate fault insertion when designing the demonstration to meet the objectives stated in the Demonstration Plan (see Section 7).

Of course, a primary concern is to avoid damaging the UUT during the fault insertion demonstration. Physical damage avoidance is important not only because the UUT may be expensive or of limited number, but also from the point of view of the validity of the demonstration–one cannot guarantee that the diagnostic system is responding to the inserted fault versus the faults induced by the physical damage. In general, components can be damaged if they are subjected to excessive temperature, current, or voltage, with high temperature being the primary cause of integrated circuit (IC) failure. These temperature-related failures can be related to combinations of excessive current through, and the resistive properties of, the semiconductor junctions and wirebonds of the IC. Extreme currents can also weaken the metalization layer of the IC, and excess voltage can cause problems in the silicon-silicon dioxide interface. For CMOS devices, voltage overshoots can cause latch-up and catastrophic device failure. Also note that printed circuit boards and integrated circuits may be sensitive to electrostatic discharge (ESD). These components shall be clearly marked and handled in accordance with their ESD prevention procedures (see MIL-STD-1686 and DoD-HDBK-263 for details).

In order to provide context for the fault insertion techniques given in the following two subsections, we give here a brief overview of the demonstration procedure (see Section 7):

1. Prior to the demonstration, the UUT should undergo board test to verify that it is fully operational and does not contain any manufacturing defects that would obscure any results of the demonstration. This board test will be an in-circuit test of the UUT's hardware and a comprehensive functional test, typically via a board-test connector.

2. With the UUT fully operational, the correct operation of the diagnostics should first be demonstrated. This procedure will execute all of the diagnostics and verify that the tests performed by this software do not detect any errors (which would be classified as false alarms).

3. The fault insertion demonstration will be performed on a fully operational UUT with known faults inserted one at a time. Faults will be inserted using both hardware and software methods. Typically each fault class (as listed in the Test-Effectiveness Analysis (see Section 9)) will have at least one fault inserted.

6.2 PHYSICAL FAULT INSERTION TECHNIQUES

This subsection defines techniques in general terms for inserting faults into the UUT. To generate a non-trivial, non-destructive demonstration, an engineering prototype with socketed components is typically required. This is because production units may be conformally coated or constructed in a manner (e.g., surface-mounted components) resulting in the lack of accessibility and/or limited fault insertion capability. Note that if an engineering prototype is used, then the contractor must certify that the prototype is a complete and faithful representation of the functional capabilities of the production unit.

Power must be turned off to the UUT before any fault can be inserted (although shorting pins together may be performed after initialization, if it can be safely performed and is more convenient). To insert a fault, first determine on which board of the UUT the given fault is to be inserted, and carefully remove this board from the UUT. Once the board is out, the fault may be inserted using one of the techniques listed below. Next, the board is placed back into the UUT. At this point the UUT can safely be turned on and the system initialized. Faults to be inserted into the UUT generally fall into ten categories, and the steps for inserting each are describe below.

1. Shorting a Pin to Supply or Ground

This fault consists of using a shorting probe lead to connect a given component's pin to either ground or supply. Connect one end of the probe to ground or supply, and connect or hold in place the other end to the proper component/pin. If it is more convenient and can be accomplished without damaging the device, these voltages can be taken from the power supply inputs to the given component. When manually holding or connecting the probe, be sure that the probe is not touching any pins other that the given pin, because inadvertently shorting some pins to ground or supply can damage the affected components. As a precaution, the shorting voltage should not be applied to the component until the component/board is properly powered. Typically, this fault is applied to isolated input pins, or the driver of that pin can be disabled. If the pin cannot be isolated, it may be necessary to place a resistive load in the shorting probe to buffer the connection.

2. Removing a Pin and Shorting It to Ground or Supply

This fault consists of removing a specific pin of a socketed component. To insert this fault pry the component out of its socket and carefully bend the appropriate pin upwards about 90 degrees. Care must be taken to prevent breaking the delicate pins. Carefully replace the component with the bent pin out of the socket. It may be necessary to insulate the exposed pin (e.g., with a small piece of paper or non-conductive tape) to ensure that it makes no electrical contact with its socket or any other pin. Next, use a shorting probe to connect this pin to either ground or supply. As a precaution, the applied voltage should not be applied until the component/board is properly powered.

This technique is similar to technique 1, but it can only be selected for those components that are socketed. Thus, no unsoldering of components is necessary, as this process could either damage the component(s) or the board on which it is mounted. This technique is suitable for inserting an isolated fault on input pins.

3. Removing a Pin of a Socketed Device and Applying either Ground or Supply to the Vacant Socket.

This fault consists of removing a specific pin of a socketed component from the circuit and inserting an external voltage in its place. To insert this fault pry the component out of its socket and carefully bend the appropriate pin upwards about 90 degrees. Care must be taken to prevent breaking the delicate pins. Insert one end of a wire into the proper socket location that is being vacated. Carefully replace the component with the bent pin out of the socket. It may be necessary to use an insulator (small piece of paper) to ensure that the bent pin makes no electrical contact with its socket, the shorting wire, or any other pin. Next, connect the shorting wire to either ground or supply. The voltage should not be applied until the component/board is properly powered. This technique is suitable for inserting an isolated fault on a device's output connection while eliminating the possibility of damaging the internal output signal drivers of the component.

4. **Removing a Pin of a Socketed Device and Leaving it Unconnected to its Associated Circuitry.**

This fault consists of removing a specific pin of a socketed component. To insert this fault pry the component out of its socket and carefully bend the appropriate pin upwards about 90 degrees. Care must be taken to prevent breaking the delicate pins. Carefully replace the component with the bent pin out of the socket. It may be necessary to use an insulator (small piece of paper) to ensure that the bent pin makes no electrical contact with its socket or any other pin. This removed pin is not connected to either ground or supply.

5. **Removing a Socketed Component from the Circuit Board.**

This is self-explanatory. The component must be removed before power is applied to the board. To minimize any damage associated with the process of unsoldering components, only socketed components should be selected for this fault.

This type of fault represents a totally dead component whose input and output leads are completely open.

6. **Shorting Two Pins of a Device Together.**

Connect two pins of a device together using a wire jumper. Be careful not to touch the wire jumper to any other pin, creating unintentional, additional faults.

Faults of this type typically occurs to adjacent pins of a device or connector.

7. **Inserting Faults on the Connector or Backplane.**

To insert faults on a connector, an extension board may be use to facilitate the fault insertion and its removal. The extension card may have switches that provide the necessary open or short of any signal that is fed through it.

8. **Disconnecting a Board from the Backplane.**

This is self explanatory. This fault simulates an improperly seated circuit board.

9. Inserting Delays

Capacitors or long pieces of wire may be added between a removed pin and its vacant socket to introduce delays in the circuitry. A detailed circuit analysis is required to ensure that the proper amount of delay is being supplied to the affected pin.

10. Break-Out Boxes

Break-out boxes are normally used for board fault emulation. The fault injector of Hummel (1988) is essentially a break-out box concept for socketed components. It uses relays to create shorts and opens and can drive output to arbitrary logic levels to simulate incorrect device outputs. To be truly effective, some form of communication between the fault injector and the diagnostic software is needed so that a faulty output for a single pattern in a sequence of patterns can be injected, instead of fixing the output during the application of the whole test pattern.

11. Backdriving

Backdriving is a particularly useful technique for overdriving the output of components to control the inputs to the device-under-test as if there were no surrounding circuitry connected to the device. This technique is commonly used for in-circuit testing of boards during their production. To prevent damage to the output drivers being backdriven, the in-circuit tester provides the necessary backdrive current in pulses of approximately 10 to 100 microseconds. As a result of the precise timing requirements, in-circuit test is primarily performed using an expensive bed-of-nails tester with the printed circuit board removed for the UUT. Also, it is the tester that controls the function of the board and it is the tester's diagnostics that detects a fault.

This technique is not recommended for the fault insertion demonstration for the following reasons:

1. the board with the injected fault is removed and isolated from the UUT,
2. the UUT is not "live" (i.e., operating with system software), and
3. the in-circuit tester has control of the system, and it is the tester's diagnostics that detects the fault, thereby defeating the purpose of the demonstration of the UUT's diagnostics.

6.3 FAULT EMULATION

Fault emulation is defined as a fault that is inserted via software. Once inserted, the fault manifests the physical behavior of an actual fault. In fault emulation, the subject device is removed from its board and the emulator probe is placed in the vacant socket. It is via this probe that the emulator can control system operation and software execution and can directly read addressable memory or register contents.

1. Microprocessor Emulation

This technique is familiar to many as the method used by microprocessor development systems to aid in the hardware design and software development of microprocessor-based boards. It is simple to adapt this technique to functional board test. One replaces the microprocessor on the board-under-test with another microprocessor of the same type but which is under control of the tester (e.g., the in-circuit emulator, or ICE). The emulation microprocessor then executes a test program from an emulation memory, which may be the same as the memory on the board under test. Ideally, the emulation microprocessor that executes this test program will be written to apply patterns to the various devices located around the board, typically focusing on bus peripherals (addressable components) and memory.

This technique is most effective when dealing with devices that are located directly on the bus, since it is through the bus that all contact with the board is made. Microprocessor emulators are equipped with only limited ability to drive and detect digital states at random points on the board. Limitations exist as to the number of such points available, their protection against board faults, ability to overdrive the board, timing control, and the means to program them in a flexible fashion. For these reasons, test thoroughness and diagnostic precision decreases rapidly when one moves farther from the bus. For example control signals are typically generated or decoded separately from the bus affecting the tester's ability to control certain states.

The fault emulations employed by the Der. .ration Plan in subsection A.3 are of this type and are as follows:

1. Modify the expected value of the test. This can be done either before the test is executed, or any time until the expected value is compared to the test result. When the test is executed under fully operational conditions, an "error" will result.

2. At pre-determined breakpoints, halt test execution, insert erroneous data and resume the test. This erroneous data can be entered either manually by the operator or via application software.

3. Misdirect test execution by either skipping the execution of blocks of code or change addresses so that the test is misdirected (in terms of data or other addresses).

2. Memory Emulation

In memory emulation, the tester substitutes its memory for the memory on the board being tested. The board's microprocessor then executes a test program that has been loaded into this memory. In addition to the advantages of the microprocessor emulation listed above, this technique has the additional advantage that it will exercise the microprocessor on the board under test, instead of requiring its removal. As with microprocessor emulation, it does its job best when dealing with the bus directly and is weak when it comes to supplying or testing non-bus digital events or creating a particular environment for test.

Assuming that the test is executed out of emulation memory that is known to be good, and that enough diagnostic logic is put into the test program, the test can be quite thorough. For example, the test programmer should write a diagnostic routine that looks at the pattern of memory failures and determine if they are caused by a decoder problem, a bit problem, or a memory chip problem.

3. Bus Cycle Emulation

This technique involves making the tester hardware mimic the activity of the microprocessor's bus interface. The microprocessor can be viewed as having an arithmetic engine and a bus interface that connects that engine to the outside world. In normal operation the bus interface, under control of the execution engine, performs the specification book waveforms, which transfer data to and from memory or I/O space. The bus cycle emulation does the same thing by executing memory read and write cycles, but under control of the test programmer instead of the microprocessor. Such cycles can be used, for example, to send commands to a serial port or retrieve data from a floppy disk controller.

In order to perform a bus cycle emulation, the microprocessor on the board being tested must be made to give up its bus interface to the tester. The most straightforward way to do this is by applying a 'bus request' to the microprocessor and waiting for it to acknowledge that it has tri-stated its bus interface. The bus is then in the control of the tester, and bus cycle emulation can be performed.

SECTION 7

GUIDELINES FOR PREPARATION OF A DEMONSTRATION PLAN

The document outlined in this section supports the demonstration effort. It describes how the demonstration will take place and records the results of the demonstration. The demonstration should take place only after the diagnostic performance analysis report has been reviewed and changes to the diagnostic capability are made if necessary. The purpose of the demonstration is to show the Government that the diagnostic capability has been constructed in accordance with the design and requirements specifications (including the diagnostic performance analysis results) and that all functions are correctly operating. The demonstration may be treated as an acceptance test, and numerical acceptance criteria can be defined. The demonstration plan described below follows the total quality management (TQM) principle of constant improvement (USDoD, 1988). Instead of numerical acceptance criteria, the demonstration requires that a plan for correcting *all* problems found during the demonstration be established and agreed to by the contractor and the Government. All numerical criteria are based on engineering judgement.

An example Demonstration Plan for a piece of modern military electronic equipment is given in subsection A.3.

OUTLINE OF A DEMONSTRATION PLAN AND REPORT

1.0 INTRODUCTION

1.1 Purpose

State the purpose of this document (to describe the demonstration test procedures for the identified diagnostic capability and the results of the application of those procedures). Refer to the documents that describe the diagnostic capability and the primary equipment under test.

1.2 Scope and Acceptance Criteria

Describe the diagnostic capabilities that are to be demonstrated. Define the means of detection and, if necessary, the means of isolation. Define the criteria to be used for

acceptance of the identified diagnostic capability. Use of statistical success criteria (e.g., m out of n successful demonstrations) is not recommended because of the inability to select faults randomly in modern complex equipment and the inaccuracy of failure rate prediction methods. It is better to state how many errors (zero is recommended) will be acceptable (errors are only acceptable when accompanied by a corresponding task resolve the problem—see step B in 1.3 below) and how many reworks will be allowed under the current contractual situation.

1.3 Overview of the Demonstration Procedure

Describe the sequence of events to take place during the demonstration and identify any decision points within this sequence. The following is a reasonable example:

Step A) Demonstrate correct operation of the diagnostic capability when the system is operational (go-chain demonstration).

Refer to Section 3 of this plan. Provide an overview of how the demonstration will take place (e.g., for a TPS, run the go chain; for a power-up test, turn the power on; for initiated BIT, initiate BIT sequence; for monitoring BIT, operate the system in all of its modes of operation and possibly provide stressful inputs and/or environmental conditions). State that if any fault is declared for an operational system, the contractor will determine the cause of the error, correct the problem, and redemonstrate operation of the diagnostic capability.

Step B) Demonstrate correct operation of the diagnostic capability under various fault conditions.

Refer to Section 4 of this plan. Either state that all fault types are demonstrated or list the fault types that can not be demonstrated and explain why (Section 2, paragraph 1, of this plan). Step A should be redone between each fault demonstration (after the fault has been removed) to ensure that no permanent fault other than the one being demonstrated is present. Either state that all possible diagnostic outcomes resulting from faulty system operation will be demonstrated or describe those that can't be demonstrated and explain why (refer to Section 2, paragraph 2, of this plan). State that for every incorrect demonstration result, the contractor will, if possible, correct the error, and later demonstrate that this particular error was corrected. State that in order to demonstrate the correction of each error, the specific fault that caused the error will be inserted or emulated and the Government, at its option, will select faults from the alternate list that are related to the error and these will also be demonstrated. State that if the error can not be corrected, the contractor shall analyze the impact on overall system performance (refer to diagnostic performance analysis report). Also state that if the Government, after reviewing the analysis, determines that a significant impact is anticipated, the contractor shall prepare a plan to mitigate the error. Assuming that the Government accepts the resulting mitigation plan, a subsequent demonstration shall be held, at which the known error will be counted against the system in determining the required number of faults to be successfully inserted/emulated by the contractor (see 2.1 below).

Step C) State that the Government, at its option, will select arbitrary faults from the alternate fault list. A predetermined number shall be selected, plus a number that shall be a function of the number of errors encountered in step B. For example, If x of the demonstration results in Step B were incorrect, select x more from the alternate list. State that any errors that occur in this step will be treated as in B. and that any errors in the repeating of step B will be treated as in Step C. State the number of times the iterative process will go on before the demonstration is terminated.

Step D) Finally, state that the government will accept the identified diagnostic capability when all errors uncovered during the demonstration have either been corrected or a plan for analyzing performance and correcting errors if needed, is provided. State that the plan for correcting errors will include a schedule that is mutually agreed to by both parties. State that the plan may include repetition of parts of this demonstration.

1.4 Responsibilities

Detail the responsibilities of Government and contractor for:

a) Providing facilities and equipment,

b) Witnessing of the test,

c) Formal acceptance,

d) Rework, if necessary.

1.5 Support Requirements

State what facilities, equipment, and personnel are needed for the demonstration to take place.

1.6 Applicable Documents

Refer to the design documents that contain detailed descriptions of the identified diagnostic capability (e.g., performance and design specifications for software, the diagnostic performance analysis report, appropriate hardware specifications, etc.).

2.0 COMPLIANCE DATA

State the comprehensiveness of the demonstration plan with respect to the following requirements.

2.1 Number of Faults

The number of faults to be demonstrated in Section 4 of this plan will be determined in accordance with the MATE guidelines for TPS verification, which specify (in tables) the necessary number of successful insertions, with an allowable number of errors, to reach a desired level of confidence in a desired level of performance (USAF, 1985, Section 6). State the MATE requirement for the number of demonstrations and the number of demonstrations (not including those that may be selected by the Government) to be performed in this plan.

2.2 Distribution of Fault Classes

The design requirements of the diagnostic capability will list the fault classes that the diagnostic capability is designed to handle (detect or isolate). These fault classes may be defined as components, component groups, functions of components, individual cells within components, or component failure modes. It is recommended that the percentage of these classes that are demonstrated in this plan should be at least 80 percent. State this requirement and the percentage actually achieved in this plan. If the plan does not meet the requirement, the reason for noncompliance must be justified and agreed to between the contractor and the Government. (NOTE: when repetitive circuitry exists, the fault classes defined for this circuitry should be merged to avoid multiple counting).

2.3 Distribution of Failure Rate

Each fault class to which the diagnostic capability was designed should be assigned a failure rate (in accordance with the test effectiveness analysis, if it was produced). Failure rate information for components within classes should be obtained using MIL-HDBK-217 or a similar source. Rates for particular failure modes for a component must be obtained using engineering judgement. If an FMEA was performed, splitting the component failure rate equally among the modes is a reasonable first-order approach. It is recommended that the sum of the failure rates for those fault classes that are demonstrated in this plan should equal or exceed 90 percent of the total failure rate defined by the design fault classes. It is further recommended that the sum of the failure rates for the fault types of which each individual demonstrated fault (and its diagnosis mechanism) is representative equal or exceed 30 percent of the total failure rate defined by the design fault classes. State these requirements and the percentage actually achieved in this plan. If the plan does not meet the requirements, the reason for noncompliance must be justified and agreed to between the contractor and the Government.

2.4 Comprehensiveness of Software Test

If the diagnostic capability contains any software element, all paths through this software should be executed in this plan. That is, all lines of code will be executed somewhere within this plan. State this requirement and the percentage of code lines executed by the test plan. If the plan does not meet the required 100 percent, the reason for noncompliance must be justified and agreed to between the contractor and the Government.

2.5 Comprehensiveness of Hardware Test

If the diagnostic capability contains any pure hardware test element, at least 85 percent of those elements should be demonstrated in this plan. State this requirement and the percentage of hardware elements demonstrated by the test plan. If the plan does not meet the required percent, the reason for noncompliance must be justified and agreed to between the contractor and the Government.

2.6 Comprehensiveness of Diagnostic Outcomes.

The diagnostic outcomes of interest are defined in subsection 1.2 of this plan. The requirement is to demonstrate all possible distinct outcomes of interest. State this requirement and the percentage of all distinct diagnostic outcomes that will be demonstrated and justify and agree to any noncompliance. As an example, in a TPS, this requirement states that all R/R call outs (fault isolation outcomes) must be demonstrated along with the go-chain. For fault detection systems, only the Go-path (diagnostic outcome = "OK") and any No-Go paths are required. Note, however, that in this case many repeats of the same No-Go outcome may be required in order to conform with requirements 3 through 5 above.

NOTE: The set of diagnostic outcomes includes all of the different manifestations of the specified means of fault detection or fault isolation. For detection, usually there are only two outcomes (e.g., pass/fail). Organize the outcomes so that the outcomes associated with each means of detection or isolation are grouped together. For isolation, the set of possible diagnostic conclusions is the set of outcomes. In describing the diagnostic outcomes, a complete list is preferable if possible (e.g., all of the terminal nodes in a TPS decision tree). If the diagnostic capability can produce arbitrary combinations of a variety of basic outcomes, then only these basic outcomes should be listed. However, in this case,

all multiple basic-outcome occurrences that are to be demonstrated should be identified here.

3.0 DETAILED DEMONSTRATION PROCEDURE FOR OPERATIONAL SYSTEM

Describe the detailed procedures for demonstrating that the diagnostic capability responds correctly when no failures are present. Describe what should be witnessed as the result of this demonstration (i.e., expected results). For monitoring types of diagnostic capabilities, the primary system should be exercised in all of its modes of operation and if possible, stressful inputs and/or environmental conditions should be applied. In all cases, the full operational status of the primary equipment should be independently established (e.g., through the use of board test) before the start of this step.

4.0 DETAILED DEMONSTRATION PROCEDURE FOR FAULTY SYSTEM STATES

Describe the insertion/emulation procedures for both primary and alternate demonstration faults. The primary faults are the faults selected by the contractor for initial demonstration and the alternate faults are those selected by the Government. For each fault, describe

a) the procedure for insertion or emulation,

b) how to tell if the correct outcome was produced,

c) a list of the diagnostic outcomes checked (cf. Section 2, paragraph 1, of this plan),

d) the procedure for selecting, inserting, and emulating alternate faults.

If the potential exists for unanticipated faults to be introduced when performing a fault insertion, then full operational status of the primary equipment should be established after the fault is removed (e.g., by executing the steps detailed in section 3.0 of this plan).

5.0 DEMONSTRATION RESULTS

This section is not provided for the initial submission of this document, which is for approval of the demo plan. For the final post-demo submission include :

a) a log of all demo activities and results,

b) plans for correcting or mitigating all errors that could not be corrected during the demo.

Collect statistics on the demonstration results including :

a) number of faults demonstrated,

b) number resulting in correct diagnostic outcomes,

c) breakdown of (b) by diagnostic outcome (if more than one),

d) percentage of fault classes demonstrated,

e) list of functional/operational modes demonstrated.

If it is determined that during the execution of any demonstration step, a real fault (other than any being inserted) was present, the result will be listed as a separate fault-insertion step and the result recorded as if the real fault was planned. For example, if the real fault is detected and/or correctly isolated during Step A, this will be listed as a correct fault insertion in Step B. On the other hand, if a real fault is present in addition to an inserted fault in Step B and the result is non-detection or incorrect isolation (i.e., neither fault is

called out in the final ambiguity group), then this will be listed as an incorrect (multiple) fault insertion for Step B.

SECTION 8

ANALYSIS METHODS

8.1 OVERVIEW

In this section we present two methods of quantifying FFD and FFI based on raw fault accountability data. The first method is a probabilistic approach that assumes independence of fault detection events across test elements. The second method has been developed to overcome the unrealistic independence assumption. Instead, bounds on FFD and FFI are computed based on minimal assumptions about unknown set intersections. Both methods require the following information to be specified:

1. a list of tests and their possible outcomes,
2. a comprehensive list of fault classes (preferably disjoint) for the system under consideration,
3. the fraction of faults in each class that can cause each test outcome (the raw fault accountability data), and
4. the decision logic that relates the relevant diagnostic outcomes to the possible test outcomes.

Figure 8-1 illustrates the information that is needed to perform either analysis. The fault coverage information is organized in a table with fault-class identifiers used as labels for the rows and test-outcome identifiers used as labels for the columns. Fault classes may be defined as components, component groups, functions of components, individual cells within components, or component failure modes. Each cell in the table represents the fraction of faults in the class identified by the row label that causes the test-outcome identified by the column label to occur. In Fig. 8-1, the total probability that fault class FC1 is detected is the probability that test outcome TO1 or TO2 or TO3 occurs when that fault is present. As will be noted, the greatest accuracy in analysis occurs when the fault classes and test outcomes are all highly specific.

A more convenient, but less accurate, analysis may be performed by treating the test outcomes and fault classes hierarchically as shown in Fig. 8-2. In Fig. 8-2, a single cell in Fig. 8-1 is considered. The test outcome and/or the fault-class of a single cell in Fig. 8-1 are

Test Outcomes / Fault Classes	TO1	TO2	TO3	Failure Rates
FC1				
FC2				
FC3				

Figure 8-1. Information Needed to Perform Test-Effectiveness Analysis

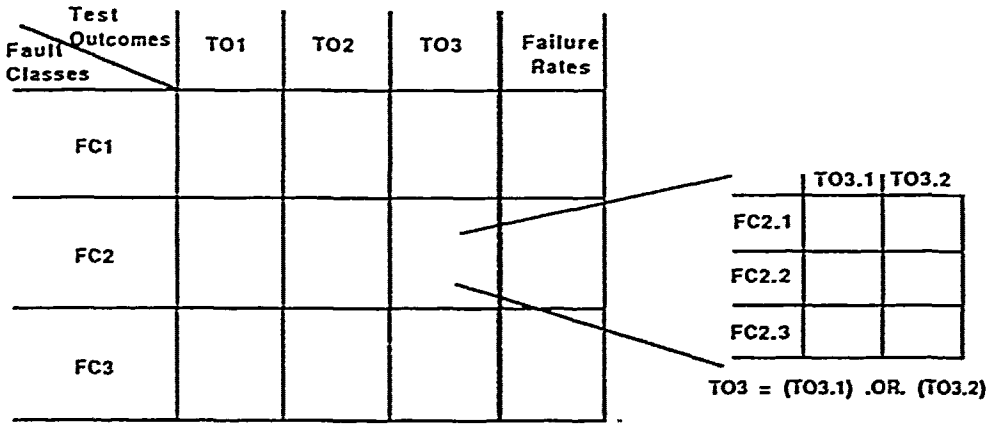


Figure 8-2. Hierarchical Analysis

further decomposed and analyzed to determine the entry in Fig. 8-2. These results are then summarized in the entry in Fig. 8-1, this summarization leads to inaccuracy, compared with performing and reporting the analysis at the lower level. Note that fault-class failure rates must be obtained using MIL-HDBK 217 (or a similar source) and engineering judgement. (An alternative source for component failure rate is the increasingly common manufacturer-warranted figure.) Similarly, fault accountability data must be obtained through the use of an FMEA/FMECA and/or engineering judgement.

From this information, the computation of two performance measures, FFD and FFI, are of interest. In both methods presented in this appendix, the results are more precise when the fault class definitions are of minimal size. In the limit, where *all* faults in each class either cause each test outcome or do not cause it (i.e., the fractions in item 3. above are either 0 or 1), both methods result in *precise* values for FFD and FFI (if not *accurate* because of approximations to failure rates). Both methods are closely related to the problem of determining signal probabilities for use in statistical fault grading of test patterns (Agrawal and Seth, 1988).

Subsection A.2 presents an application of the probabilistic approach to calculation of FFD for a modern military electronic unit.

8.2 INTRODUCTION AND MOTIVATION

Quantification of diagnostic performance measures such as FFD and FFI from basic information about a diagnostic capability's design characteristics has been, and continues to be, an important step in the design and validation of diagnostic systems. Today, there are many so-called "testability analysis" methods that can be used for this purpose (e.g., see Agrawal and Mercier (1985), Binnendyk (1989), Binnendyk and Remeis (1989), Huisman (1988), and Simpson et al. (1986)). One way to distinguish between these methods is by the design properties that must be specified in order for the analysis to be carried out.

For example, determination of FFD by s/a fault simulation requires a gate-level description of the primary digital circuit *and* a list of the test patterns that will be applied to the circuit. Coverage is determined by either exhaustive fault simulation or by random sampling, when the number of faults is too large. For modern electronic equipment, this method is infeasible since gate level descriptions of proprietary devices may be unavailable and since computation of board-level performance measures would require that many high-density VLSI device models be captured on a single CAD system.

In MIL-STD-2165, a system-level approach is taken. Computation of FFD requires a list of failure rates for each device on a board (or board in a box, or subsystem within a system) and a

specification of FFD for that device (or board or subsystem). The latter is determined by decomposing the device (or board, etc.) into various fault classes, assigning failure rates to each fault class and then adding up the failure rates for those fault classes that are detected by the diagnostic capability. Computation of FFI (or "fault resolution" as it is called in MIL-STD-2165) requires a fault dictionary, which amounts to a list of fault classes, their failure rates, and a fault signature associated with each fault class. All faults with the same signature are indistinguishable, thereby allowing computation of FFI.

There are several problems in applying this method to modern diagnostic capabilities. In computing FFD, for example, we must be able to decompose a device into mutually exclusive fault classes and specify which fault classes are detected and which are not detected. The first step in this process is easy. Simply decompose the device (or board) into regions that are and are not exercised by the diagnostic capability. However, we can not assume that all faults contained in each region that is exercised will be detected (if we did, a highly optimistic estimate of FFD would result). To determine what fraction of faults in each exercised region is detected requires a further decomposition. For example, testing a multiplier unit within a CPU by multiplying several numbers and checking the results is a common test technique. However, such an approach frequently will not detect all multiplier faults. To determine how many single stuck-at faults are detected and how many are undetected would require fault simulation, which in-turn requires at least a gate-level model, which may or may not be available or practically constructed. Furthermore, single stuck-at faults may only be a subset of the faults that may be of interest. However, methods to determine coverage of other fault types (besides single stuck-at) are still in the research stage.

Another problem is that modern diagnostic techniques make use of functional testing and overlay strategies. In these strategies, we may have one test (or test sequence) that is specifically designed to detect faults in a certain device (or region of a device or board). Other tests, designed to detect other faults, however, may also exercise the first device as well. This admits the possibility that more faults could be detected (ignoring these makes the FFD estimate inaccurate,

but pessimistic). As a result, the test-sequences that need to be evaluated include *all* tests that exercise a particular device. Thus, even if a diagnostic capability uses an established procedure for testing a device (and hence its fault coverage is known), all additional tests that merely exercise the device should also be analyzed for fault coverage to produce accurate estimates of FFD.

In computing FFI, MIL-STD-2165 requires that "fault signatures" for each fault class be established. In systems using functional testing and overlay strategies this is a difficult task. For example, consider the multiplier test example above. Suppose in addition to this test sequence, we also have a functional test sequence that exercises the multiplier, as well as other devices in the system. Now, how can the fault signature for the multiplier be determined? With two test sequences there are four possible signatures (both tests fail, one test fails, and both don't fail). Since neither test sequence detects all multiplier faults (as discussed above), it is conceivable that *all* of the four fault signatures could be produced when some multiplier fault occurs. Some faults in the multiplier will be detected by both test sequences. Some may be detected by the initial multiplier test and not by the functional test. Similarly, there may be some detected by the functional test, but not by the initial multiplier test, and some that are not detected by either test. Thus we see that all fault signatures are possible for faults that occur in the multiplier unit. To use the MIL-STD-2165 methodology, we must list the fraction of multiplier faults that cause each of the four possible fault signatures to occur. How can this be done? The only alternative is to decompose the multiplier further into fault classes that result in each signature. As with the fault detection problem, this can only be done, in general, by fault simulation, which has all of the drawbacks cited above.

What the above discussion indicates is that application of the MIL-STD-2165 methodology ultimately requires full fault simulation of the entire equipment under each test stimulus condition provided by the diagnostic capability. With today's CAD technology and the increasing use of proprietary designs, this is not practical in large systems containing many VLSI devices.

These problems have motivated consideration of an alternative analysis approach. In this approach, we assume that the designer can specify or estimate the fraction of faults in each of several mutually exclusive fault classes that would cause each test (or test-sequence) executed in

the diagnostic capability to fail (or in multi-outcome cases, that causes each test outcome to occur). More precisely, suppose we subdivide the equipment under consideration into sets of mutually exclusive fault classes, F_i. The diagnostic capability is subdivided into individual test or test sequence outcomes, t_j. Let T_J be the set of faults that cause outcome t_j. The approaches developed in this appendix require only that the fraction of F_i faults that is also in T_J (called f_{ij}) are specified. The way in which the equipment and diagnostic capability are subdivided is left to the analyst. However, it should be clear that the deeper one subdivides both equipment and diagnostics, the more precise the analysis can be. In fact in the limit, where the f_{ij} are either 0 or 1, the method reduces to the MIL-STD-2165 approach.

The advantage of this approach is that we avoid the extensive effort associated with correct application of the MIL-STD-2165 approach without compromising accuracy. Below we introduce the notation to be used in developing this new approach. Two techniques are then described. In subsection 8.2 a probabilistic approach that relies on an assumption of test independence is described. Since this assumption is often not valid, a new method that allows computation of bounds on FFD and FFI is developed in subsection 8.3.

NOTATION

d_i:	diagnostic outcome i, a boolean variable with $d_i = 1$ implying that diagnostic outcome i is "true" (i.e., it has or will be generated by the diagnostic capability). Standard boolean operators will be used throughout: addition to denote union or logical "or" and multiplication to denote intersection or logical "and". d is also used without a subscript to represent the diagnostic outcome when only one is being considered.
f_i.	an often-used variable denoting fraction of faults, defined wherever it is used.
f_{ij}:	the fraction of faults in F_i that causes $t_j = 1$. The f_{ij} for all i and j are the raw accountability data from which the analysis starts. Note that $f_{ij} = S(T_j F_i) / S(F_i)$ assuming that all physical faults in the two sets are equally likely to occur. Also note that this must be calculated using an FMEA and/or engineering judgement.
F_i:	fault class i, which should be interpreted as a set of physical faults.
max:	the maximization operator on a set—the result is the largest element in the set.
min:	the minimization operator on a set—the result is the smallest element in the set.

$S(x)$:	the cardinality of (total number of elements in) the set x.
t_i:	refers to outcome of test i. We will consider t_i to be a boolean variable (it takes on values of zero or one; 1=the outcome t_i has or will be generated, 0=it won't). Note that for pass/fail tests, only one outcome need be considered. In general, if there are n outcomes of a specific test, only n-1 need to be considered; and usually the outcome corresponding to the one produced when no failure is present (e.g., pass) is the one not considered.
T_i:	the set of physical faults that can cause $t_i = 1$ when they occur individually.
$p(x)$:	unconditional probability of the occurrence of event x.
$p(x\|y)$:	probability of event x given that event y has occurred (conditional probability).
$[x_L,x_H]$:	we will use the shorthand notation with square brackets, $x=[x_L,x_H]$, to denote that the scalar variable x is bounded from below by x_L and from above by x_H, i.e., $x_L \leq x \leq x_H$.
$\bar{z}$:	the complement of boolean variable or set z.
λ_i:	failure rate associated with F_i. Note, this must be calculated using MIL-HDBK-217 (or a similar source) and engineering judgement.
λ_T:	total system failure rate (may be the sum of all λ_i if all relevant fault classes have been enumerated). Again, MIL-HDBK-217 and engineering judgement must be applied.

8.3 A PROBABILISTIC METHODOLOGY

8.3.1 An Introductory Example

To introduce the concepts in this section, consider the following problem. Suppose we are told that fault class F_1 is isolated whenever test-1 fails, or whenever test-2 fails and test-3 passes. Let the boolean variable "d" denote the single *diagnostic outcome* "F_1 is isolated." Let t_i be the boolean variable associated with *test outcome* "test-i fails." Then,

$$d = t_1 + t_2 \bar{t_3}$$

Now, as part of an FFI calculation we might be interested in determining the fraction of F_1 faults that actually cause the diagnostic outcome (d=1) to occur (note we are *not* assuming as in MIL-STD-2165 that the callout of F_1 guarantees that only F_1 faults (and no others) result in this diagnostic outcome). The raw accountability data (the f_{ij}) may be organized in a table as shown below (Table 8-1). For example, we might have $f_{11} = 0.90$, $f_{12} = 0.80$, and $f_{13} = 0.05$. Notice that the sum of the f_{ij} over all j need not be equal to one.

TABLE 8-1. RAW FAULT ACCOUNTABILITY DATA

	t_1	t_2	t_3
F_1	f_{11}	f_{12}	f_{13}

In the probabilistic methodology, we treat all fractions as probabilities. That is, the fraction of F_1 faults causing d = 1 is interpreted as the probability that d=1 given that a fault in F_1 has occurred, or formally $p(d=1 \mid F_1)$. The f_{ij} are interpreted as the probability that test outcome t_j occurs given a fault in F_i has occurred, or $p(t_j = 1 \mid F_i)$. To compute $p(d=1 \mid F_1)$ from the $p(t_i = 1 \mid F_1)$ simply involves repeated application of standard probability axioms and theorems (e.g., see Papoulis (1965)). For clarity, we repeat the important properties here. (Note that we use the notation "a" and "b" to represent both events and the boolean indicator variable for those events; also the notation p(x) means p(x=1); this will be used in general in the remainder of this section.)

Property 1: $$p(\bar{a}) = 1 - p(a).$$

Property 2: $$p(a + b) = p(a) + p(b) - p(ab).$$

Property 3: If events a and b are *independent* then $p(ab) = p(a) \times p(b)$.

Now, we can apply these axioms and properties to the problem of computing $p(d \mid F_1)$ from the $p(t_i \mid F_1)$ and the decision logic of Eq. 8-1. The following steps provide the desired result. (Note that all probabilities are conditioned on F_1, so we drop the conditional notation for the time being.)

a) $p(\bar{t_3}) = 1 - p(t_3)$

b) $p(t_2 \bar{t_3}) = p(t_2) (1 - p(t_3))$ by assuming that t_2 and t_3 are independent

c) $p(d) = p (t_1 + t_2 \bar{t_3}) = p(t_1) + p(t_2 \bar{t_3}) - p(t_1)p(t_2 \bar{t_3})$ assuming that t_1 is independent of t_2 and t_3. Using the result of step (b) and switching back to f_{ij} notation, we have

$$p(d=i \mid F_1) = f_{11} + f_{12}(1 - f_{13}) - f_{11}f_{12}(1 - f_{13})$$

For the numerical values $f_{11} = .90$, $f_{12} = .80$ and $f_{13} = .05$ we have $p(d = 1 \mid F_1) = 0.976$ (.98 to two significant figures).

8.3.2 A More Complex Example

Suppose now that we consider the problem of three tests and four fault classes, with hypothetical fault accountability data (f_{ij}) as shown in Table 8-2. Suppose further that there are four diagnostic outcomes of interest (one detection and three isolation) that are defined by:

d_0 = fault detected,

d_1 = fault is in RU A (fault class 1, F_1),

d_2 = fault is in RU B (fault class 2, F_2), and

d_3 = fault is in RU C (fault class 3 or 4, F_3 or F_4).

The decision logic defining each of these outcomes is assumed to be:

$$d_0 = t_1 + t_2 + t_3$$

$$d_1 = t_1$$

$$d_2 = t_2\bar{t}_1$$

$$d_3 = t_3.$$

We now wish to compute FFD and and FFI.

TABLE 8-2. HYPOTHETICAL FAULT ACCOUNTABILITY DATA

	t_1	t_2	t_3
F_1	0.95	0.25	0.00
F_2	0.10	0.95	0.01
F_3	0.00	0.10	0.95
F_4	0.00	0.15	0.90

PROCEDURE FOR COMPUTING FFD

Step 1: Compute $p(d_0=1 | F_1) = f_1$, the fraction of F_1 faults causing $d_0 = 1$.

a) $f_{11} = 0.95$, $f_{12} = 0.25$, $f_{13} = 0.00$

b) $p(t_1 + t_2 | F_1) = 0.95 + 0.25 - (.95)(.25) = .9625$

c) $p ((t_1 + t_2) + t_3 | F_1) = .9625 + 0 - (.9625)(0) = .9625$ (.96 to two significant figures).

Step 2: Compute $p(d_0=1 | F_2) = f_2 = .95545$ (.96 to two significant figures).

Step 3: Compute $f_3 = .955$ (.96 to two significant figures).

Step 4: Compute $f_4 = .915$ (.92 to two significant figures).

Step 5: Assume that all fault classes are disjoint, then

$$FFD = \frac{\sum_{i=1}^{4} \lambda_i f_i}{\sum_{i=1}^{4} \lambda_i}$$

and numerically, FFD = 3.80/4 = .95 (since all failure rates are assumed to be equal). Notice that this equation is the same as the MIL-STD-2165 equation for system level test effectiveness (para. 50.7.5, Appendix A). The difference here is that to get the f_i we need first to analyze the raw fault accountability data in terms of the defined decision logic. While this procedure therefore requires more effort in general, it provides a firmer basis for determining the f_i when test coverage overlap exists. Notice that in the case where no test coverage overlap is present (e.g., $f_{ij} = 0$ for all $i \neq j$), this method reduces to the MIL-STD-2165 procedure.

PROCEDURE FOR COMPUTING FFI

For FFI, assume that groups of size one are of interest. Since all diagnostic outcomes result in single RU ambiguity groups all outcomes are considered here. (In general we would not consider those outcomes resulting in RU ambiguity groups of size larger than one).

Case 1: Fraction of All Faults Isolated

Here we interpret FFI strictly as the fraction of *all* faults correctly isolated to groups of size one. In Case 2 we treat the case where FFI refers to the fraction of *detected* faults correctly isolated to groups of size one.

Step 1: Compute f_1, the fraction of F_1 faults that cause correct isolation to RU ambiguity groups of size one. If an F_1 fault is present, then correct isolation implies $d_1 = 1$, which has a single RU ambiguity group. Thus $f_1 = p(d_1 = 1 \mid F_1)$. Since $d_1 = t_1$, $f_1 = 0.95$ from Table 8-2.

Step 2: Compute f_2, the fraction of F_2 faults that cause correct isolation to RU ambiguity groups of size one. Since correct isolation for F_2 faults is equivalent to $d_2 = 1$, $f_2 = p(d_2 \mid F_2)$. Assuming that t_2 and t_1 are independent and applying property 1, we get $f_2 = .95$ $(1 - .10) = .855$ (.86 to two significant figures).

Step 3: $f_3 = p(d_3 = 1 \mid F_3) = 0.95$

Step 4: $f_4 = p(d_3 = 1 \mid F_4) = 0.90$

Step 5: Assuming disjoint fault classes and equal failure rates for each of the four classes, then FFI = (.95 + .86 + .95 + .90) / 4 = .915 (0.92 to two significant figures).

Contrast this approach now with MIL-STD-2165 (para. 50.7.3.2, Appendix A). In that procedure, it is assumed that a fault dictionary exists. In the notation used here, a fault dictionary implies that the f_{ij} are either 0 or 1. As we have discussed, such accountability data can only be achieved if the fault classes are small enough (e.g., individual physical faults). In modern equipment, however, it is generally not possible to create such fault classes (e.g., every stuck-at fault in a combinational logic circuit would have to be considered a separate fault class). When the fault accountability table contains only 0 and 1 entries, the method discussed here reduces to the 2165 method. In such a case, no independence assumptions need to be made, and therefore the result is more precise.

Since FFI = .92, then 8 percent of all faults are either cause an incorrect isolation outcome or are not detected or both. A question naturally arises here: what is the percentage of all faults that cause an incorrect isolation outcome?

PROCEDURE FOR COMPUTING FFI_w

The procedure for computing the fraction of all faults that are incorrectly isolated, FFI_w, is similar to the procedure for FFI (all failure rates assumed equal). For our example:

Step 1: Compute f_1, the fraction of F_1 faults that can cause incorrect isolation. Since correct isolation is $d_1 = 1$, incorrect isolation is defined by $d_2 + d_3 = 1$. Therefore $f_1 = p(d_2 + d_3 \mid F_1)$. Using the definitions of d_i and the assumption that the t_i are independent we have

$$f_1 = f_{13} + f_{12}(1-f_{11}) - f_{13}f_{12}(1-f_{11}) = .0125 \text{ (0.01 to two significant figures)}$$

Step 2: $f_2 = p(d_1 + d_3 \mid F_2) = f_{21} + f_{23} - f_{21}f_{23} = 0.109$ (0.11 to two significant figures)

Step 3: $f_3 = p(d_2 + d_1 \mid F_3) = p(t_1 + \overline{t_1} t_2 \mid F_3)$.

To evaluate this expression, we have a slightly more complex problem. In the previous examples, the assumptions of test independence allowed us to apply iteratively the properties of probability. In this case, however, we are faced with computing a probability of the form p(a + b) where a and b are not independent because both a and b depend on t_1. In general, the calculation of probabilities for arbitrary boolean expressions such as the one in Step 3 is a very difficult problem when the expression is large (it is, in fact, NP-complete (Papadimitriou and Steiglitz, 1982)). However, for small expressions (this will be the case for many practical systems), the following procedure can be used:

Substep 3.1: Construct the truth table for the relevant expression. In this case we have

t_1	t_2	$t_1 + \overline{t_1} t_2$
0	0	0
0	1	1
1	0	1
1	1	1

Substep 3.2: For each case in which the desired expression is equal to one, compute the fraction of relevant faults (in this case F_3) that could cause the combination of test outcomes represented by that row in the truth table. In this case, assuming independence, we have

t_1	t_2	Conditional Prob
0	1	(1.0 - 0.0) (0.1) = .1
1	0	(0.0) (1.0 - 0.1) = 0.0
1	1	(0.0) (0.1) = 0.0

where for each row the third column is calculated as the product of the probability (given that F_3 is true) of the indicated value of t_1 (column one) times the probability (given that F_3 is true) of the indicated value of t_2 (column two).

Substep 3.3: Since each row in the truth table is mutually exclusive with all other rows, add up the results of substep 3.2. In this case we get $f_3 = 0.1$.

Note that we could also have obtained this result by recognizing that the t_1 and $\overline{t_1}\, t_2$ are mutually exclusive so that $f_3 = 0.0 + (1.0 - 0.0)\,(0.1) = 0.1$ by Property 2 with p(ab)=0. Since the boolean expressions representing fault isolation decisions are usually short, either the procedure outlined in Step 3 may be followed, or manual application of the properties of probability for the relevant expression may be effective.

Step 4: $f_4 = p(\, d_2 + d_1 \mid F_4) = p(\, t_1 + \overline{t_1}\, t_2 \mid F_4)$

Following the steps outlined in Step 3, we have

$$f_4 = (1.0 - 0.0)\,(0.15) + (0.0)\,(1.0 - 0.15) + (0.0)\,(0.15) = 0.15.$$

Step 5: Assuming disjoint fault classes and equal failure rates as in the FFI calculation, we have

$$FFI_w = (0.01 + 0.11 + 0.10 + 0.15) / 4 = 0.0925 \text{ (0.09 to two significant figures).}$$

Notice now that $FFI + FFI_w > 1$ in this example. This is a real effect (not due to rounding) because the d_i in this case are not mutually exclusive (i.e., d_1 and d_3 can both be 1 at the same time, as can d_2 and d_3). As a result, some faults may cause both an incorrect and a correct isolation decision (e.g., an F_2 fault can cause $t_1 = 0$, $t_2 = 1$, and $t_3 = 1$, resulting in d_2 and d_3 equal to one; one correct isolation (d_2) and one incorrect (d_3)). In diagnostic systems that are organized in a fault-isolation tree (e.g., a TPS), this will not happen because the isolation outcomes are mutually exclusive.

As an aside, suppose we change the decision logic so that $d_i = t_i$ for i=1, 2, 3. Recalculating FFI and FFI_w using the above procedures we get FFI = 0.94 and FFI_w = 0.15. Notice that while we achieved a 2 percent increase in FFI (from .92 to .94), FFI_w increased by more than 60 percent (from 0.09 to 0.15). This shows the importance of looking at both FFI and FFI_w in evaluating diagnostic performance (at least when isolation decision functions are not mutually exclusive). In the example, it is worth noting that the major increase in FFI_w came from incorrect isolation of F_1 faults to d_2 (as would be expected from Table 2). This kind of information can provide guidance for improving the overall effectiveness of the diagnostic capability (e.g., as measured by FFI/FFI_w).

Case 2: Fraction of Detected Faults Isolated

In general, if a fault is isolated (e.g., $d_i = 1$) then we say it is detected. That is $d_0 = d_1 + d_2 + \ldots$. (This was not the case in our example, although it is easy to show for that example that isolation implies detection there as well). Since this is generally true, we will not treat the case where isolation does not imply detection (this case is treated in subsection 8.4, however).

When isolation implies detection, then we are guaranteed that FFI ≤ FFD since the faults that are correctly isolated are a subset of all the faults that are isolated (correctly or not), which are, in turn) a subset of the faults that are detected (by assumption). As a result, the fraction of detected faults that is correctly isolated, FFI_d, is just given by

$$FFI_d = FFI / FFD$$

For our numerical example, FFI_d = .97 (to two significant figures).

8.3.3 Summary

In this subsection, we demonstrated how the properties of probability could be used to relate raw fault accountability data to overall diagnostic performance measures such as FFD and FFI. The method assumed that tests are independent since specification of test overlap in addition to the raw accountability data would be infeasible. Unfortunately, test independence is frequently a

poor assumption since faults are *not* assigned to test groups, T_i, at random. In the next subsection, we develop a new approach that provides upper and lower bounds on FFD and FFI by assuming minimal and maximal values for the size of the overlapping coverage sets.

8.4 A SET-THEORETIC METHODOLOGY

In this subsection we present a new method of computing the desired diagnostic performance measures from fault accountability data. The approach is an alternative to making the independence assumption, which can not be justified in many of the situations to which it is applied.

In this subsection the theoretical results that are needed to solve the problems of determining overall performance measures from raw fault accountability data are presented by example. A summary of the results is given at the end, along with a prescription for applying the results in determining FFD and FFI.

8.4.1 FFD for Dedicated Test

The trivial case assumes that there is a one-to-one mapping between t_i and F_i, that $t_i = 1$ only if a fault in F_i is present (i.e., only dedicated test functions), that F_iF_j is empty unless i=j (i.e., all fault classes are disjoint), and that detection occurs if at least one $t_i = 1$.

A fault accountability table for a three-fault class and three-test diagnostic capability is shown in Table 8-3. The diagnostic outcome for fault detection, d, is given by the boolean

TABLE 8-3. DEDICATED TEST SITUATION

	T_1	T_2	T_3
F_1	f_{11}	0	0
F_2	0	f_{22}	0
F_3	0	0	f_{33}

expression $d = t_1 + t_2 + t_3$, and FFD (for N fault classes) is computed by

$$FFD = \frac{\sum_{I=1}^{N} f_{ii}\lambda_i}{\lambda_T}$$

Note that we need not know the actual sizes of any F_i or T_i to compute FFD.

8.4.2 FFD with Overlapping Coverage.

Unfortunately, in most modern diagnostic capabilities the assumption that all test resources are dedicated test functions is likely to be invalid. This is because of the prevalent and effective use of functional testing and overlay BIT techniques and because of interactions between subsystems when they are electrically connected (as opposed to operating in isolation). (Even dedicated BIT may detect faults in modules other than those to which it is dedicated, (e.g., the power supply and loading faults).

Suppose we have two tests outcomes, t_1 and t_2, and a *single fault class*, F_1. We now show that the problem of determining FFD is not trivial when only f_{ij} is given and that the result depends strongly on the decision logic used to combine t_1 and t_2 to make an overall fault detect decision.

CASE 1: LOGICAL UNION

Assume that $d = t_1 + t_2$. By definition, FFD is precisely defined (since there is only one fault class) by

$$FFD = \frac{S(T_1 + T_2)}{S(F_1)}$$

Of course, neither the numerator nor the denominator of this expression can be directly determined. If we are given f_{11} and f_{12}, then by definition $S(T_1)/S(F_1) = f_{11}$ and $S(T_2)/S(F_1) = f_{12}$. From this we can write $FFD = f_{11} + f_{12} - S(T_1T_2)/S(F_1)$. Unfortunately, FFD still can't be computed exactly because there is no specification of the size of T_1T_2. The problem is shown in Fig. 8-3. Since it is

usually impractical to specify the size of the overlap between T_1 and T_2, we would like to be able to at least compute bounds on FFD that consider all possible overlap conditions. The following theorem gives optimal bounds for FFD in this case. Note that there is no failure-rate term since there is only one fault class.

Theorem 1: Assume that we have a single fault class F_1. Let $d = t_1 + t_2$ and the fraction of faults that make d=1 be called FFD. Then

$$FFD = [\ \max\ \{\ f_{11}, f_{12}\ \}\ ,\ \min\ \{\ 1, f_{11} + f_{12}\ \}\ ]$$

with these bounds being optimal (the upper bound is the least upper bound and the lower bound is the greatest lower bound).

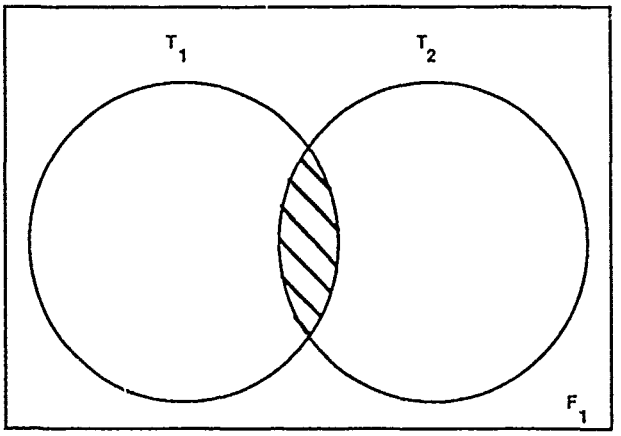


Figure 8-3. Overlapping Test Coverage

Proof: The least upper bound corresponds to the largest union of T_1 and T_2. The largest union corresponds to the smallest intersection, which is zero if $S(T_1) + S(T_2) \leq S(F_1)$, and in this case $FFD = f_{11} + f_{12}$. If $S(T_1) + S(T_2) \geq S(F_1)$, then the smallest intersection is the one that makes $S(T_1) + S(T_2) = S(F_1)$ and thus FFD = 1.

The greatest lower bound corresponds to the smallest union or largest intersection. If $S(T_1) \leq S(T_2)$ then the largest intersection is when T_1 is a subset of T_2 and $FFD = f_{12}$. If $S(T_1) \geq S(T_2)$ then the largest intersection is when T_2 is a subset of T_1 and $FFD = f_{11}$. QED

CASE 2: LOGICAL INTERSECTION

Again, assume a single fault class and two test outcomes, with a decision logic defined by $d = t_1 t_2$. This case might represent part of an overlay strategy designed to reduce false alarms. In this case FFD can be precisely computed from

$$FFD = \frac{S(T_1 T_2)}{S(F_1)}$$

As in Case 1 this formula is not useable when only f_{11} and f_{12} are known since the size of the intersection is unknown. Through arguments similar to that above, we can derive optimal bounds on FFD in this case.

Theorem 2: Given a single fault class F_1 and decision logic $d = t_1 t_2$. The fraction of faults that make d=1, FFD, obeys

$$FFD = [\ \max\{\ 0, f_{11} + f_{12} - 1\ \},\ \min\{\ f_{11}, f_{12}\ \}\ ]$$

with these bounds being optimal (the upper bound is the least upper bound and the lower bound is the greatest lower bound).

Proof: The least upper bound on FFD corresponds to the largest intersection of T_1 and T_2. The largest intersection occurs if either T_1 is a subset of T_2 or T_2 is a subset of T_1. If T_1 is a subset of T_2, then the size of the intersection is $S(T_1)$ and FFD is equal to f_{11}. If T_2 is a subset of T_1, then the size of the intersection is $S(T_2)$ and FFD is equal to f_{12}. The least upper bound is therefore the largest of f_{11} and f_{12}.

The greatest lower bound corresponds to the smallest intersection of T_1 and T_2. If $f_{12} + f_{11} \leq 1$, then the smallest intersection is when T_1 and T_2 are disjoint and the size of the intersection is therefore equal to 0. If $f_{11} + f_{12} > 1$, then the smallest intersection is bigger than zero. The smallest intersection is the one that makes the size of T_1+T_2 equal to 1, which is clearly equal to $f_{11} + f_{12} - 1$ (since the $S(T_1+T_2) = f_{11} + f_{12} - S(T_1T_2)$). QED

CASE 3: DECISION LOGIC HAVING COMPLEMENTS OF TEST OUTCOME VARIABLES

Theorem 3: Given one class of faults F_1 and the test outcome t_1 with fault set T_1. If the fraction of faults that make $t_1 = 1$ is f_{11}, then the fraction of faults that make $\overline{t_1} = 1$ is $1 - f_{11}$.

Proof: By definition $f_{11} = S(T_1) / S(F_1)$. The faults that make $\overline{t_1} = 1$ make $t_1 = 0$, and are those faults in $\overline{T_1}\, F_1$. The size of this set, $S(\overline{T_1}\, F_1)$, is clearly $S(F_1) - S(T_1)$. The theorem is proved by dividing $S(F_1) - S(T_1)$ by $S(F_1)$. QED

For example, consider $d = t_1 + \overline{t_2}$. Applying Theorems 1 and 3, we have

$$FFD = [\max \{ f_{11}, 1 - f_{12} \} , \min \{1, f_{11} + 1 - f_{12} \}]$$

In general, if $\overline{t_j}$ appears in the decision logic, then Theorems 1 and 2 apply with $1 - f_{ij}$ used everywhere in place of of f_{ij}.

CASE 4: GENERAL DECISION LOGIC

In principle the decision logic that relates a diagnostic outcome to a set of test outcomes may consist of an arbitrary boolean expression. To determine correctly the optimal bounds on the fraction of faults that cause the specified diagnostic outcome from the known coverages of the individual tests is a very difficult problem in general (as stated in subsection 8.3, this problem is NP-complete).

Consider a decision logic expression of the form

$$d = t_1 t_2 + \overline{t_2}$$

The truth table for d is shown in Table 8-4. The fraction of faults in the *single fault class* F_1 that cause d=1 will be called FFD. As in subsection 8.3.2, an algorithmic method can be employed to compute FFD based on the truth table. However in this case, where independence is not assumed, the algorithm is considerably more involved and is therefore omitted here. Suboptimal bounds, however, can be obtained by sequentially applying Theorems 1 through 3 to portions of the arbitrary decision logic expression.

TABLE 8-4. TRUTH TABLE FOR CASE 4

t_1	t_2	d
0	0	1
0	1	0
1	0	1
1	1	1

For example, bounds on FFD for the above decision logic expression would be computed by first computing bounds for the expressions t_1t_2 and $\overline{t_2}$ separately. Then bounds on d are computed using Theorem 1 to combine these two bounds. Let the fraction of faults causing $t_1t_2 = 1$ be called r. From Theorem 2,

$$r = [\ \max\ \{0, f_{11}+f_{12}-1\}\ ,\ \min\ \{f_{11}, f_{12}\}\ ].$$

Similarly, the fraction of faults causing $\overline{t_2} = 1$ will be called s. From Theorem 3, $s = 1-f_{12}$, or using similar notation as for r,

$$s = [\ 1 - f_{12},\ 1 - f_{12}\ ]$$

To get FFD, we now apply Theorem 1 recognizing that the decision logic is of the form $d = a + b$, with the coverage of the boolean variable 'a' given by 'r' and coverage of the boolean 'b' given by 's'. Thus, formally we have

$$FFD = [\ \max\ \{r, s\}\ ,\ \min\ \{1, r+s\}\ ]$$

Now, r and s are not known precisely so we must combine their ranges in some way. This is done using the following theorem, which is stated without proof.

Theorem 4: Given two numbers r and s that are imprecisely known but bounded such that $r = [r_L, r_H]$ and $s = [s_L, s_H]$:

a) $\max \{ r , s \} = [\max \{ r_L, s_L \} , \max \{ r_H, s_H \}]$

b) $\min \{ r , s \} = [\min \{ r_L , s_L \}, \min \{ r_H , s_H \}]$

c) $(r + s) = [r_L + s_L , r_H + s_H]$

d) $(r - s) = [r_L - s_H , r_H - s_L]$

e) $(r/s) = [r_L/s_H , \min\{r_H/s_L , 1\}]$

Thus, for this example, we have

$$FFD = [\max \{ r_L , s_L \} , \min \{ r_H+s_H , 1 \}]$$

with

$$r_L = \max \{ 0 , f_{11}+f_{12}-1 \},$$

$$r_H = \min \{ f_{11} , f_{12} \}, \text{ and}$$

$$s_L = s_H = 1 - f_{12}.$$

With the values $f_{11} = 0.90$ and $f_{12} = 0.60$, then FFD = [0.5, 1].

It must be mentioned again that when bounds for the fraction of faults that cause d = 1 are computed by iterative use of Theorems 1 through 4, as was done above, there is no guarantee that the bounds are optimal. This is because Theorems 1 and 2 assume that no knowledge of the overlap between the coverage of the two component decision logic portions is available. This is true when the two decision logic portions are individual test outcomes. However, when the two decision logic portions are boolean functions of the same test outcomes, then some knowledge of overlap may exist.

For example, in the above calculations we applied Theorem 1 for d = a + b. Now, Theorem 1 assumes that S(ab) is unknown. But since $a = t_1 t_2$ and $b = \overline{t_2}$, we know that S(ab) = 0. Thus, bounds in this case may actually be determined by FFD = r+s. Using Theorem 4c and the bounds on r and s given above,

$$FFD = [\ 1 - f_{12} + \max\{0, f_{11}+f_{12}-1\}\ ,\ 1 - f_{12} + \min\{f_{11}, f_{12}\}\ ].$$

In the numerical example with $f_{11} = 0.90$ and $f_{12} = 0.60$, the bounds are FFD = [0.90, 1.0]. These happen to be the optimal bounds for this case. The following theorem applies to unions of mutually-exclusive decision logic portions, and in general will give tighter bounds than the iterative application of Theorems 1 through 4 (although they are not necessarily optimal either).

Theorem 5: Given a *single fault class* F_1 and a logical decision function, d, in the form

$$d = c_1 + c_2 + \ldots + c_N$$

$$c_i = \prod_j s_j$$

with s_j = either t_k or $\overline{t_k}$ and $c_i c_j = 0$ for all $i \neq j$ (i.e., the decision logic portions c_i are mutually exclusive). Let r_i = the fraction of F_1 faults causing $c_i = 1$. Then the fraction of F_1 faults causing d=1, FFD, is

$$FFD = \sum_i r_i \qquad \Diamond$$

or when the r_i are imprecisely known but are bounded (i.e., $r_i = [\ L_i\ ,\ H_i\ ]$), then

$$FFD = [\sum_i L_i\ ,\ \sum_i H_i\ ] \qquad \Diamond$$

Note here that these bounds may not be optimal either since in general, the individual decision logic portions c_i may be functions of the same test outcomes. As a result, a non-obvious coupling of the bounds on each decision logic portion may occur.

While the bounds computed through iterative application of Theorems 1 through 4 may be loose in general, we note here that the decision logic of interest in many applications takes a form in which tighter bounds can usually be obtained. For example, a common decision function for detection of faults is the union of a set of tests (e.g., the go-chain of a TPS). Theorem 1 is readily extended to provide optimal bounds for the union of more than two test outcomes:

Theorem 1a: Assume that we have a single fault class F_1. Let $d = t_1 + t_2 + t_3 + \ldots t_N$ and the fraction of faults that make d=1 be called FFD. Then

$$FFD = [\ \max\ \{\ f_{11}, f_{12}, f_{13}, \ldots, f_{1N}\ \}\ ,\ \min\ \{\ 1, f_{11} + f_{12} + f_{13} + \ldots + f_{1N}\ \}\ ]$$

with these bounds being optimal. ◊

Similarly, for decision logic that is structured in a decision tree (e.g., a TPS), each branch of the tree represents a multiple intersection decision logic. In computing FFI, we will want to find all branches that result in removals/replacements of specific size and determine what fraction of faults contained in the correct fault classes result in each diagnostic outcome (each branch). In this case Theorem 2 is readily extended:

Theorem 2a: Assume that we have a single fault class F_1. Let $d = t_1t_2t_3 \ldots t_N$, and the fraction of faults that make d=1 be called FFD. Then

$$FFD = [\ \max\ \{\ 0, f_{11} + f_{12} + f_{13} + \ldots + f_{1N} - (N-1)\ \}\ ,\ \min\ \{\ f_{11}, f_{12}, f_{13}, \ldots, f_{1N}\ \}\ ]$$

with these bounds being optimal. ◊

8.4.3 FFD with Disjoint and Nondisjoint Fault Classes of Overlapping Coverage

CASE 1: DISJOINT FAULT CLASSES WITH OVERLAPPING COVERAGE

In Example 2 we dealt with a single fault class. Disjoint fault classes with dedicated test functions were discussed in Example 1. The case of disjoint fault classes with overlapping coverage is easily handled by applying Theorem 4c to the formula given in Example 1. In particular, suppose we are given the f_{ij} and a specific decision logic and, using the examples discussed above, we compute *for each fault class* the fraction of faults in that class that causes the diagnostic outcome. Call these fractions f_i. In general, only bounds on each f_i will be known, i.e., $f_i = [\ L_i\ ,\ H_i\ ]$. If all fault classes are disjoint, then it is easy to show that

$$FFD = \left[\frac{\sum_{i=1}^{N} L_i \lambda_i}{\lambda_T} \cdot \frac{\sum_{i=1}^{N} H_i \lambda_i}{\lambda_T} \right]$$

Disjoint fault classes can be ensured when performing a validation analysis by decomposing the system along physical boundaries (note that this is also a good way of ensuring that failure rates can be estimated using MIL-HDBK-217 or a similar source). Sometimes, however, it may be useful to decompose the faults of a specific device according to the functions the device performs or the various specific modes of failure that are known. Now, fault classes formed in this way are seldom disjoint since the physical mechanisms that cause faulty behavior may cause faults in more than one class. For example, a bond-wire failure will affect all functions that require use of the pin connected to the faulty wire. In general, nondisjoint fault classes are likely to arise anytime a functional, rather than physical, decomposition of a UUT is performed.

CASE 2: NONDISJOINT FAULT CLASSES WITH OVERLAPPING COVERAGE

When nondisjoint fault classes are involved, the above equation for FFD does not hold, and the following theorem must be used.

Theorem 6: Given two nondisjoint fault classes F_1 and F_2 with failure rates λ_1 and λ_2, respectively, and given f_1 and f_2, the fractions of faults in each class that cause the relevant diagnostic outcome, d = 1, then the total fraction of faults that causes d = 1, FFD, is given by

$$FFD = \left[\frac{N_{max}}{\lambda_1 + \lambda_2 - N_{min}} \cdot \min \left\{ \frac{\lambda_1 f_1 + \lambda_2 f_2}{\lambda_{max}}, 1 \right\} \right]$$

where

$$N_{max} = \max \{ \lambda_1 f_1, \lambda_2 f_2 \},$$

$$N_{min} = \min \{ \lambda_1 f_1, \lambda_2 f_2 \}, \text{ and}$$

$$\lambda_{max} = \max \{ \lambda_1, \lambda_2 \}.$$

Proof: If a given test, i, covers f_i of the faults in F_i, then it covers $\lambda_i f_i$ faults (per time unit), all in F_i. Now consider two tests, 1 and 2, which cover f_1 faults in F_1 and f_2 faults in F_2, respectively (see Fig 8-4). The question is, given that F_1 and F_2 are not disjoint, what are the largest and smallest fractions of total faults that are covered by the two tests? This, of course, depends upon the size of the intersection of F_1 and F_2 *and* how many faults in that intersection are covered by both tests.

Maximum Coverage. The maximum coverage is obtained when the intersection between F_1 and F_2 is maximum and there are no faults common to T_1 and T_2 (since this results in the smallest total set size and the largest cover). The smallest total set of faults occurs when either F_1 is a subset of F_2 or vice versa, depending upon which fault class is larger. Therefore, the smallest total number of faults is $\lambda_{max} = \max \{ \lambda_1, \lambda_2 \}$. The largest cover occurs when the intersection (or overlap)

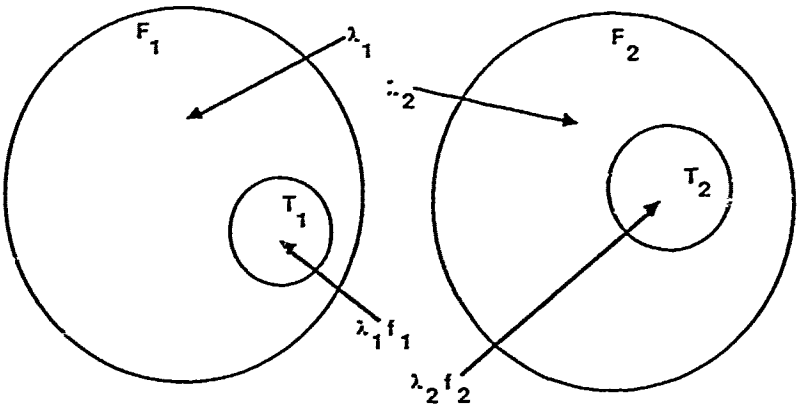


Figure 8-4. Fault Coverage of Tests 1 and 2

between T_1 and T_2 is zero, in which case the total number of faults covered (per time unit) is $\lambda_1 f_1 + \lambda_2 f_2$. Therefore, the maximum fraction of total faults covered is

$$\frac{\lambda_1 f_1 + \lambda_2 f_2}{\lambda_{max}},$$

which can be greater than one (clearly this bound is not optimal).

Minimum Coverage. The minimum *number* of faults covered occurs when either T_1 is a subset of T_2 or vice versa, depending upon which set is larger. Since there are $\lambda_i f_i$ faults (per time unit) in T_i, the minimum number of faults covered is $N_{max} = \max \{ \lambda_1 f_1, \lambda_2 f_2 \}$. Now, in order for T_1 to be a subset of T_2 (or vice versa) there must be overlap between F_1 and F_2 as shown in Fig. 8-5. The minimum coverage occurs when $S(F_1+F_2)$ is maximum or $S(F_1F_2)$ is minimum. From the figure it is clear that the minimum of $S(F_1F_2)$ when T_1 is a subset of T_2 is $S(F_1F_2)=\lambda_1 f_1$, and when T_2 is a subset of T_1 it is $S(F_1F_2)=\lambda_2 f_2$. In general the minimum of $S(F_1F_2)$ is $N_{min} = \min \{ \lambda_1 f_1, \lambda_2 f_2 \}$. The resulting fault universe has size $S(F_1F_2)=\lambda_1 + \lambda_2 - N_{min}$. Therefore, the minimum fraction of total faults covered is

$$\frac{N_{max}}{\lambda_1 + \lambda_2 - N_{min}}. \quad \text{QED}$$

In computing overall measures like FFD or FFI, Theorem 6 would be used after computing the fractions of each fault class that cause the relevant diagnostic outcome (i.e., after the f_i are

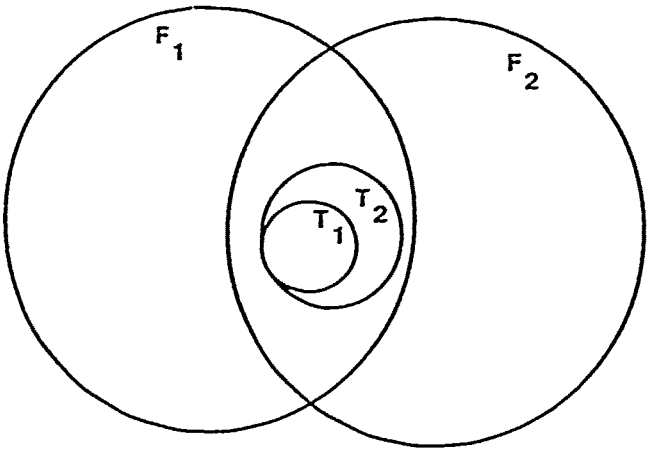


Figure 8-5. Overlapping Test Coverage

determined). Theorem 6 would be applied separately to *all groups* of fault classes that may be nondisjoint. These groups, along with all other disjoint fault classes, would then be combined in the disjoint fashion presented above.

8.4.4 FFI Calculations

This subsection presents no new theory. It simply demonstrates how the above calculations can be combined to estimate FFI.

First, recall that the definition of FFI is the fraction of lifetime faults that are correctly isolated to RU groups of specified size. Thus, all faults that are incorrectly isolated and all faults that result in RU ambiguity groups of larger than the specified size are not included in FFI. To compute FFI, we need to specify all of the diagnostic outcomes that result in RU ambiguity groups of the specified size. Then, for each of these outcomes, we need to define which fault classes are the correct fault classes (i.e., fall within those RU ambiguity groups). Finally, using the theory developed in the preceding examples, we must compute the fraction of each correct fault class that results in each diagnostic outcome, and then combine all correct fault classes of all outcomes.

CASE 1: FRACTION OF ALL FAULTS ISOLATED

Consider the fault coverage information given in Table 8-2 (see subsection 8.3.2). Suppose that the fault classes are *disjoint* with *equal* failure rates, and further suppose that the fault isolation logic consists of three separate outcomes:

d_1 = fault is in RU A (fault class 1, F_1),

d_2 = fault is in RU B (fault class 2, F_2), and

d_3 = fault is in RU C (fault class 3 or 4, F_3 or F_4)

The decision logic defining each of these outcomes is assumed to be

$$d_1 = t_1$$

$$d_2 = t_2\overline{t_1}$$

$$d_3 = t_3$$

In this example, we assume that FFI for RU groups of size one is of interest. Since all outcomes result in single RU ambiguity groups, all outcomes are considered here. In general we would not consider those outcomes resulting in RU ambiguity groups of size larger than one.

Step 1: Compute f_1, the fraction of F_1 faults that cause correct isolation to RU ambiguity groups of size one. If an F_1 fault is present, then correct isolation implies $d_1 = 1$, which has a single RU ambiguity group. Since $d_1 = t_1$,

$$f_1 = 0.95.$$

Step 2: Compute f_2, the fraction of F_2 faults that cause correct isolation. For F_2 faults, correct isolation is equivalent to $d_2 = 1$. Since $d_2 = t_2\overline{t_1}$, we use Theorems 2 and 3 to get

$$f_2 = [0.85, 0.90].$$

Step 3: Compute f_3. For F_3, correct isolation is equivalent to $d_3 = 1$. Thus

$$f_3 = 0.95.$$

Step 4: Compute f_4. For F_4, correct isolation is $d_3 = 1$. Therefore

$$f_4 = 0.90.$$

Step 5: Applying the result of Example 3, Case 1 (we assumed disjoint fault classes) and Theorem 4c,

FFI = (.95 + [.85, .90] + .95 + .90) / 4, or (to two significant figures)

FFI = [0.91, 0.93].

For comparison, recall that in subsection 8.3 for this same example but assuming *independent* tests, we calculated FFI = 0.92, which lies within the FFI bounds just calculated.

Another measure of interest is the fraction of faults incorrectly isolated (whether the decision logic results in RU ambiguity groups of specified size or not). To compute the fraction of faults incorrectly isolated (FFI_w), we follow similar steps.

Step 1. Compute f_1, the fraction of F_1 faults that can cause incorrect isolation. Since $d_1 = 1$ is the correct outcome for F_1 faults, $d_3 + d_2 = 1$ is incorrect. The fraction of F_1 faults

that results in $d_3 + d_2 = t_3 + t_2\bar{t}_1 = 1$ is computed using Theorems 1 through 4 as $f_1 = [0, 0.05]$

Step 2: Compute f_2, the fraction of F_2 faults that cause $d_1 + d_3 = t_1 + t_3 = 1$. Using Theorem 1, $f_2 = [.10, .11]$.

Step 3: f_3 = the fraction of F_3 faults that cause $d_1 + d_2 = t_1 + t_2\bar{t}_1 = 1$. Using Theorems 1 through 4 we have $f_3 = 0.10$.

Step 4: $f_4 = 0.15$.

Step 5: Since we assumed disjoint fault classes, $FFI_w = ([0, .05] + [.10, .11] + 0.10 + 0.15) / 4$ or

$$FFI_w = [\ 0.09, 0.10].$$

For comparison, recall that in subsection 8.3 for this same example but assuming *independent* tests, we calculated $FFI_w = 0.09$, which lies within the FFI_w bounds just calculated.

Notice that FFI + FFI_w may be greater than 1 according to this analysis. This is real (and not due to rounding) because, for this decision logic, there are some faults that can cause both correct and incorrect isolation (e.g., Table 8-2 admits the possibility of some F_2 faults that cause $t_1 = 0$, $t_2=1$, and $t_3=1$ resulting in $d_2= 1$ and $d_3= 1$; one correct and one incorrect outcome). In general, this type of result is possible whenever the decision logic for the various outcomes are not mutually exclusive, as is the case here (d_1 and d_3 can be true at the same time as can d_2 and d_3).

Finally, notice that if we change the decision logic to $d_i = t_i$ for i = 1, 2, 3 and recalculate FFI and FFI_w, we get FFI = 0.94, and FFI_w – [.09 .15]. Thus, with this different decision logic we achieve a slightly higher FFI, but FFI_w could be over 50 percent larger. This shows the importance of computing FFI_w in addition to FFI for evaluation of diagnostic fault isolation systems.

CASE 2: FRACTION OF DETECTED FAULTS ISOLATED

Above we computed the fraction of total faults correctly and incorrectly isolated to RU ambiguity groups of size one. To determine what percentage of the detected faults this corresponds to, we first need to define the set of detected faults. Usually, this set is specified only in

terms of the decision logic used to check for faults (e.g., the go-chain). Suppose, for example, that the go-chain fails (a fault is detected) if any t_i fails. That is, the detection outcome, d, is defined by

$$d = t_1 + t_2 + t_3.$$

For the coverage data of Table 8-2 (assuming disjoint fault classes and equal failure rates), FFD = [.94, 1.0]. For comparison, recall that in subsection 8.3 for this same example but assuming *independent* tests, we calculated FFD = 0.95, which lies within the FFD bounds just calculated.

To get bounds on the fraction of detected faults that are correctly (or incorrectly) isolated to RU ambiguity groups of the specified size, the following theorem may be applied.

Theorem 7: If the fraction of all faults that are correctly isolated is r and the fraction of all faults that are detected is called s, then the fraction of detected faults that are correctly isolated, FFI_d, is bounded by

$$FFI_d = [\ \max\{0, r+s-1\ \}, \min\{r, s\}] / s$$

Proof: The numerator in Theorem 7 consists of bounds on the fraction of total faults that are both detected and isolated and is obtained using Theorem 2. This is then divided by the fraction of faults that are detected. QED

For example using the results for FFI in Case 1 of Example 4, r = [.91, .93] and from above s = [.94, 1.0]. Theorems 4 and 7 can be used to get:

$$FFI_d = [0.85, 0.99].$$

(A general rule of thumb: when evaluating the lower (upper) bounds of expressions that are functions of uncertain quantities, use the lower (upper) bound of the appropriate quantity). Notice that even though both FFD and FFI are greater than 90 percent, Theorem 7 says that the fraction of detected faults that are correctly isolated may be as low as 85 percent. Again for comparison, recall

that in subsection 8.3 for this same example but assuming *independent* tests, we calculated FFI_d = 0.97, which lies within the FFI_d bounds just calculated.

While Theorem 7 provides a mechanism for computing bounds on FFI_d, these bounds may be far from optimal (they may be loose). This is because Theorem 7 assumes that the size of the set of faults that causes both detection and isolation is unknown (and therefore must be bounded using Theorem 2). However, this may in fact be untrue since the detection and isolation decision logic are frequently both functions of the same individual test outcomes. Therefore, a more accurate bound can be obtained if the decision logic for the intersection of detection and isolation decision functions is simplified.

In the above numerical example we first notice that using the actual definitions of d and d_i, i=i,2,3, all faults that cause an isolation outcome (correctly or not) are also detected. As a result we have (to two significant places):

$$FFI_d = [0.91, 0.93] / [0.94, 1.0] = [0.91, 0.99] .$$

This example is important since for most diagnostic systems all faults that cause an isolation outcome are also detected. In particular, it is true for any diagnostic logic that can be structured into a decision tree (e.g., a TPS) since all isolation outcomes "fall out of the go-chain" (i.e., are detected). In general, however, this result may not hold.

8.4.5 Summary

For reference, the primary results presented in this subsection on the *set-theoretic methodology* are repeated here.

LOGICAL UNION

Theorem 1a: For a single fault class F_1 with decision logic $d = t_1 + t_2 + t_3 + ... t_N$, the fraction of faults that make d=1, FFD obeys

$$FFD = [\max \{ f_{11}, f_{12}, f_{13}, ... , f_{1N} \} , \min \{ 1, f_{11} + f_{12} + f_{13} + ... + f_{1N} \}]$$

with these bounds being optimal.

LOGICAL INTERSECTION

Theorem 2a: For a single fault class F_1 with decision logic $d = t_1 t_2 t_3 \ldots t_N$, the fraction of faults that make d=1, FFD, obeys

$$FFD = [\ \max\ \{\ 0, f_{11} + f_{12} + f_{13} + \ldots + f_{1N} - (N-1)\ \}\ ,\ \min\ \{\ f_{11}, f_{12}, f_{13}, \ldots, f_{1N}\ \}\ ]$$

with these bounds being optimal.

COMPLEMENTS

Theorem 3: Given one class of faults F_1 and the test outcome t_1 with fault set T_1. If the fraction of faults that make $t_1 = 1$ is f_{11}, then the fraction of faults that make $\overline{t_1} = 1$ is $1 - f_{11}$.

When evaluating bounds for arbitrary boolean expressions involving the complement of a boolean variable or clause, $\bar{a}$, use $1 - f_a$ in the relevant formulas, where f_a is the fraction of faults causing $a = 1$.

ALGEBRAIC COMBINATIONS OF UNCERTAIN QUANTITIES

Theorem 4: Given two numbers r and s that are imprecisely known but bounded such that $r = [r_L, r_H]$ and $s = [s_L, s_H]$:

a) $\max\ \{\ r\ ,\ s\ \} = [\ \max\ \{\ r_L, s_L\ \}\ ,\ \max\ \{\ r_H, s_H\ \}\ ]$

b) $\min\ \{\ r\ ,\ s\ \} = [\ \min\ \{\ r_L\ ,\ s_L\ \}, \min\ \{\ r_H\ ,\ s_H\ \}\ ]$

c) $(r + s) = [\ r_L + s_L\ ,\ r_H + s_H\ ]$

d) $(r - s) = [\ r_L - s_H\ ,\ r_H - s_L\ ]$

e) $(r/s) = [\ r_L/s_H\ ,\ \min\{r_H/s_L\ ,\ 1\}\ ]$

A good rule of thumb is that when an "internal" algebraic expression appears in evaluating the lower (upper) bound of another expression, use the lower (upper) bound of the internal expression.

GENERAL DECISION LOGIC

A useful procedure for evaluating the fraction of faults that might cause an arbitrary expression to be true is to apply iteratively the previous results to portions of the overall decision

logic expression. For example, to compute the fraction of faults causing d = (a+b) c (e+f), one could first compute the fraction of faults causing (a+b) and (e+f) separately using Theorem 1, and then combine these results with the fraction of faults causing c using Theorem 2.

This method may or may not produce tight bounds. Tight bounds are not produced if any two decision logic portions contain reference to the same test outcome. For this reason, a more accurate bound is obtained when the decision logic is simplified. If the expression is simplified to standard union form (i.e., the union of intersection terms with each term being a minterm, or equivalently mutually exclusive with all other terms), then the total fraction of faults causing the expression to be true is bounded by the sum of the fraction of faults causing each term. This is summarized in the following theorem.

Theorem 5: Given a single fault class F_1 and a decision function d in the form

$$d = c_1 + c_2 + \ldots + c_N$$

$$c_i = \prod_j s_j$$

with s_j = either t_k or $\overline{t_k}$ and $c_i c_j = 0$ for all $i \neq j$ (i.e., the decision logic portions c_i are mutually exclusive). Let r_i = the fraction of F_1 faults causing $c_i = 1$. Then the fraction of F_1 faults causing d=1, FFD, is

$$FFD = \sum_i r_i$$

or when the r_i are imprecisely known but are bounded (i.e., $r_i = [L_i, H_i]$), then

$$FFD = [\sum_i L_i , \sum_i H_i]$$

COMBINING RESULTS FOR MULTIPLE FAULT CLASSES

Disjoint Fault Classes

Let the fraction of each fault class that causes the relevant diagnostic outcome be defined as f_i. Given $f_i = [L_i, H_i]$, then if all fault classes are disjoint

$$FFD = \left[\frac{\sum_{i=1}^{N} L_i \lambda_i}{\lambda_T} , \frac{\sum_{i=1}^{N} H_i \lambda_i}{\lambda_T} \right].$$

Nondisjoint Fault Classes

Theorem 6: Given two nondisjoint fault classes F_1 and F_2 with failure rates λ_1 and λ_2, and given f_1 and f_2, the fractions of faults in each class that cause the relevant diagnostic outcome, d = 1, then the total fraction of faults that causes d = 1, FFD, is given by

$$FFD = \left[\frac{N_{max}}{\lambda_1 + \lambda_2 - N_{min}} , \min \left\{ \frac{\lambda_1 f_1 + \lambda_2 f_2}{\lambda_{max}} , 1 \right\} \right]$$

where

$$N_{max} = \max \{ \lambda_1 f_1, \lambda_2 f_2 \},$$

$$N_{min} = \min \{ \lambda_1 f_1, \lambda_2 f_2 \}, \text{ and}$$

$$\lambda_{max} = \max \{ \lambda_1, \lambda_2 \}.$$

GENERAL PROCEDURES FOR FFD AND FFI CALCULATION

Definitions:

FFD:	the fraction of lifetime faults that are detected by specified means.
FFI:	the fraction of lifetime faults that are correctly isolated to RU ambiguity groups of specified size by specified means.
FFI_w:	the fraction of lifetime faults that are incorrectly isolated to RU ambiguity groups of any size by specified means.
FFI_d:	the fraction of those lifetime faults that are detected by specified means that are also correctly isolated to RU ambiguity groups of specified size by specified means.

General Procedure:

Step 1: Develop the fault accountability data f_{ij}

Step 2:

Substep 2.1: For FFD, define the decision logic (go-chain).

Substep 2.2: For FFI,

a) define the decision logic for all diagnostic outcomes with RU ambiguity of the specified size.

b) for each fault class, specify the correct diagnostic outcome.

Substep 2.3: For FFI_w,

a) define the decision logic for all diagnostic outcomes.

b) for each fault class, specify all incorrect diagnostic outcomes.

Substep 2.4: For FFI_d,

a) form the intersection of the decision logics for FFD and FFI.

b) simplify each expression where possible (for decision tree logic organization, the intersection simplifies to the isolation decision logic).

c) for each fault class specify the correct diagnostic outcome.

Step 3: For each fault class, compute bounds on the fraction of faults in that fault class that could cause the relevant diagnostic outcome.

Step 4: Combine bounds for all fault classes into the overall measure. Group fault classes into disjoint groups if possible. Use Theorem 6 for groups of nondisjoint fault classes.

Substep 4.1: If substep 2.4 was done, then to get FFI_d use Theorem 4 to derive the overall measure of fraction of faults that are both detected and correctly isolated by FFD.

Substep 4.2: If substep 2.4 was not done and FFI_d is desired, apply Theorem 7.

Theorem 7: If the fraction of all faults that are correctly isolated is r and the fraction of all faults that are detected is called s, then the fraction of detected faults that are correctly isolated, FFI_d, is given by:

$$FFI_d = [\max\{0, r+s-1\}, \min\{r, s\}] / s$$

SECTION 9

GUIDELINES FOR PREPARATION OF A DIAGNOSTIC TEST-EFFECTIVENESS ANALYSIS REPORT

This section provides a guideline for creating a Diagnostic Test-Effectiveness Analysis Report tailored to the analysis methods described in Section 8. Other methods may require other report formats.

The Government may also elect to prepare a Performance Analysis Review report (either by itself or through an independent contractor) to provide feedback to the contractor on the analysis report. The analysis report and review should be conducted prior to the demonstration (see Section 7) to allow time for modifications to the diagnostic capability to be made if necessary. In the following, the general content of the performance analysis report is outlined.

Subsection A.2 contains an example Test-Effectiveness Analysis Report for a modern military electronic unit.

OUTLINE FOR A DIAGNOSTIC TEST-EFFECTIVENESS ANALYSIS REPORT

1.0 INTRODUCTION

1.1 Purpose

State the purpose of this document (to analyze the performance of the identified diagnostic capability).

1.2 Scope

Briefly describe the diagnostic capability. Define the means of detection and isolation under consideration. State the measures that are to be estimated, if any, and their numerical goals, if available. If the diagnostic capability has different modes of operation or several distinct features, it may be more appropriate to compute measures for each feature or mode instead of a single figure of merit. Briefly discuss the fault universe under consideration in this report (e.g., are digital delay faults being considered or only stuck-at faults, are internal device faults being considered or only faults at the interfaces between devices, etc.)

1.3 Overview of Analysis

Define the performance measures to be analyzed, if any. Describe the analysis procedure used and justify its use. Describe assumptions and accuracy limitations of all quantitative methods. Describe the procedure by which the analysis was conducted (e.g., how was various data gathered, what review cycles were made, were any changes to the diagnostic design made as a result of the analysis process, etc.). State if a qualitative assessment of performance will also be made.

1.4 Applicable Documents

Refer to the design documents that contain detailed descriptions of the diagnostic capability (e.g. performance and design specifications for software, the diagnostic performance analysis report, appropriate hardware specifications, etc.).

2.0 DEFINITIONS FOR ANALYSIS

2.1 Fault-Class Decomposition

List the universe of fault-classes for which this analysis is to be done and assign failure rates. To the maximum extent possible, fault classes should either be characterized by disjoint physical circuitry or by failure modes of disjoint physical circuitry. The decomposition should be uniformly given to the next level of hardware indenture (e.g., at least specify each component as a fault class for board-level diagnostic capabilities).

If MIL-STD-217 was used to determine device or component failure rates, either refer to existing documentation or include the failure rate analysis in an appendix to this document. Engineering judgement, together with the results of any previously performed FMEA, must be relied upon to assign rates to failure modes of a device or component. State if it was assumed that all failure modes are equally likely. Include all devices, connectors, and interconnects as physical elements. Include diagnostic hardware if appropriate (the correct diagnostic outcome for diagnostic hardware faults should have been specified as part of the performance specification for the diagnostic capability). For each physical element, describe all of the failure-types to be considered and assign an identifier and a failure rate to each resulting fault class.

If the measures of interest pertain only to *critical failures*, a failure mode effects and criticality analysis (FMECA) should be made. This analysis must define the fault universe of interest for the test-effectiveness analysis as well as the fault probabilities. The fault probabilities are the probabilities that a mission-critical failure will occur over the mission time due to the fault; namely, p_i = Prob(fault i causes a mission critical effect) $\times$ $[\, 1 - e^{-(\lambda_i T_{mission})} \,]$, which replaces λ_i/λ_T in the analysis equations. The value of Prob(fault i causes a mission critical effect) may be considered a criticality value that takes on values between 0 and 1.

2.2 Detailed Description of the Diagnostic Capability

2.2.1 Overview

Provide an overview of the diagnostic capability. For sequentially operating capabilities such as initiated BIT or a TPS, this is best accomplished by describing the sequence of events that take place. For monitor-type capabilities, a top-down hierarchical description may be best. Clearly state the means of detection and isolation under consideration.

2.2.2 Definitions for Analysis

Define the basic diagnostic outcomes and assign identifiers. For all isolation outcomes, define the RU ambiguity group size. Define the individual test outcomes and assign identifiers to each. Define the logic that relates each diagnostic outcome to each test outcome (this may include demonstration that faults of the diagnostic capability hardware are not detected in some instances).

3.0 TEST APPLICABILITY ANALYSIS (OPTIONAL)

The analyses in this section assume that the diagnostic capability is correct. That is, if a particular test outcome is supposed to occur under given fault conditions, it will *always* do so. The results are therefore an upper bound to performance of the diagnostic capability.

For Fault Detection: List all of the failure modes that will result in the diagnostic capability failing to indicate that a fault is present (e.g., in TPSs, failure modes that result in all Go-Chain tests passing). Compute their failure rates and compute FFD as one minus the ratio of the sum of these failure rates to the sum of the failure rates of all failure modes of the equipment under test.

For Fault Isolation: Assume that all detected faults are correctly isolated (i.e., the diagnostic outcomes are always correct). Compute the failure rate for each fault isolation ambiguity group, λ_m. Compute the fraction of faults isolated to groups of the specified size using standard methods (e.g. MIL-STD 2165, Appendix B).

4.0 DETAILED FAULT ACCOUNTABILITY ANALYSIS

This section may be organized in one of two ways: a) by test or b) by fault. In a "by-test" organization, each test is considered in a separate paragraph as described below. In a "by-fault" organization, each paragraph corresponds to a particular fault-class as listed in Section 2 above. In this method the primary test that covers this fault mode is listed and the percentage of faults in the class under consideration that are covered by each relevant test is determined.

Organization by Test:

For each test outcome described in 2.2.2;

a) State the purpose of the test.

b) Provide a qualitative coverage assessment.

For example, list the fault classes defined in 2.1 that will cause the test outcome. Also, list any fault mechanisms (considered in 2.2.1 or not) that should, ideally, cause the test outcome (according to the purpose of the test), but in fact do not. For example, the purpose of a checkerboard memory test is to determine that data can be stored and retrieved. It will detect static data losses, but it may not pick up row/column coupling errors.

c) Quantitative Coverage Estimate

Estimate the fraction of faults in each fault class that could cause the test outcome. State how each number was arrived at (e.g., previously verified diagnostic capability, exhaustive fault simulation, sampled fault simulation, statistical fault grading technique, engineering analyses, etc.).

NOTE: 100 percent coverage of faults in a given class (e.g., in a given component) is guaranteed when a test applies an exhaustive set of inputs (e.g., all test patterns) to the fault-class and performs no output data compression. Otherwise, 100 percent fault coverage can not be guaranteed. For example, a test that exercises all functions of a device is not guaranteed to cover 100 percent of the faults even though it "covers" 100 percent of the functions. Similarly, if a test exercises X percent of the functions, then, if the faults that cause the various functions to fail are disjoint (no single fault causes multiple function failures) and there are an equal number of faults that can cause each function to fail, then the fault coverage is no greater than X percent (if there are common fault modes, then fault coverage can be greater or less than X percent). The same is true if the number of faults is different and coverage is determined by gate-count for each function.

When using fault simulations to determine the percent coverage of a test-group with respect to any fault class, the guidelines given in Debaney (1989) should be adhered to.

Summarize the quantitative coverage analysis in tabular form for easy reference in Section 5 below.

5.0 PREDICTION OF DIAGNOSTIC FIGURES OF MERIT

Outline the procedure(s) to be used and describe the organization of this section. Document both intermediate and final results. A separate subsection for each figure of merit or each separate means of detection or isolation should be provided.

NOTE: Section 8 provides two methods for accomplishing this task. In utilizing the Section 8 methods, this section should include: 1) a definition of the logical relationship between diagnostic outcomes and test outcomes, 2) the fraction of each fault class that causes each diagnostic outcome, 3) the fraction of all faults causing each diagnostic outcome, and 4) the final figure of merit (FFD or FFI).

6.0 ANALYSIS INPUTS TO MATURATION PLAN

After performing the analysis, it may be evident that improvements to the diagnostic capability can be made or that there are potential risks associated with parts of the design.

Describe any improvements to the diagnostic capability or the primary equipment that could result in significantly improved performance. Describe how performance can be monitored during developmental testing, operational testing, and initial fielding of the diagnostic capability to determine if these improvements are warranted.

Describe the potential risk areas and how performance can be monitored to determine if the risk is realized. Outline the changes that may be required if the risk is realized.

State if the Government and contractor have elected to rectify any problems that are uncovered during this analysis before a demonstration takes place. If changes to the diagnostic system are made, the above analyses may be repeated and the results summarized in an appendix.

SECTION 10

SUMMARY AND SUGGESTIONS FOR FUTURE WORK

We have presented the rationale for, the development of, and an application of a two-phase approach to the validation and demonstration of diagnostic system performance. The first phase consists of a diagnostic test-effectiveness analysis (supported by simulation and engineering judgement) that is used to predict performance, in the most precise quantitative manner possible, as a means to uncover any design deficiencies and to aid in the development of a maturation plan for the diagnostic capability, if desired. The second phase consists of a demonstration, in a factory environment, that verifies that the functional requirements, consistent with the assumptions and conclusions of the test effectiveness analysis, have been met. We highly recommend the use of an engineering prototype for this demonstration, to maximize the types of faults that can be inserted and also the randomness of these insertions. In essence, this two-phased approach employs as many fault insertion demonstrations as are practical and uses simulation and engineering judgement as backup means for providing the Government with assurances of the quality of the diagnostic capability being procured.

We feel that this effort is a reasonable first step in bridging the gap between current practices, in which all too often there is little correlation between the diagnostic figures of merit "demonstrated" and the subsequent fielded failure results, and the reliable, precise ideal, which is, for both numerical and knowledge acquisition reasons, impossible for modern electronic systems. However, there are several additional activities from which this methodology could benefit:

- Qualitative and quantitative coverage information should be developed for standard testing strategies when applied to standard circuit types (e.g., how well does a checkerboard test test a memory chip).
- More rules of thumb for estimating quantitative coverage, in the spirit of Section 8, should be developed.

- The RAC Field-Failure Return Program should be expanded, and the results should be used to standardize failure modes for both fault insertion and test analysis. Significantly more effort is needed in the area of characterizing, perhaps parametrically, major fault mode insertable/observable behavior. This RAC program represents the best means for the Government to obtain information that presently is, or would be if it existed, proprietary.

REFERENCES

Abraham, J.A. and W.K. Fuchs, "Fault and Error Models for VLSI," *Proc. IEEE*, Vol. 74, No. 5, pp. 639-654, May 1986.

Agrawal, V.D. "Sampling Techniques for Determining Fault Coverage in LSI Circuits," in *Test Generation for VLSI Chips*, Agrawal, V.D. and S.C. Seth, Eds., IEEE Computer Society Press, Washington, DC, pp. 241-247, 1988.

Agrawal, V.D. and M.R. Mercer, "Testability Measures–What Do they Tell Us?," in *Tutorial: VLSI Testing and Validation Techniques*, IEEE Computer Society Press, Washington, DC, pp. 401-406, 1985.

Agrawal, V.D. and S.C. Seth, "Chapter IV: Test Evaluation," in *Test Generation for VLSI Chips*, Agrawal, V.D, and S.C Seth, Eds., IEEE Computer Society Press, Washington, DC, pp. 159-167, 1988.

Anderson, J.M., "From a Sow's Ear – Quantitative Diagnostic Design Requirements from Anecdotal References," *Proc. AUTOTESTCON*, Philadelphia, PA, pp. 195-200, October 1989.

Anderson, W. and S.C. Binari, "Radiation Effects in GaAs Devices and ICs," *Proc. Reliability Physics Symposium*, Phoenix, AZ, pp. 316-319, 1983.

Binnendyk, F. and P. Remeis, "Fault Grading: Test Analysis By Design," *Electronics Test*, Vol. 12, No. 3, pp. 32-37, March 1989.

Binnendyk, F., "Major Advances in Fault Grading Technology," *Proc. ATE & Instrumentation Conf. East*, Boston, MA, pp. 101-108, June 1989.

Breuer, M.A. & Associates and A.J. Carlan, "State-of-the-Art Assessment of Testing and Testability of Custom LSI/VLSI Circuits, Vol. III: Fault Mode Analysis," SD-TR-83-20, Space Division, Air Force Systems Command, Los Angeles, CA, October 1983.

Brglez, F., "On Testability of Combinational Networks," in *Test Generation for VLSI Chips*, Agrawal, V.D. and S.C. Seth, Ed., IEEE Computer Society Press, Washington, DC, pp. 293-297, 1988.

Carter, J.L., V.S. Iyengar, and B.K. Rosen, "Efficient Test Coverage Determination for Delay Faults," *Proc. Int'l Test Conf.*, pp. 418-427, 1987.

Case, G.R., "Analysis of Actual Mechanisms in CMOS Logic Gates," *Proc. Design Automation Conf.*, pp. 265-270, 1976.

Chang, H.P. and J.A. Abraham, "Use of High Level Descriptions for Speedup of Fault Simulation," *Proc. Int'l Test Conf.*, pp. 278-285, 1987.

Coit, D., W. Denson, K. Key, S. Flint, and W. Turkowski, "VLSI Device Reliability Models," RADC-TR-84-182, AD-A153268, Rome Air Development Center, Griffiss AFB, NY, December 1984.

Cunningham, B.T., W.K. Fuchs, and P. Banerjee, "Fault Characterization and Delay Fault Testing of GaAS Circuits," *Proc. Int'l Test Conf.*, Washington, DC, pp. 836-842, 1987.

Debaney, W.H., "A Military Test Method for Measuring Fault Coverage," *Proc. Int'l Test Conf.*, Washington, DC, p. 951, 1989.

Denson, W.K., "IC Failure Mode Distribution Summary," Personal Communication, August 1989.

Denson, W.K. and P. Brusius, "VHSIC/VHSIC-Like Reliability Prediction Modeling," RADC-TR-89-177, AD-A214601, Rome Air Development Center, Griffiss AFB, NY, October 1989.

Devadas, S., "Delay Test Generation for Synchronous Sequential Circuits," *Proc. Int'l Test Conf.*, Washington, DC, pp. 144-152, 1989.

Franklin, M., K.K. Saluja, and K. Kinoshita, "Design of a BIST RAM with Row/Column Pattern Sensitive Fault Detection Capability," *Proc. Int'l Test Conf.*, Washington, DC, pp. 327-336, 1989.

Fujiwara, H., *Logic Testing and Design For Testability*, MIT Press, Cambridge, MA, 1985.

Galiay, J., Y. Crouzet and M. Vergniault, "Physical Versus Logical Fault Models in MOS LSI Circuits: Impact on Their Testability," *IEEE Trans. on Computers*, Vol. C-29, No. 6, pp. 527-531, June 1980.

Ghate, P.B., "Electromigration-induced Failures in VLSI Interconnects," *Proc. Reliability Physics Symposium*, San Diego, CA, pp. 292-299, 1982.

Green, T.J., "Getting the Facts from the Field...Real World Failure Data Collection and Analysis," *Government Microcircuit Applications Conf. (GOMAC) Digest of Papers*, Orlando, FL, pp. 105-109, October 1987.

Green, T.J., "A Review of Microcircuit and Hybrid Field Failures from Air Force Avionic Equipment," *Proc. National Aerospace and Electronics Conf.*, Dayton, OH, pp. 1544-1548, 1988.

Green, T.J. and W.K. Denson, "A Review of EOS/ESD Field Failures in Military Equipment," *Proc. EOS/ESD Symposium*, 1988.

Gruska, G.F. and M.S. Heaphy, "A Structured Approach to Meeting the DoD Quality Policy," *Proc. National Aerospace and Electronics Conf.*, Dayton, OH, pp. 1665-1673, May 1989.

Ho, P.S., "Basic Problems of Electromigration in VLSI Applications," *Proc. Reliability Physics Symposium*, San Diego, CA, pp. 288-291, 1982.

Hoover, S.V. and R.F. Perry, *Simulation–A Problem-Solving Approach*, Addison Wesley, Reading, MA, p. 198, 1989

Huisman, L., "The Reliability of Approximate Testability Measures," in *IEEE Design & Test of Computers*, Vol. 5, No. 6, pp. 57-67, December 1988.

Hummel, R. A., "Automated Fault ı. ːtion for Digital Systems," *Proc. Reliability and Maintainability Symposium*, ⌣os Angeles, CA, pp. 112-117, 1988.

Jain, S. and V.D. Agrawal, "Satistical Fault Analysis," in *Test Generation for VLSI Chips*, Agrawal, V.D. and S.C. Seth, Eds., IEEE Computer Society Press, Washington, DC, pp. 234-240, 1988.

Lochner, R., "A Sequential Process for Total Quality Management," *Proc. National Aerospace and Electronics Conf.*, Dayton, OH, pp. 1641-1644, May 1989.

Malaiya, Y. and S.Y.H. Su, "Testability of VLSI Leakage Faults in CMOS," RADC-TR-83-202, AD-A138978, Rome Air Development Center, Griffiss AFB, NY, September 1983.

Maly, W., "Realistic Fault Modeling for VLSI Testing," *Proc. 24th ACM/IEEE Design Automation Conf.*, Miami Beach, FL, pp. 173-180, 1987.

Mangir, T.E., "Sources of Failures and Yield Improvement for VLSI and Restructurable Interconnects for RVLSI and WSI: Part I–Sources of Failures and Yield Improvement for VLSI," *Proc. IEEE*, Vol. 72, No. 6, pp. 690-708, June 1984.

Meth, M.A. and T.A. Musson, "Finding Faults—A Dilemma for Statisticians," *Proc. Reliability and Maintainability Symposium*, Orlando, FL, pp. 351-355, January 1983.

Micro Control Company, "Standard Patterns for Testing Memories," *Electronics Test*, Application Note, Vol. 4, No. 4, pp. 22-26, April 1981.

MIL-HDBK-217E, U.S. Department of Defense, "Reliability Prediction of Electronic Equipment," 27 October 1986.

MIL-I-38535, U.S. Department of Defense, "General Specification for Integrated Circuits (Microcircuits) Manufacturing," December 1989.

MIL-STD-470A, U.S. Department of Defense, "Maintainability Program for Systems and Equipment," March 1966.

MIL-STD-471A, U.S. Department of Defense, "Maintainability Verification/Demonstration/ Evaluation," March 1973.

MIL-STD-883, U.S. Department of Defense, "Method 5012 Fault Coverage Measurement for Digital Microcircuits," 1989.

MIL-STD-2077 (AS), U.S. Department of Defense, "General Requirements for Test Program Sets," March 1978.

MIL-STD-2165, U.S. Department of Defense, "Testability Program for Electronic Systems and Equipment," January 1985.

Nagel, L.W., "SPICE 2: A Computer Program to Simulate Semiconductor Circuits," ERL-M520, Department of EE and CS, University of California, Berkeley, CA, August 1981.

Papadimitriou, C.H. and K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw Hill, New York, 1965.

Rossi, M., "Microcircuit Device Reliability–Field Experience Database," FMDR-21A, Reliability Analysis Center, PO Box 4700, Rome, NY, June 1986.

Seth, S.C., L. Pan, and V.D. Agrawal, "PREDICT – Probabilistic Estimation of Digital Circuit Testability," in *Test Generation for VLSI Chips*, Agrawal, V.D. and S.C. Seth, Eds., IEEE Computer Society Press, Washington, DC, pp. 298-303, 1988.

Sie, C.H., R.A. Youngblood, J.H. Liao, and A. Turk, "Soft Failure Modes in MOS RAMs," *Proc. Reliability Physics Symposium*, Las Vegas, NV, pp. 27-32, 1977.

Siewiorek, D. and R. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, MA, 1982.

Simpson, W. R., J.H. Bailey, K.B. Barto, and E. Esker, "Prediction and Analysis of Testability Attributes: Organizational-Level Testability Prediction," RADC TR-85-268, Rome Air Development Center, Griffiss AFB, NY, February 1986.

Stockman, S. and D. Rash, "Microcircuit Device Reliability Trend Analysis," RAC Doc. MDR-21, Reliability Analysis Center, PO Box 4700, Rome, NY, July 1985.

Thomas, J.J., "Automated Diagnostic Test Program for Digital Networks," *Computer Design*, pp. 63-77, August 1971.

Timoc, C., M. Buehler, T. Griswold, C. Pina, F. Scott, and L. Hess, "Logical Models of Physical Failures," *Proc. Int'l Test Conf.*, Philadelphia, PA, pp. 546-553, November 1983.

U.S. Air Force, "MATE TPS Verification Guidelines," USAF Doc. 2806645 Rev. B, 1 April 1985.

U.S. Air Force, "RADC Reliability Engineers Toolkit," Systems Reliability and Engineering Division, Rome Air Development Center, Griffiss AFB, NY, July 1988.

U.S. Army Program Manager for TPS, "TPS Procedures Manual–Rev. 1," AMCPM-TMDE-T, Fort Monmouth, NJ, 1987.

U.S. Department of Defense, "Total Quality Management Master Plan," August 1988.

Wadsack, R., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits," *Bell System Technical Journal*, Vol. 57, No. 5, pp. 1449-1473, May-June 1978.

Weiss, J.L., M. Doyle-Buckley, and J.C. Deckert, "A Prototype Test-Engineering Workstation for Analog Electronics," *Proc. ATE & Instrumentation Conf. East*, Boston, MA, pp. 539-550, June 1989.

Williams, T.W., W. Daehn, M. Gruetzner, and C.W. Starke, "Aliasing Errors in Signature Analysis Registers," *Proc. Int'l Test Conf.*, Washington, DC, pp. 282-288, 1986.

Wong, K.L., J.M. Kallis, and A.H. Burkhard, "Culprits Causing Avionic Equipment Failures," *Proc. Reliability and Maintainability Symposium*, Philadelphia, PA, pp. 416-421, 1987.

Zins, E. and G. Smith ,"R&M Attributes of VHSIC/VLSI Technology," *Proc. Reliability and Maintainability Symposium*, Philadelphia, PA, pp. 403-406, 1987.

APPENDIX

APPLICATION OF THE METHODOLOGY TO A MODERN MILITARY ELECTRONIC UNIT

A.1 INTRODUCTION

In this appendix we provide an illustration of the application of the methodology developed in the body of this report to the Prognostic Diagnostic Interface Unit (PDIU), a modern Army electronic unit developed by GE-ASD to perform diagnostics and limited prognostics for the M109E5 Improved Howitzer. Specifically, we use the methodology to validate and demonstrate the PDIU's self-test capability. Subsection A.2 provides a self-contained Test-Effectiveness Analysis Report for the PDIU self-test feature, which serves as an example of an application of the guidelines given in Section 9. Subsection A.3 contains a self-contained Demonstration Plan for the PDIU self-test feature, an example of an application of the guidelines given in Section 7. This plan incorporates the physical and software insertion techniques of Section 6. Finally, subsection A.4 provides a summary and conclusions from the demonstration effort.

A.2 PDIU TEST EFFECTIVENESS ANALYSIS REPORT

This subsection consists of a *self-contained* Test-Effectiveness Analysis Report for the PDIU's self-test feature. All section, subsection, figure, and table numbers are internally consistent, but they are not consistent with the other portions of this report.

The primary purpose of this subsection is to *illustrate* the application of the techniques developed in this report, organized according to the guidelines presented in Section 9, to a modern military electronic unit. Because of the limited scope of this effort, only the probabilistic analysis methodology of subsection 8.3 is employed here. In an actual application, use of the more tedious–but also more accurate–set-theoretic analysis methodology of subsection 8.4 would probably be preferable.

TEST-EFFECTIVENESS ANALYSIS REPORT FOR THE PROGNOSTIC DIAGNOSTIC INTERFACE UNIT

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 PURPOSE

This report presents a detailed diagnostic test-effectiveness analysis of the self-test feature of the Prognostic Diagnostic Interface Unit (PDIU). The objective of this report is to provide a prediction of the fraction of field-faults detected (FFD) by the PDIU self-test capability and to document the PDIU faults that are, and are not, detected by PDIU self-test.

1.2 SCOPE

The primary mission of the PDIU is to perform diagnostics and limited prognostics on the Army M109E5 Improved Howitzer. The PDIU performs three major functions: 1) the Operational Mode function, in which the PDIU monitors major howitzer subsystems, 2) the Maintenance Mode function, which supports Howitzer fault detection and isolation activities at the unit level, and 3) PDIU self-test. The PDIU self-test feature performs fault detection on the PDIU itself. This report is an analysis of the PDIU self-test fault detection capabilities (no fault-isolation capability is provided in the PDIU self-test and is therefore not considered here). All software, including self-test software is stored in non-volatile EEPROM memory within the PDIU.

The PDIU self-test feature consists of three distinct features: 1) Power-Up Confidence Test (PUCT), 2) On-Demand/On-Event (OD) tests, and 3) Continuous Monitoring (CM) mode. On PDIU power-up, a BIT Lamp on the PDIU casing is turned on and the PUCT routines are run. This lamp remains on during PUCT operation. If no fault is found by PUCT, then the BIT lamp is turned off. If the lamp remains on, a fault is indicated. Additionally, faults are indicated by transmission of a fault code message over the RS-422 bus to a display and control unit (DCU), which causes a message to be displayed to the operator on a CRT. On-Demand tests occur when they are requested by the operator through the DCU when the PDIU is in maintenance mode. Similarly, On-Event tests are automatically run during specific events that are initiated by the oper-

ator from the DCU. The continuous monitoring feature operates during execution of application or system software by the PDIU processor. CM mode consists of a watchdog timer I/O board reference voltage test, and processor board supply voltage test. The watchdog timer must be reset by any application code being run by the PDIU processor. If the reset fails to occur, then the BIT lamp is again turned on, PUCT is run, and the BIT lamp set according to PUCT results.

For this report, the means of detection are defined by operator recognition of a fault through either the BIT lamp, the display of an error message on the DCU, or both. Three separate fault detection mechanisms will be examined: 1) detection of faults by PUCT; 2) detection by sequential execution of all OD tests; and 3) detection during CM mode .

The fault universe considered for this report will be limited. The entire PDIU is composed of six major boards and interconnections between boards (see Fig. 1). These major boards consist of a processor board, two memory boards, an I/O board, a signal protection/MUX board, and a power supply board. To limit the scope of this report, the fault universe will consist only of faults in the non-external-interface portions of the PDIU processor, memory, and I/O boards (all components except the 1553 chip-set and the RS-422 transceiver components).

1.3 ANALYSIS OVERVIEW

The analysis of the PDIU self-test capability presented in this report consists of a qualitative fault-coverage analysis and a quantitative prediction of fraction of faults detected (FFD) by PDIU self-test during field operations.

Qualitative coverage is presented by reference to the PDIU Failure Mode Effects and Criticality Analysis (FMECA) in which all major fault modes of the PDIU are derived [1] (references in this report refer to those listed in subsection 1.4). For each fault mode, the test steps within the PDIU self-test capability that may detect that fault mode are listed. If the fault mode has the possibility of being undetected, an explanation of how this situation might occur is given.

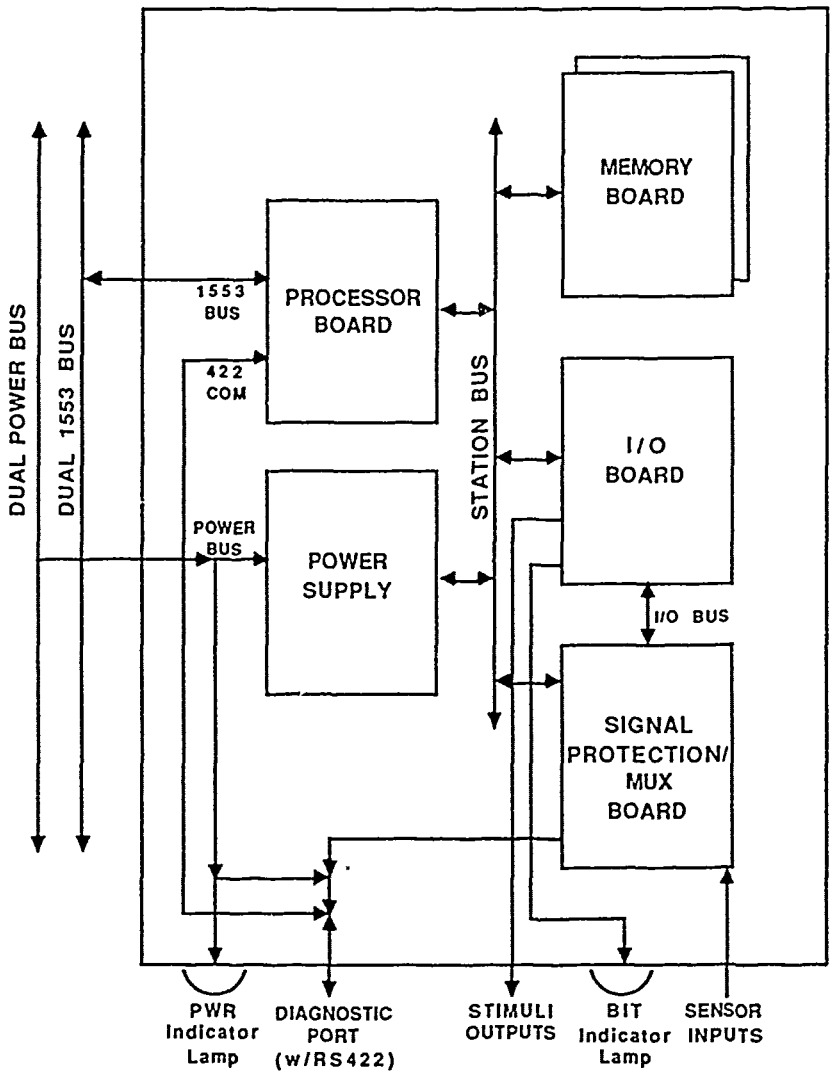


Figure 1. PDIU Configuration

Quantitative prediction of FFD is accomplished by the probabilistic fault-accountability approach discussed in [2]. For each fault class discussed in the qualitative coverage analysis, a percentage figure is given for each test step that is capable of detecting that fault class. The percentage figure represents the fraction of physical faults encompassed by the fault class that are actually detected by the specified test-step. These figures are determined by engineering analysis alone. No fault simulation has been used to derive these numbers. For each fault class, the fraction of physical faults encompassed by that fault mode that will cause the primary diagnostic outcome to occur is computed. It is then assumed (realistically) that the fault classes are disjoint (there is no common physical fault that can cause failures in more than one fault class) and FFD is

computed. If f_i is the fraction of fault class i faults that cause the primary diagnostic outcome (determined by the above accountability analysis), then FFD is computed by

$$FFD = \frac{\sum_i \lambda_i f_i}{\sum_i \lambda_i}$$

where λ_i is the failure rate for the fault class i.

This analysis was performed after the PDIU was fully designed, tested, and accepted. No changes to the PDIU hardware or its self-test software were made as a result of this analysis.

1.4 APPLICABLE DOCUMENTS

1. *Failure Mode Effects and Criticality Analysis for the PDIU*, GE Document, CDRL A119, Prime Contract DAAA21-86-C-0023, February 1987.

2. "Analysis Methods," Section 8 in *Analysis and Demonstration of Diagnostic Performance in Modern Electronic Systems*, Final Report on Contract F30602-88-C-0068, TR-494, ALPHATECH, Inc., March 1991.

3. *Testability Analysis for the CID for the M1A2*, GE Document, SDRL 023B, Letter Subcontract WPG000313, Prime Contract DAAE07-89-C-R045, 18 July 1990.

2. DEFINITIONS

2.1 FAULT CLASSES

This subsection defines the hardware fault classes of the PDIU self-test feature considered in this report. Table 1 indicates the fault classes for the PDIU, exclusive of the power supply board, which was excluded from PDIU self-test requirements due to other preferable means of coverage. Additional fault classes that were intentionally excluded from PDIU self-test coverage for the same reason are shown with strikethroughs in Table 1. Only the remaining fault classes will be included in this study. Table 1 also indicates the failure rate for each fault class in failures per million hours (FPMH).

The failure rates for the fault classes in Table 1 were determined in the following manner. (This approach was taken in order to avoid performing another FMEA for the PDIU for this effort, since the FMECA in [1] did not develop failure rates for the individual fault classes on each board. Also, while [1] did enumerate high-level component failure modes, no failure rates were developed for those modes.) First, the failure rates for each board had been determined in compliance with the MIL-HDBK-217D parts-count method [1]. Next, we utilized the FMEA [3] performed for another GE-ASD product, the Commander's Integrated Display (CID), which has functionality, components, topology, and fault classes similar to the PDIU. We used the fraction of the failure rate for each fault class of the CID relative to its total board failure rate to calculate the corresponding PDIU fault class failure rate from its board failure rate. These resulting fault class failure rates were then adjusted, where necessary, for the presence of multiple devices.

2.2 DESCRIPTION OF THE PDIU SELF-TEST FEATURE

Overview

The PDIU is designed with several levels of self-diagnostic capabilities. These include the Power-Up Confidence Test (PUCT), On-Demand/On-Event (OD) Tests, and a Continuous

TABLE 1. PDIU FAULT CLASSES AND FAILURE RATES

Fault Class	Component/ Component Group	Failure Rate (FPMH)
Processor Board		
210	Microprocessor and Support Electronics	10.5
220	Local Bus (Address Latch and Control Signal Buffer)	1.5
230	Station Bus (Address Latch, Data and Control Transceiver)	4.8
240	Buffer/Transceiver Control Circuitry (PAL & Glue Logic)	2.0
~~250~~	~~Test Buffer~~	~~not included~~
260	SRAM (32K × 16)	12.8
270	1553 SRAM (2K × 16)	0.8
280	PROM	6.4
~~290~~	~~RS-422 Interface~~	~~not included~~
2A0	1553 Remote Terminal Interface	15.1
~~2B0~~	~~1553 Transformers~~	~~not included~~
Memory Board (Each, Two Total)		
310	EEPROM Memory Devices (24–8K × 8)	48.0
320	Decode and Control	20.0
330	Station Bus and DTACK Generator	1.2
~~340~~	~~Testability Transceiver~~	~~not included~~
I/O Board		
410	Programmable Counter/Timer	2.0
420	Decode and Control Interface	1.5
430	Station Bus and DTACK Generator	1.2
440	BIT Lamp Driver	0.2
450	Vcc Monitor	2.0
460	A/D Converters	2.5
~~470~~	~~Filters~~	~~not included~~
~~480~~	~~Multiplexers~~	~~not included~~
Signal Protection (SP)/Mux Board		
~~510~~	~~Input Conditioning~~	~~not included~~
~~520~~	~~Multiplexers~~	~~not included~~
~~530~~	~~Control and Interface with I/O Board~~	~~not included~~

Monitoring (CM) mode. PUCT occurs automatically when the PDIU is initially powered up and after a system reset. In addition, PUCT can be initiated by the operator through the DCU when in maintenance mode, and PUCT is run whenever certain events occur during the operation of the PDIU that cause a system reset (e.g., watchdog time-out, error in EEPROM checksum). During the operation of the PDIU, the CM watchdog timer test and voltage checks are continuously executed to monitor system operation fidelity. If a failure prevented the PDIU's system software from resetting the timer in the normal periodic manner, the watchdog timer resets the PDIU and initiates PUCT. Following such a reset, the PDIU reports the watchdog time-out error to the DCU

and uses PUCT to verify system failures. If at any time a test fails, a software hook is set. The software hook is coded such that each bit in the software hook corresponds to a different test. In this way, it is easy to recover the status of all of the self-test procedures. PDIU self-test failure detections are brought to the operator's attention by turning on the PDIU BIT failure indicator lamp (see Fig. 1) and sending an error or warning message to the DCU (if the failure does not prevent fault reporting).

Power-Up Confidence Test (PUCT)

The PDIU contains procedures that test its processor, memory, and hardware timers. Since these tests will interfere with the operation of the system, they must be run before the Ada Run-Time Environment (RTE) is initialized. Consequently, all tests are written in assembly language and run immediately after a power-up reset. These tests store their results in designated memory locations that can be accessed during system operation. After execution of the PUCT, control passes to the initialization code for the Ada RTE.

If PUCT did not detect any failures, the PDIU BIT failure indicator lamp will be turned off. Otherwise, the lamp will be left on, and the PDIU will send an error message to the DCU (provided that the failure does not interfere with the mechanism for doing so). PUCT normally takes approximately five seconds to complete.

The PUCT consists of three major test groups: 1) Processor Test, 2) RAM Test, and 3) Timer Test. These tests are now described in more detail.

PROCESSOR TEST

The processor test portion of PUCT tests the 80186 processor's registers, addressing modes, and instruction set. If any of these tests fail, the PUCT software jumps to an ERROR routine, and the processor has failed the processor check. The STATUS CODE, stored in 80186 register AX, has bit one, the processor bit, set to 1. This signifies that the processor check failed. Control is then passed to RAM test.

Instruction Set Test

All but six 80186 instructions are tested (92 out of 98 instructions). The instructions not included are: LEA, ESC, LOCK, HLT, WAIT, and BOUND. The instructions are tested by comparing the execution of the particular instruction with the expected result. An error is found if the instruction does not execute properly or yield the expected result. The flag and jump instructions are tested first to insure the validity of their use throughout the remainder of the processor test.

Register Tests

The general purpose registers and segment registers are tested by writing a value to each register and reading back the value to determine if the registers can be written to and read from correctly.

The eight 16-bit general purpose registers are written with alternating 1s and 0s (AAAAH and 5555H) and read to check for errors. The test is a wraparound test, in which one register is written with data and the data is then passed to all registers sequentially.

The segment registers are tested similarly to the general purpose registers, verifying that each can be written to and read correctly. This test requires the use of the CS and IP registers as the program counter and assumes that they are fault-free. If either of these registers is faulty, this test will yield unpredictable results. Therefore, if the test is executing correctly, it is implied that the CP and IP registers are functioning properly, and they do not have to be explicitly tested. If any error is found, the test fails, and a software bit is set.

Address Test

All eight addressing modes of the 80186 (register operand, immediate operand, direct, register indirect, based, indexed, based-indexed, and based-indexed with displacement mode) are tested. Immediate and register addressing modes are used to set up registers AX, BX, CX so that the remaining modes can be tested. The remaining modes are tested by writing a known value to

whichever location is specified and comparing the value read with the expected value. The test fails if there is a difference between the expected value and the actual value.

RAM TEST

The PUCT tests every RAM location on the processor board (all 64K of RAM). This includes the SRAM dedicated to the microprocessor and the SRAM shared with the 1553B remote terminal interface (RTI). The RAM test consists of first writing alternating 1s and 0s (5555H) to each RAM location, and then reading back this data to verify proper functionality. If correct data are found, then AAAAH is written to each location, to complete the test for stuck-at memory locations. Once again, the RAM is read and if all the data are correct, then this portion of the PUCT has passed. If at any time during the RAM test incorrect data is encountered, a software hook bit is set indicating a failure in the RAM test, and the PUCT moves on to the next test.

When the test is successfully completed, the status code is copied from register AX into RAM location 8000H. If RAM test finds a faulty location, location 7FFFH contains the first faulty location encountered. If a faulty 1553 RAM location is found, the 1553 RAM bit is set and the faulty location is stored in 7FFEH.

PROCESSOR TIMER TESTS

The Intel 80186 microprocessor uses three internal timers, whose count rate is derived from the internal CPU clock rate. This test checks the operation of the 80186's timers. This test tests two conditions of these timers: the timers' counting ability and the timers' overflow handling. Both of these conditions are tested in the same manner. A counting loop of known run time is executed. When the loop is finished, the counter register value is compared with the expected value of the loop. For the first condition, if the counter register value is within 10 percent of their expected value, then this portion of the test has passed. For the second condition, the MAX Count value is set such that executing the counting loop will cause the counter register to overflow. Upon overflow, the timer should reset to zero. The high and low limits for this test are both 0, the overflow value. The third timer is only tested for its counting ability because it is driven by the

second timer, which has already been tested. If at any time a failure is encountered, a software hook bit is set corresponding to the processor timing test, and the PUCT moves on to the next test.

I/O BOARD TIMER TESTS

The I/O board has three wrap-around timers. The timer is set so that it will wrap around. The test program loop is then run, and the value of the timer is checked at the end of the loop. The test has passed if the value stored in the timer is within 10 percent of the expected value. Two of the three timers are tested in this manner. The third timer is the watchdog timer. This is tested by setting the timer value high enough such that the timer will not reach zero (i.e., overflow) by the time it is checked following a software timing loop.

On-Demand/On-Event Self-Tests

On-Demand tests are executed when the operator request a specific test via the DCU. On-Demand tests include execution of PUCT and tests on EEPROM, analog input paths, and voltage sources to ensure functionality of all hardware components. When executed, each of these test subprograms returns a parameter indicating that the test passed or a code identifying which portion of the test failed. In addition, this unit contains a module that allows resetting of the watchdog timer before it times out. On-Event tests occur after certain normal PDIU events such as the making an analog measurement and writing to/from the EEPROM data space. Details of the On-Demand/On-Event (OD) tests are given below.

PROCESSOR BOARD PROM TEST

The 16K bytes of PROM on the processor board are tested by performing a checksum. If the value of the output of the test does not agree with the stored value, then an error is detected and logged in the software hook.

1553B SELF-TEST

The 1553B self-test is initiated by the 1553 bus controller at the DCU which issues a self-test command. The 1553B RTI decodes the message and begins the self-test after the host

computer detects this data and resets the RTI. At the end of the 1553B self-test, the the following commands have been tested: initiate self-test, transmit, receive, synchronize, reset remote terminal and transmit status word. These tests are performed on both the A and B channels to thoroughly test the RTI. The results of the self-test are stored in memory location 413H and are read by the DCU when the bus controller issues a transmit BIT word code.

MEMORY BOARD EEPROM TEST

For On-Demand testing, EEPROM on the memory board is tested by performing a checksum calculation verification. In this test the current memory contents are used to calculate the checksum and the result is compared to the value that already stored in memory. Since EEPROM is updated during the normal operation of the PDIU, software has been provided to modify and update the value of the checksum. This will prevent false error messages from occurring. If the output value of the test is different from the stored checksum value, an error is detected and the appropriate software hook bit is set.

I/O BOARD ANALOG DATA ACQUISITION TEST

This test is designed to determine the integrity of the hardware in the analog data acquisition path on the I/O Board. It includes tests for: faulty A/D reading, any error bringing a constant reference voltage through the analog path, and A/D output not varying linearly with the input. This last test is accomplished by varying the input voltage, the programmable filter value, and the programmable gain amplifier.

Continuous Monitoring

WATCHDOG TIMER

Approximately every 30 seconds the system software will reset the watchdog timer to a value of 66 seconds. If the watchdog timer is allowed to time-out (i.e., system software failed to reset it in the allotted time) the PDIU status lamp will be shut off, and the PDIU will be reset. The

ensuing PDIU power up will find (by reading an EEPROM location) that the previous shutdown was abnormal and send a message to the DCU indicating this fact.

REFERENCE VOLTAGE TEST

Additionally a form of continuous background test exists for the I/O board. Immediately prior to every measurement taken on the I/O board during normal operations, a reference voltage is switched in and measured to verify the soundness of the I/O board measurement path.

VCC MONITOR

A drop in the +5 volt supply (Vcc) below a preset threshold of approximately 4.65 volts is detectable by the Vcc Monitor circuitry on the I/O board. Vcc is sensed on the Processor Board by a pair of sense lines, and input to the Vcc Monitor on the I/O Board. The Vcc Monitor circuitry issues a reset signal to the processor if Vcc falls below the threshold, below which timing parameters for the processor are not guaranteed. A reset signal is issued to ensure that the processor does not write invalid data into EEPROM memory when Vcc is below the threshold.

2.3 HARDWARE CONFIGURATION OF THE PDIU

PDIU Processor Board

The core of the processor board (see Fig. 2) is the 16-bit Intel 80186 microprocessor (6 MHz military version). The microprocessor communicates with the local board functions via a (proprietary) local bus. The local bus consists of the address, data, and control lines that run from the 80186 microprocessor to other circuitry on the processor board. The processor board provides a station bus interface for communicating with the other PDIU boards. The station bus consists of the address, data and control lines which run between the processor board, I/O board and memory board.

For external communications the processor board provides RS-422 circuitry and MIL-STD 1553B RTI hardware. The RS-422 circuitry consists of a transceiver and a USART (Universal Synchronous Asynchronous Receiver Transmitter) and is intended to interface with a portable

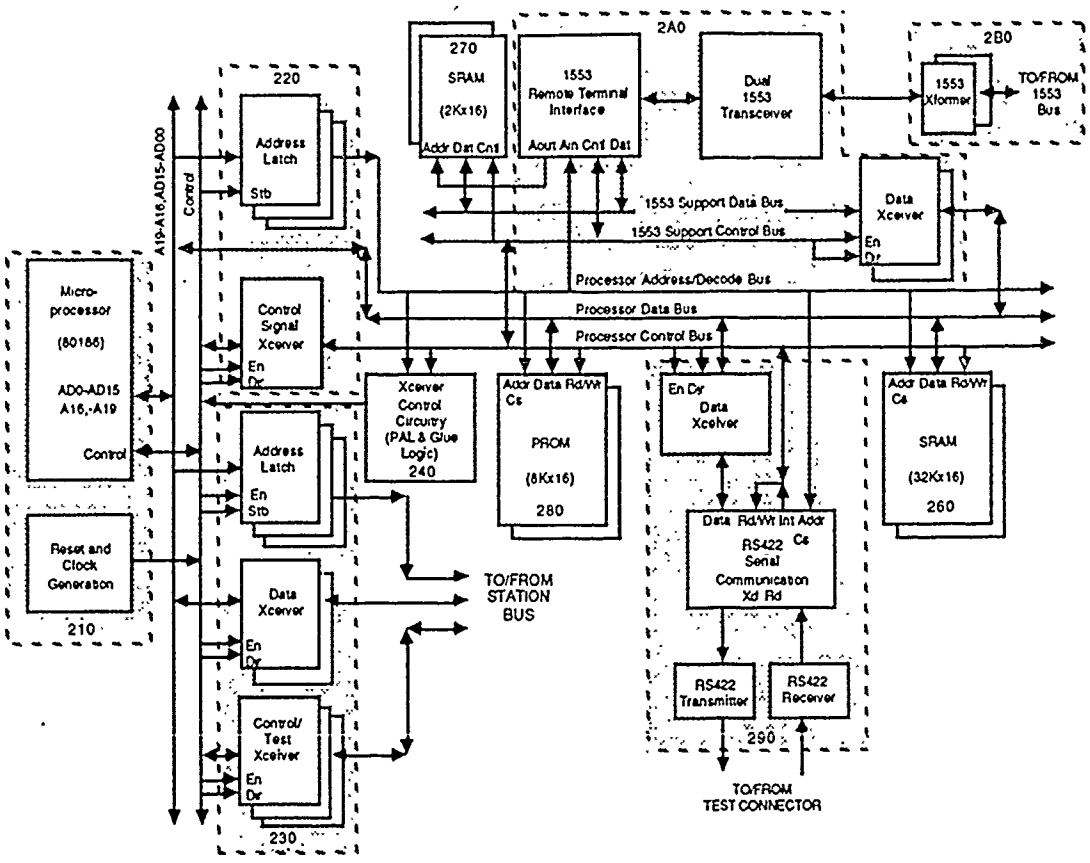


Figure 2. PDIU Processor Board Configuration

maintenance aid or peripheral equipment. The 1553B interface is used to communicate with the DCU and the rest of the howitzer.

On-board memory consists of 16K bytes of EEPROM, 64K bytes of dedicated SRAM and 4K bytes of SRAM shared with the 1553B RTI. The EEPROM is used for system initialization, while the dedicated SRAM is used for locating interrupt vectors and storing elements of the Ada Language System .

PDIU Memory Boards

The PDIU memory boards serve as extensions of the processor board memory. Each board (see Fig. 3) consists of 24 8K × 8 EEPROMs for a total of 192K bytes of EEPROM used for the storage of system software and various constants. Programmable array logic (PAL) is used to decode the address of the EEPROM. Data on each memory board is accessed by the processor board over the PDIU station bus.

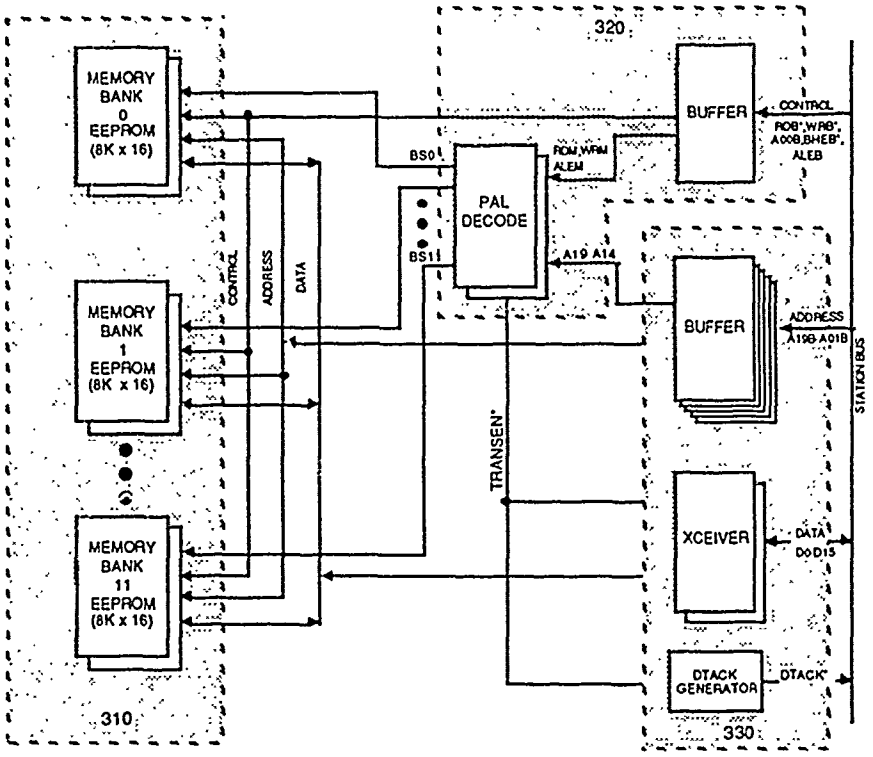


Figure 3. PDIU Memory Board Configuration

PDIU I/O Board

The PDIU I/O board (see Fig. 4) serves as a link between outside signals and the PDIU processor board. The I/O board receives conditional analog and digital signals from the SP/MUX (Signal Protection/Multiplexer) board and makes them available to the microprocessor. The entering signal proceeds through the final multiplexers into a differential amplifier. The output of the amplifier passes through a programmable filter, into a programmable gain amplifier and finally through a 12-bit A/D converter. The output of the A/D converter is available to the station bus and can be read by the processor board. The I/O board also drives the BIT failure indicator lamp.

PDIU SP/MUX Board

The PDIU SP/MUX board protects the PDIU electronics from damage due to faults and high voltage transients on the incoming signal lines. Overvoltage protection, short-circuit protection, and protection against inadvertent grounding is supplied. Attenuation networks are used where necessary to reduce the signal voltages to a safe level for the I/O board to read. The SP/MUX board also provides the first and second level of signal input multiplexers. The

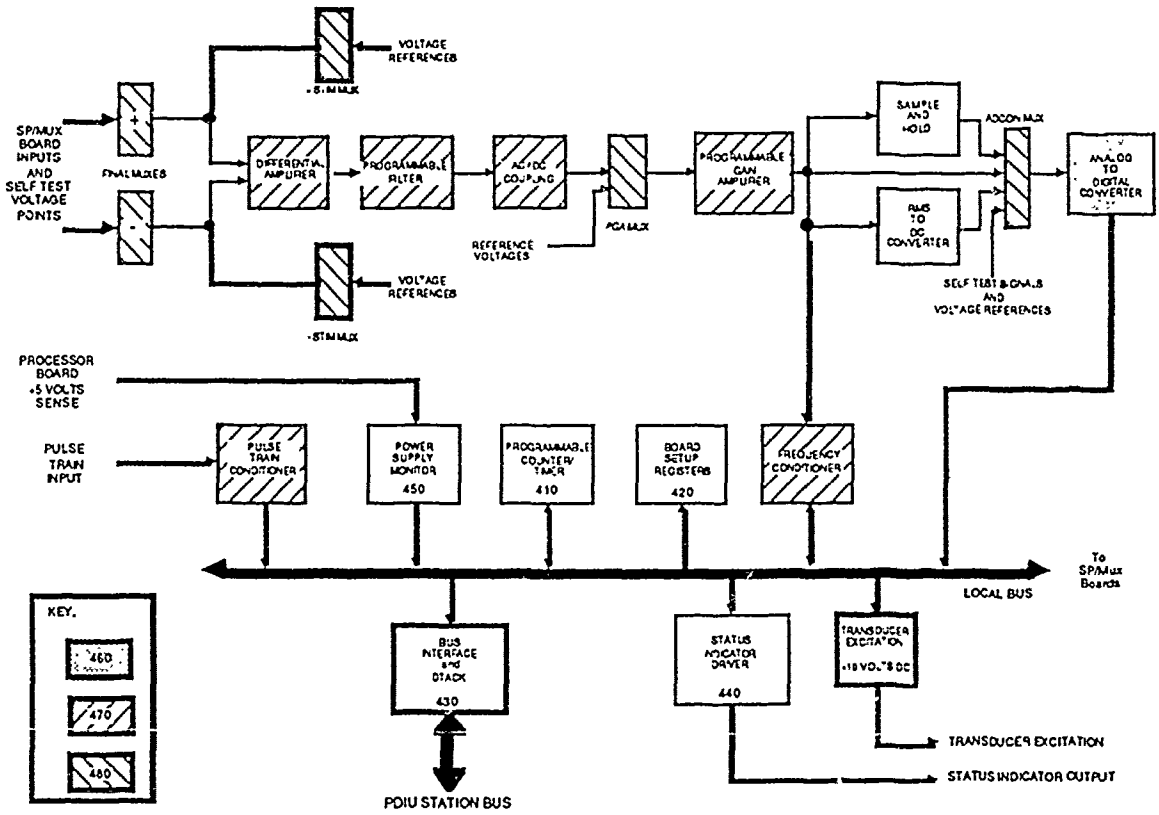


Figure 4. PDIU I/O Board Configuration

multiplexers are addressed from the I/O data bus. The output of the SP/MUX board is a differential signal that goes to the final multiplexers located on the I/O board.

PDIU Motherboard

The PDIU motherboard supplies the interconnections required for communication between hardware on different boards. The motherboard includes the station bus and I/O data bus between the I/O and SP/MUX board. The power lines included in the station bus consist of +5V, +15V, -15V, analog ground, and digital ground. It supplies power to all PDIU boards. The station bus also provides the data, address, and control lines of the 80186 to the I/O board and SP/MUX board. The station bus is used to set up the multiplexers on the SP/MUX board and return input signals.

PDIU DC-DC Converter

The PDIU DC-DC converter receives conditioned 28V DC input power from the howitzer. All electrical connections to the DC-DC converter are made via plug-in connectors. From this input voltage the converter supplies three regulated outputs: +5V, +15V, and -15V. The +5V output has its own return, while the +/- 15V outputs share a common return line. The converter has a maximum power output of 40 watts.

3. TEST APPLICABILITY ANALYSIS

This optional analysis was not performed for the PDIU.

4. FAULT ACCOUNTABILITY ANALYSIS

For this study, we assume that a fault is detected if either the BIT Lamp remains on for more than about one minute or any error message is displayed on the DCU (or both). Since there are several entry points to the PDIU self-test software, we will provide separate fault coverage predictions for each entry point: 1) PUCT ; 2) serial execution of all OD tests; and 3) CM mode.

Table 2 indicates the diagnostic tests comprising the PDIU self-test feature and their test designators. Note that for each of these self-test modes, detection for the mode (PUCT, OD, and CM) occurs if any one of its subordinate tests detects a fault, and self-test detection occurs if any mode detects the fault (i.e., the detection logic is *union* for all modes and self-test itself).

TABLE 2. PDIU TESTS

TEST NAME	TEST DESIGNATOR
PUCT	P
Processor Test	P.1
Instruction Set Test	P.1.1
Register Test	P.1.2
Address Test	P.1.3
RAM Test	P.2
Processor RAM	P.2.1
Shared RAM (1553)	P.2.2
Timer Tests	P.3
Processor Internal Timers	P.3.1
I/O Board Timers	P.3.2
On-Demand Tests	OD
Processor PROM Test	OD.1
1553B Self-Test	OD.2
Memory Board EEPROM Test	OD.3
I/O Board Data Acquisition Test	OD.4
Continuous Monitoring	CM
Watchdog Timer	CM.1
I/O Board Reference Voltage Test	CM.2
Processor Vcc Monitor	CM.3

Table 3 contains the fault accountability matrix that provides the coverage information linking the PDIU diagnostic tests and the PDIU fault classes. The coverage figures for the

TABLE 3. PDIU FAULT ACCOUNTABILITY MATRIX

FAULT CLASS	FAILURE RATE (FPMH)	FRACTION OF FAULT CLASS FAILURES COVERED													TOTAL SELF-TEST COVERAGE
		PUCT							ON-DEMAND TESTS				CONTINUOUS MONITOR			
		PROCESSOR TEST			RAM TEST		TIMER TEST									
		P.1.1	P.1.2	P.1.3	P.2.1	P.2.2	P.3.1	P3.2	OD.1	OD.2	OD.3	OD.4	CM.1	CM.2	CM.3	
210	10.5	0.30	0.20	0.20	0.10	0.10	0.30	0.20	0.20	-	0.20	-	0.40	-	0.10	
		PUCT = 0.70 (0.78)							OD = 0.36				CM = 0.46			0.90 (0.92)
220	1.5	-	-	0.50	0.50	0.50	-	-	0.50	0.50	-	-	0.20	-	-	
		PUCT = 0.50 (0.88)							OD = 0.50 (0.75)				CM= 0.20			0.70 (0.98)
230	4.8	-	-	0.50	-	-	-	0.20	-	-	0.90	-	-	-	-	
		PUCT = 0.60							OD = 0.90				CM = 0.00			0.90 (0.96)
240	2.0	0.10	-	-	0.10	0.10	-	-	0.30	0.30	-	-	0.40	-	-	
		PUCT = 0.20 (0.27)							OD = 0.40 (0.51)				CM = 0.40			0.50 (0.79)
260	12.8	-	-	0.40	0.90	-	-	-	-	-	-	-	-	-	-	
		PUCT = 0.90 (0.94)							OD = 0.00				CM = 0.00			0.90 (0.94)
270	0.8	-	-	0.40	-	0.90	-	-	-	-	-	-	-	-	-	
		PUCT = 0.90 (0.94)							OD = 0.00				CM = 0.00			0.90 (0.94)
280	6.4	0.30	-	0.20	0.20	0.20	-	-	0.70	-	-	-	0.80	-	-	
		PUCT = 0.40 (0.64)							OD = 0.70				CM = 0.80			0.90 (0.98)
2A0	15.1	-	-	-	-	-	-	-	-	0.50	-	-	-	-	-	
		PUCT = 0.00							OD = 0.50				CM= 0.00			0.50
310	98.0	-	-	0.10	-	-	-	-	-	-	0.90	-	0.30	-	-	
		PUCT = 0.10							OD = 0.90				CM = 0.30			0.90 (0.94)
320	40.0	-	-	0.10	-	-	-	-	-	-	0.90	-	0.50	-	-	
		PUCT = 0.10							OD = 0.90				CM = 0.00			0.90 (0.96)
330	2.4	-	-	0.10	-	-	-	-	-	-	0.90	-	0.40	-	-	
		PUCT = 0.10							OD = 0.90				CM = 0.40			0.90 (0.95)
410	2.0	-	-	-	-	-	-	0.90	-	-	-	0.20	0.10	-	-	
		PUCT = 0.90							OD = 0.20				CM = 0.10			0.90 (0.93)
420	1.5	-	-	-	-	-	-	0.40	-	-	-	0.40	0.10	0.40	-	
		PUCT = 0.40							OD = 0.40				CM = 0.50 (0.46)			0.60 (0.81)
430	1.2	-	-	-	-	-	-	0.90	-	-	-	0.20	0.10	-	-	
		PUCT = 0.90							OD = 0.20				CM= 0.10			0.90 (0.93)
440	0.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
		PUCT = 0.00							OD = 0.00				CM = 0.00			0.00
		(Note: visual inspection during execution of BIT provides 0.8 fault coverage)														
450	2.0	-	-	-	0.30	0.30	-	-	-	0.30	0.30	-	0.50	-	-	
		PUCT = 0.30 (0.51)							OD = 0.30 (0.51)				CM = 0.50			0.50 (0.88)
460	2.5	-	-	-	-	-	-	-	-	-	-	0.30	-	0.30	-	
		PUCT = 0.00							OD = 0.30				CM = 0.30			0.30 (0.51)
TOTAL	201.7															171.5 (182.1)

NOTE: (.) figures ignore test overlap. Other figures based on FMECA [1] and engineering judgement.

individual tests (e.g., P.1.1) in Table 3 are based on the PDIU FMECA [1] and engineering judgement. Each coverage figure represents the fraction of times a failure in that particular fault class will be detected by that particular test.

5. PREDICTION OF FFD

For each fault class in Table 3, an overall coverage figure is given for each high-level test group (i.e., PUCT, OD, and CM), and for the total of all of the diagnostic tests. These figures, are based on the FMECA and engineering judgement, and take into account the overlap that exists for some individual tests and fault classes. For comparison, a figure is given in parentheses (if it is different) that is calculated using the probabilistic methodology of [2], assuming all test outcomes are independent (i.e., no test overlap), i.e., $p(a+b)=p(a)+p(b)-p(a)p(b)$. (Note that the figure assuming independence is always larger, and that in general the two numbers are similar for the total of all of the tests for a particular fault class.) The last row in the table indicates the sum of the failure rates of all of the PDIU fault classes (201.7 FPMH), and the estimated failure rate detected by the PDIU diagnostic tests computed using the two methods discussed (171.5 FPMH and 182.1 FPMH). Thus, the estimated FFD for the PDIU self-test feature is 85 percent (90 percent if we do not account for test overlap).

A.3 PDIU DEMONSTRATION PLAN

This subsection consists of the Demonstration Test Plan for the PDIU self-test feature. The primary purpose of this subsection is to *illustrate* the application of the techniques developed in this effort, organized according to the guidelines presented in Section 7. To support this purpose, this subsection is *self-contained.* Therefore, all section, subsection, figure, and table numbers are internally consistent, but they are not consistent with the other portions of this report.

DEMONSTRATION PLAN FOR THE PDIU SELF-TEST FEATURE

CONTENTS

CONTENTS (continued)

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 PURPOSE

This document is a description of the Demonstration Plan (DP) for verifying the self-test feature of the Prognostic Diagnostic Interface Unit (PDIU). The PDIU is a 80186-based system with analog and digital measurement features, data communication interfaces, and provisions for interaction with the soldier/operator conducting the test. The PDIU has no fault isolation requirements, but the results of its diagnostic tests can facilitate fault isolation by a trained technician.

The intention of this DP is to demonstrate to the Government that the self-test capability of the PDIU has been constructed in accordance with its fault detection and isolation design requirements [1 - 2] (references in this DP refer to those listed in subsection 1.6). A description of the PDIU and an overview of its self-diagnostic capability is given in [3].

Currently, General Electric uses the procedures in Section 3 (normal operating mode) to perform acceptance testing of the PDIU prototype hardware for the Army.

1.2 SCOPE

This DP defines all demonstration procedures that are required to completely exercise the PDIU's self-test in fault-free and fault-inserted conditions. It includes a system acceptance test which verifies the PDIU software and hardware interfaces, and individual tests of each PDIU function. This DP is designed to encompass digital VLSI testability issues and will therefore be limited to testing only the PDIU's digital functions. The analog functions/circuits and their diagnostic tests will not be demonstrated. In addition, to keep scope of the demonstration within contract fund limits, the fault universe will consist only of faults in the non-external-interface portions of the PDIU processor, memory, and I/O boards (all components except the 1553 chip-set and the RS-422 transceiver components). There will be two types of demonstration procedures used: procedures that verify the correct operation of a fully operational PDIU, and procedures that

include insertion of known faults to demonstrate and verify the fault detection capability of the PDIU. Fault demonstrations will be accomplished by hardware and software fault insertion.

In this plan a fault refers to the condition when one or more functions on the PDIU processor, memory, or I/O boards fail to meet its performance specification. Faults are reported to the operator by two means: the illumination of the BIT Failure Indicator Lamp and any error message displayed on the Display and Control Unit (DCU), which monitors and controls the operation of the PDIU via the 1553 bus, under normal operating conditions. A list of these messages is shown in Table 1. Under normal (no-fail) operations the DCU displays the message "PDIU is OK," and the BIT lamp is off. When the PDIU self-test detects a fault, either the BIT lamp is on or an error message is displayed on the DCU screen, or both. The definition of a diagnostic error is when the PDIU self-test fails to detect and report a failure, or when it produces a false alarm (e.g., the BIT lamp remains on or a DCU error message is displayed during normal operation). Also, a fault is considered detected if the PDIU does not respond at all on power up. The tests under consideration in this DP are primarily software-based tests, except for the Vcc monitor on the I/O board (see [1] for details).

1.3 OVERVIEW OF THE DEMONSTRATION PROCEDURE

The demonstration will be a one-day, four- to eight-hour demonstration to be conducted at General Electric Automated Systems Department's Burlington, MA, facilities in the presence of representatives from ALPHATECH, Inc., and the Government (RADC).

The objective of this demonstration is to verify that the PDIU's self-test capability satisfies its functional design requirements in a fault-free and a fault-inserted system environment. Faults in the PDIU will be detected by three tests/monitors: the Power-Up Confidence Test (PUCT), the On-Demand/On-Event (OD) tests, the Continuous Monitoring (CM) mode. These tests are functional tests and report the operational status of the PDIU. Testing is conducted on the functional units of the PDIU at the unit test level and system test level by exercising various functions and

TABLE 1. DCU ERROR MESSAGES AND ERROR CODES FOR THE PDIU

NUMBER	NAME	MESSAGE
0	All Tests	00 - All tests passed, PDIU OK XX - One or more tests failed
1	Processor Test	00 - Test passed 01 - Test failed Note: this test is only run at power-up
2	RAM Test	00 - Test passed 01 - Bad SRAM 02 - Bad 1553 SRAM Note: this test is only run at power-up
3	Timer Test	00 - Test passed 01 - Microprocessor timer failed 02 - I/O Board timer failed 03 - Both timers failed Note: this test is only run at power-up
4	PROM Test	00 - Test passed 01 - PROM checksum failed

comparing the results of the tests with their expected values. If no faults are detected, the tests will declare the PDIU as operational. More detail of the PDIU's self-diagnostic capability is located in [1] and [3].

The demonstration will perform PUCT, OD, and CM during normal operational (no failures) and fault-inserted modes as detailed below. An engineering prototype of the PDIU will be used for the demonstration to facilitate the procedure.

1.3.1 No-Fail Demonstration Procedure

Prior to the PDIU demonstration, the PDIU will undergo board test to verify that it is fully operational. For this demonstration, board test will be a comprehensive functional test (more extensive than the self-test) of the PDIU digital hardware, via a board-test connector and an Intel board tester.

The correct operation of the diagnostic capability of the PDIU when the system is fully operational will then be demonstrated. This procedure will execute all of the PDIU's self-test

system software and verify that the tests performed by this software do not detect any errors and that no false alarms are generated.

For this demonstration, any diagnostic errors (false alarms) will be investigated and recorded as described in subsection 1.3.3.

1.3.2 Fault Insertion Demonstration Procedure

Faults will be inserted one at a time. Faults will be inserted using both hardware (i.e., physically inserted faults) and software (e.g., loading digital registers with erroneous data) methods (see Appendix B for hardware insertion techniques). Faults will be selected from the fault-classes that the PDIU self-diagnostics were designed to detect in accordance with the purpose of this plan (to verify compliance with design requirements). These fault classes are described in [4]. Both primary and alternate fault insertion procedures are provided as part of this plan for use as described below in Section 4.

Fault insertion demonstration will consist of two phases. First, all primary faults will be inserted individually. All PDIU self-diagnostics elements will be executed in sequence for each fault insertion, and the diagnostic outcomes (BIT lamp status, DCU message, and PDIU response) will be recorded. The fault insertion procedure will be deemed successful if either the BIT lamp remains on, a DCU error message is displayed, or the PDIU fails to respond to commands (since this provides operator recognition of a fault). If none of these outcomes occur, a diagnostic error has occurred and will be handled in accordance with subsection 1.3.3.

In the second phase, the Government will randomly select faults from the alternate fault list. The maximum number of faults in this phase will not exceed the number of fault classes represented by the primary fault insertion list. Diagnostic errors occurring in this phase will be treated as outlined in subsection 1.3.3.

Unless otherwise instructed by the Government, PUCT must be performed after each fault insertion procedure. If PUCT fails after the fault is supposedly removed, then the steps outlined in subsection 1.3.3 will be performed to determine whether a diagnostic error has occurred.

1.3.3 Diagnostic Error Handling

A diagnostic error occurs when the PDIU self-test capability responds incorrectly to a demonstration procedure. For this plan, an incorrect response is either indication of a fault (by any of the means detailed in subsection 1.2) when the PDIU is fully operational, or no indication of a fault when a known fault is inserted or emulated.

Should this demonstration uncover any diagnostic errors, the following steps will be taken. If a diagnostic error occurs during the demonstration, the first step will be to repeat the procedure on which the error occurred. The primary reason for repeating the procedure is to ensure that the demonstration test operator is not the source of the error. Should the diagnostic error be repeated, and operator error has been eliminated as the source, evaluation of the demonstration unit is required to determine the nature of the error. Depending on the nature of the diagnostic error, evaluation of the hardware and/or software may be required to assess how and where the problem is arising. If an unintentional fault (one that is not meant to be inserted) is found to be the source of the diagnostic error, then the fault will be removed and the demonstration unit will be re-tested, to verify its integrity. Note, if this unintentional fault occurs during supposedly "fault-free" conditions and is detected, we will score the unintentional fault as a successful fault-insertion test. On the other hand, if a testability design error is found to be the cause of the error, it will be reported as a diagnostic error (if the self-test feature was meant to detect it).

Normally, provisions would be made for correcting and re-testing failed demonstration test steps, but this test plan has none since this is outside the scope of this feasibility demonstration. For example, typically, if a diagnostic error occurs during fault insertion testing, the error will be fixed and the offending insertion step will be repeated. Furthermore, the Government would be allowed to select at most three additional fault insertions from the same fault class to be demonstrated.

1.3.4 Acceptance Criteria

The PDIU self-diagnostics feature will be deemed acceptable in any of the following conditions:

1) No diagnostic errors occurred.

2) All diagnostic errors that occurred initially were rectified and all subsequent additional demonstrations resulted in no errors (rectification will not be attempted for this effort).

3) An acceptable plan for:

   a) investigating, during operational test and/or initial fielding, the impact each error has on overall diagnostic performance, and

   b) correcting the problem if the error has a large impact has been submitted.

1.4 RESPONSIBILITIES

1.4.1 Contractor Responsibilities

General Electric is the responsible contractor for the PDIU. It will be responsible for physically operating the PDIU in accordance with the requirements of this document and is responsible for keeping a log of activities and providing a summary of the results in accordance with the requirements of this document.

1.4.2 Procuring Agency Responsibilities

The Government will be required to select alternate faults for the second phase of the fault insertion demonstration and evaluate the acceptance criteria. If rectification of diagnostic errors were part of this plan, the Government would also be responsible for selecting additional faults to be inserted within the offending fault class.

1.5 SUPPORT REQUIREMENTS

Table 2 indicates the specific test equipment (or its equivalent) required for testing the PDIU. Personnel requirements include a system software engineer, a hardware engineer, and a hardware technician to participate in the demonstration and provide any necessary technical information.

TABLE 2. DEMONSTRATION HARDWARE REQUIREMENTS

Part Number	Vendor	Quantity	Description
HIPSQA01	Quality	1 set	Floppy disks containing test software
HIPSQA02	Quality	1	Floppy disks containing test software
3465A	HP	2	Digital Multimeter with probes
-	Intel	1 probe	I2ICE Development System with 80186 emulation
6291A	HP	2	Power Supply, Variable (40V, 5A)
-	HP	1	Precision Voltage Source
-	HP	1	Storage Oscilloscope
-	GenRad	1	Decade Resistor
SBA100F	Loral	1	1553 Bus tester and monitor
VT220	DEC	1	ASCII terminal, screen and keyboard
-	GE-ASD	1	PDIU Engineering Development Test (EDT) Unit
-	GE-ASD	2	Power Cables
-	GE-ASD	2 sets	1553 Bus terminators and 1553 Bus connectors
-	GE-ASD	1	PDIU-to-Serial, Communication Converter Cable
-	GE-ASD	1	RS422-to-RS232 Serial Communication Converter
-	GE-ASD	1	Serial Communication Converter-to-Terminal Cable
-	GE-ASD	1 set	Floppy disks containing additional demonstration test software (generated specifically for this demonstration and not under quality control)

1.6 APPLICABLE DOCUMENTS

1. *System Hardware Specification for Critical Item Development of the PDIU*, GE Document Number 2722630, revision 2, 10 November 1986.

2. *Program Performance Specification for the PDIU*, GE Document Number 12562789, 30 March 1989.

3. "Test Effectiveness Analysis Report for the Prognostic Diagnostic Interface Unit," subsection A.2 in *Analysis and Demonstration of Diagnostic Performance in Modern Electronic Systems*, Final Report on Contract F30602-88-C-0068, TR-494, ALPHATECH, Inc., March 1991.

4. "Guidelines for Preparation of a Demonstration Plan," Section 7 in Analysis and Demonstration of Diagnostic Performance in Modern Electronic Systems, Final Report on Contract F30602-88-C-0068, TR-494, ALPHATECH, Inc., November 1990.

5. MIL-STD 1686, *Electrostatic Discharge Control Program for the Protection of Electrical and Electronic Parts, Assemblies and Equipment.*

6. DoD-HDBK 263, *Electrostatic Discharge Control Handbook for the Protection of Electrical and Electronic Parts, Assemblies and Equipment.*

7. *MATE Guidelines*, Document Number 2806645, Rev B, 1 April 85. Applicable to Acceptance Test and Verification of Testability, subsections 3.7.4 – 3.7.9.

2. COMPLIANCE DATA

This demonstration plan is fully compliant with the comprehensiveness requirements stated in [4] as follows:

2.1 NUMBER OF FAULTS

The number of fault classes for which the PDIU self-diagnostics has been designed to detect faults is 17 (see Table 3). The number of primary inserted and emulated faults required by [4, subsection 2.1] to achieve a 95% confidence in a 90% FFD with zero diagnostic errors is 32. A total of 36 primary faults will be either inserted or emulated in this plan. In addition, the Government will also select up to 35 faults from the alternate list in accordance with subsection 1.3.2 of this plan.

2.2 PERCENTAGE OF FAULT CLASSES DEMONSTRATED

This is the percentage of fault classes in which faults are inserted or emulated. The entire universe of faults may be larger than the classes used here since the requirement is only for demonstrating the fraction of those classes that the PDIU self-diagnostic capability is designed to detect. The requirement as stated in [4, subsection 2.2] is 80 percent. Out of the 17 fault classes that the PDIU is designed to detect (see [3]), the demonstration procedures in Section 4 cover 11 of those classes, for a percentage of 65 percent. (Note that as stated in subsection 1.2, we have restricted this demonstration to the PDIU's digital functions, of which there are 16 fault classes. Thus, the percentage of these relevant fault classes demonstrated is 69 percent. Note also that if this were an actual demonstration plan versus an illustration, this figure would not be acceptable)

2.3 PERCENTAGE OF THE FAILURE RATE DEMONSTRATED

This is a two-part requirement. First is the ratio of the sum of failure rates of the demonstrated fault classes to the sum of the failure rates for those fault classes that the PDIU

TABLE 3. COMPREHENSIVENESS OF PDIU FAULT INSERTIONS

Fault Class	Number and Type of Insertions*	Component/Component Groups	Failure Rate (FPMH)	Reason for No Insertion**
Processor Board				
210	6FE	Microprocessor and Support Electronics	10.5	
220	–	Local Bus (Address Latch and Control Signal Buffer)	1.5	A
230	2FI	Station Bus (Address Latch, Data and Control Transceiver)	4.8	
240	–	Buffer/Transceiver Control Circuitry (PAL & Glue Logic)	2.0	A,B
250	N/A	Test Buffer	NI	
260	1FI/3FE	SRAM (32K × 16)	12.8	
270	1FI/3FE	1553 SRAM (2K × 16)	0.8	
280	2FI	PROM	6.4	
290	N/A	RS-422 Interface	NI	
2A0	–	1553 Remote Terminal Interface	15.1	C,D
2B0	N/A	1553 Transformers	NI	
Memory Board (Each, Two Total)				
310	3FI, 2FE	EEPROM Memory Devices (24–8K × 8)	48.0	
320	–	Decode and Control	20.0	A
330	3FI	Station Bus and DTACK Generator	1.2	
340	N/A	Testability Transceiver	NI	
I/O Board				
410	3FI, 3FE	Programmable Counter/Timer	2.0	
420	–	Decode and Control Interface	1.5	A,D
430	1FI	Station Bus and DTACK Generator	1.2	
440	2FI	BIT Lamp Driver	0.2	
450	1FI	Vcc Monitor	2.0	
460	–	A/D Converters	2.5	A,D
470	NA	Filters	NI	
480	NA	Multiplexers	NI	

*Key:

FI	Hardware Fault Insertion
FE	Software Fault Emulation
N/A	Not Applicable–Purposely Not Addressed in PDIU Self-Test Feature Design
NI	Not Included

** Key:

A	No Physical Access (Unsocketed)
B	No Software Access (Component Not Software Addressed)
C	Unavailable Support Equipment Required
D	Emulation Requires New/Modified BIT Software

self-diagnostic capability is designed to detect. The requirement stated in [4, subsection 2.3] is 90 percent. The total failure rate for the PDIU-detectable fault classes is 201.7 faults per million hours (FPMH), while the failure rate for the classes demonstrated is 143.1 FPMH, for a percentage of

class failure rate demonstrated of 71 percent. The second part of the requirement is the ratio of the sum of the failure rates of those failure types for which the specific faults demonstrated are representative to the sum of the failure rates for the fault classes that the PDIU self-diagnostic capability is designed to detect. The requirement stated in [4, subsection 2.3] is 30 percent. Appendix C presents the manner in which the figure was calculated for this demonstration—20.50 percent. (Because this demonstration is merely an illustration of the approach, the requirements of [4] need not be met.)

2.4 PERCENTAGE OF DIAGNOSTIC FEATURES DEMONSTRATED

The PDIU self-test consists of PUCT, OD, and watchdog timer software-based tests and two voltage monitoring hardware-based tests. The software-based tests are all single string, so that by initiating their execution all tests are run. Since all software- and hardware-based tests are demonstrated, 100 percent of the PDIU features are demonstrated. This complies with the 100 percent requirement in [4, subsection 2.4] and 85 percent requirement of [4, subsection 2.5], respectively.

2.5 DIAGNOSTIC OUTCOMES DEMONSTRATED

The PDIU has only fault detection requirements. Therefore the correct diagnostic outcomes are either that the PDIU correctly detects the presence of a fault or that it does not generate false alarms when no faults are present. Faults are reported to the operator by the illumination of the PDIU BIT lamp and error messages on the DCU. For the PDIU there are two outcomes:

a) Under no-fault conditions: BIT lamp off and DCU message "PDIU OK"

b) Under fault-inserted/fault-emulated conditions: BIT lamp on and DCU error messages (e.g., "PDIU NOT OK")

Outcomes are operator recognition of either: a) when the PDIU is fault-free, or b) when the PDIU has a fault inserted/emulated. Outcome a) will be demonstrated once in Section 3 and between every fault insertion/emulation step in Section 4. Outcome b) will be demonstrated for every fault

insertion step in Section 4. The requirement of 100 percent demonstration of all diagnostic outcomes is therefore satisfied.

3. DETAILED DEMONSTRATION PROCEDURE FOR OPERATIONAL SYSTEM

These steps describe the steps to validate the proper operation of the tests when applied to a fully-operational PDIU.

3.1 TEST SETUP

Prior to the demonstration, the PDIU will have successfully completed board test to verify that the PDIU is fully operational. In addition, its self diagnostic test software has been loaded into its non-volatile memory (Processor Board PROM, 280) using the I2ICE development system. Unless otherwise specified all tests will be performed at 25 +/-10 degrees C. Record results on the Result sheet of this test step. Figure 1 illustrates the proper demonstration setup.

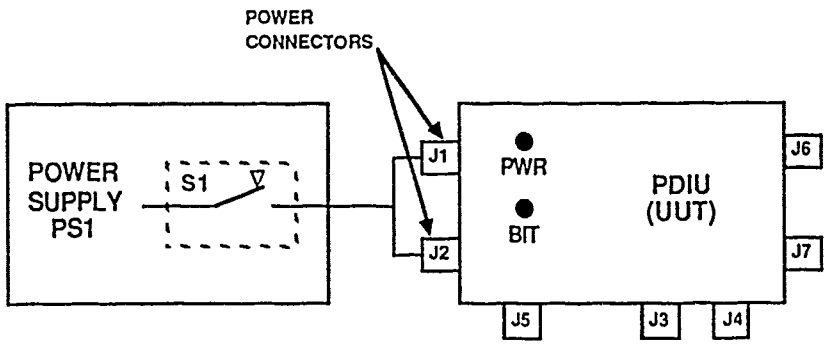


Figure 1. Demonstration Setup

The test setup is performed as follows:

- Verify that the power switch is in the open position.
- Adjust input power supply to 28V +/-0.5V, no current limit.
- Verify that the POWER and BIT lamps on the UUT are illuminated.
- With a DVM, verify that the following volta[illegible]e within specified limits on J5 of the UUT (PDIU Test Connector)

Signal Name	(+)lead of DVM	to	Signal Name	(-)lead of DVM	Limits
28OUT	(J5-M)		28RETOUT	(J5-N)	27.00V to 29.00V
+5P	(J5-A)		SGND	(J5-D)	4.85V to 5.25V
+15P	(J5-B)		SGND	(J5-D)	14.45V to 15.45V
-15P	(J5-C)		SGND	(J5-D)	-15.45V to -14.45V

- Adjust the input voltage, PS1 to 18V +/-0.1V
- With a DVM, verify that the proper voltages (5P,+15P,-15P) are present on the J5 connector of the UUT. Use voltage limits specified above.
- Adjust the input voltage, PS1 to 33V +/-0.1V
- With a DVM, verify that the proper voltages (5P, +15P,-15P) are present on the J5 connector of the UUT. Use voltage limits specified above.
- Return the input voltage, PS1, to 28V +/-0.5V.
- Disconnect the Cable to J1 of the UUT.
- Verify that the proper voltages are present on the J5 connector of the PDIU. Use the voltage limits as specified above.
- Reconnect the Cable to J1 of the UUT.
- Disconnect the Cable to J2 of the PDIU.
- Verify that the proper voltages are present on the J5 connector of the PDIU. Use the voltage limits as specified above.
- Reconnect the Cable to J2 of the UUT.
- Open Switch, S1.

3.2 DEMONSTRATE PUCT/OD TESTS IN A FULLY OPERATIONAL PDIU

PUCT for the PDIU consists of a number of tests, which validate the proper operation of the following hardware functions:

Processor Instruction Set Test

Processor Register Test

Processor Address Test

RAM Test (64K SRAM and 4K of shared (1553) memory)

Timer Tests (three timers on the Processor board and three timers on the I/O board)

A complete description of PUCT and OD tests is given in [3]. After successful completion of the PUCT, the OD tests are then performed. This step continues to validate the proper operation of the PDIU. Specific OD tests performed are:

Processor PROM test

1553B self-test

Memory board EEPROM test

I/O board analog data acquisition test

At the completion of these tests, if an error condition is detected, the PDIU will illuminate the BIT lamp. These tests are invoked immediately after a power-up reset. The results of the PUCT will be summarized in the PUCT Status code word (Table 4) and will be recorded in EEPROM. The results will be reported to the operator via the BIT lamp.

With the test configuration setup as in Fig. 1, the test steps are as follows:

- Close switch S1
- Verify that the POWER and BIT lamps on the UUT are illuminated immediately upon power up.
- Verify that the BIT lamp is extinguished within approximately five seconds.

  (The order of the previous two steps ensures that the BIT lamp is functional and that the PDIU's processor and I/O boards are fully operational.)
- Record test results

3.3 DEMONSTRATE THE WATCHDOG TIMER MONITOR FOR A FULLY OPERATIONAL PDIU

After PUCT is successfully repeated, the watchdog timer is initialized and activated by system software. Thus, during the OD tests, if the watchdog timer did not timeout and cause a system reset (indicated by the BIT lamp being illuminated), then the timer has demonstrated proper functionality. This self-test feature will not be explicitly demonstrated. Instead, the proper operation of its functionality will be inferred from the successful demonstrations of PUCT and the

TABLE 4. PUCT STATUS CODES

Bit 0	Contains the result of the Processor Test
Bit 1	Contains the result of the RAM Test
Bit 2	Contains the results of the 1553 RAM Test
Bit 3	Contains the results of the Timer Test, I/O Counter/Timer 0
Bit 4	Contains the results of the Timer Test, I/O Counter/Timer 1
Bit 5	Contains the results of the Timer Test, I/O Counter/Timer 2
Bit 6	Contains the results of the Timer Test, Microprocessor Timer 0
Bit 7	Contains the results of the Timer Test, Microprocessor Timer 1
Bit 8	Contains the results of the Timer Test, Microprocessor Timer 2
Bits 9 - 15	Are not used
Convention of bit values	'0' means a passed test '1' means a failed test

OD tests. Note, this feature has been thoroughly tested and demonstrated during the formal sell-off of the PDIU and its application software.

3.4 DEMONSTRATE THE VCC MONITOR FOR A FULLY OPERATIONAL PDIU

The steps for this demonstration are the following:

- Disconnect power to the PDIU. Verify that the POWER lamp is not illuminated.
- Remove the I/O board. Insert an extension card and replace the I/O board on the end of the extension board.
- Apply power to the PDIU, verify that the POWER and BIT lamps are illuminated. Verify that the BIT lamp is extinguished after approximately five seconds.
- With a DVM verify that the voltage across pins A-9 (DGND_SENSE) and B-9 (+5V_SENSE) of the I/O board connector is greater than +4.63 V (the low threshold limit of the Vcc Monitor).
- Remove power to the PDIU. Remove the I/O board and the extension card. Replace the I/O.

4. DEMONSTRATION PROCEDURE FOR FAULTY SYSTEM STATES

This subsection provides detailed instructions for performing fault insertion demonstrations on the PDIU. The demonstrations are divided into two separate subsections: hardware fault insertion and software fault emulation. Each subsection provides a template for the fault insertion or fault emulation process. This is then followed by a list of selected fault classes to be demonstrated. Within each fault class, specific instances are selected as primary demonstrations. A description of alternate demonstration candidates within each fault class, for selection by the Government if necessary (see subsection 1.3.3), is also provided. Appendix B provides guidance on the fault insertion techniques to be used in this demonstration

4.1 HARDWARE FAULT INSERTION

This demonstration is being performed using the PDIU engineering prototype to facilitate broader fault insertion demonstration tests. The PDIU prototype is more amenable to fault insertions because it contains socketed components that the production unit is not permitted to contain (due to environmental specifications). In general, the faults inserted for this demonstration will be accomplished by the following means:

- Open leads or remove socketed components.
- Generate shorts between accessible nodes with jumpers.
- Use of extension cards to produce faults on the board/backplane connector.
- Produce shorts, opens or faulty signal behavior at the PDIU 1553 Bus Interface.

Appendix B discusses the specific techniques to be used in creating these faults.

4.1.1 Fault-Insertion Demonstration Procedure Template

The following steps are to be followed in demonstrating each of the hardware fault insertion tests in this demonstration plan. The specific faults to be inserted are given in subsection

4.1.2. It is assumed that the PDIU has undergone board test and has successfully demonstrated its self-diagnostics as in Section 3.

A. PRETEST INPUTS

1) Verify that the PDIU power switch (S1) is in the OFF position. Verify that the POWER and BIT lamps are not illuminated.

2) Power up the PDIU by placing the power switch in the ON position. PUCT/OD Tests will be performed automatically as specified in subsection 3.2. Verify that the POWER and BIT lamps are illuminated upon power up. Verify that the BIT lamp is extinguished within approximately 5 seconds after power up. If the BIT lamp remains illuminated, then the PDIU may be faulty and must be repaired before continuing the fault insertion demonstration. If a fault insertion demonstration was performed immediately before this step, then it will be deemed invalid and must be repeated. See subsection 1.3 and Part E of this subsection for details on how this situation will be reported and what steps will be followed if it is deemed that the PDIU is not faulty (a diagnostic error).

3) Turn off power to the PDIU by placing the power switch in the OFF position. Verify that both the POWER and BIT lamps are extinguished.

B. FAULT INSERTION

Note, printed circuit boards and integrated circuits (ICs)/components may be sensitive to electrostatic discharge (ESD). To prevent ESD damage, the components should be handled in accordance with ESD prevention procedures (see [5] and [6] for details). While the fault is being physically inserted, the boards/components shall not be powered.

1) Locate the board on which the fault is to be inserted in the PDIU test cage. Carefully remove the board from the PDIU and place it on an ESD grounding mat.

2) Locate the proper component on the board according to the circuit card assembly drawings (see Appendix D). Remove the device from its socket. Locate the proper pin on the component and perform the desired fault insertion technique as described in Appendix B. Carefully re-insert the device with only its known fault onto the board.

3) Re-insert the board back into its original slot in the PDIU card cage.

C. DEMONSTRATION OF PDIU SELF-TEST FAULT DETECTION PERFORMANCE

Fault detection is facilitated by the PDIU self-diagnostics through the BIT lamp and through DCU error messages. For PUCT/OD tests demonstration, the BIT lamp and any DCU

error message indicate that a fault is detected. No demonstration of the PDIU's CM fault detection capability is provided in this plan. The specific steps are as follows.

1) Turn on power to the PDIU. PUCT/OD tests will be initiated automatically upon power up.

   a) Verify that the POWER and BIT lamps are illuminated upon power up.

   b) Observe and record BIT lamp status after approximately 5 seconds (upon power up) and whether any error message is displayed on the DCU.

2) Interrogate the EEPROM for BIT status (optional). This will indicate which test detected the fault.

3) Turn off power to the PDIU by placing the power switch in the OFF position. Verify that the POWER and BIT lamps are extinguished.

D. REMOVAL OF INSERTED FAULT AND VERIFICATION OF ITS REMOVAL

1) Follow the steps in part B to locate the PDIU board containing the known fault, remove board, remove the affected device.

2) Remove/repair the inserted fault on the component, or if the fault insertion was destructive, obtain a new component with the same part number.

3) Re-insert the non-faulty component onto the proper PDIU board, and then re-insert the board into the proper slot of the PDIU test card cage.

4) Power up PDIU. PUCT/OD tests will be performed automatically. Verify that the POWER and BIT lamps are illuminated. Verify that the BIT lamp is extinguished within approximately 5 seconds (upon power up) and no error messages are displayed on the DCU) and that no other faults have been accidentally inserted.

Note, Step 4 is a repeat of Step 2 in Part A. If either the BIT lamp remains on or a DCU error message is displayed, the PDIU may still be faulty and must be repaired before continuing (see subsection 1.3 and Part E, below, of this plan). Execution of PUCT and OD tests for verifying that the PDIU is fully operational need only be done once between each fault insertion. After Part D is complete, another fault is selected and Parts B-D will be repeated.

E. DEMONSTRATION GRADING CRITERIA

1) During Part A (or D), if PUCT/OD tests detect and report any errors, then the PDIU may be faulty and must be repaired. A detailed examination of the PDIU will be made to find and repair the fault. If no fault can be found, and upon re-execution of PUCT and OD tests, the PDIU status is declared operational, then a false alarm will be recorded in Section 5. But before continuing this demonstration, the PDIU must be

returned to operational status. When the PDIU has been repaired and returned to a fully operational state, the previous demonstration step will be repeated.

2) If PUCT/OD tests indicate the presence of fault(s) after removal of an inserted fault (Part D), then the previous fault-insertion demonstration step is deemed invalid and must be repeated after the PDIU is repaired. This is because additional/unintentional fault(s) may have been introduced during either the fault insertion or the fault removal process, and therefore it is impossible to ascertain whether the particular demonstration was valid. It is immaterial whether PUCT or the OD tests detected a fault during Part C, because they could have detected the unintentional fault.

3) If the inserted fault was not detected and reported to the operator during Part C, then the demonstration step is deemed a failure (a diagnostic error has occurred). This failing result will be reported in Section 5 of this plan. In addition, as outlined in subsection 1.3.3 of this plan, for each diagnostic error of this type the Government will select between 1 and 3 additional faults from the same fault class for additional demonstration. Note, normally such a diagnostic error would required examination of the PDIU self-test feature for possible software or hardware changes to correct the error. This analysis is beyond the scope of this illustrative demonstration plan.

4) If PUCT/OD Tests detected a fault during Part C and no faults were detected after the fault was removed (during part D), then the demonstration step is deemed a success. This result will be reported in Section 5.

4.1.2 Fault Classes and Primary and Alternate Faults To Be Inserted

This subsection indicates the fault classes whose failure is to be inserted in the demonstration. Primary faults are those that will be inserted by the contractor, while alternate faults are available for insertion at Government request. The fault class numbers correspond to those shown in Table 3.

4.1.2.1 PROCESSOR BOARD

The Processor Board is located in slot 3 of the PDIU.

4.1.2.1.1 Station Bus (230)

a) Multiple Open Pins on the Station Data Bus Transceiver

Primary Fault: Remove component U37, using Technique 5 from Appendix B.

Alternate Faults: Disconnecting any data pin (pins 11-18) on U37 is allowed.

b) Short Between Input Pins of the Station Address Bus Transceiver

Primary Fault: Short address pins 6 and 9 of U34 together using Technique 7.

Alternate Faults: Shorting any two address pins (pins 2, 5-6, 9, 12, 15-17, 19) together is allowed.

4.1.2.1.2 SRAM (260)

a) Short Adjacent Address Pins

Primary Fault: Short address pins 1 and 3 together on SRAM U10 using Technique 7.

Alternate Faults: Shorting any other two address pins together on SRAM U10 is allowed.

4.1.2.1.3 1553 SRAM (270)

a) Data Pin Stuck at High

Primary Fault: Short data pin 19 on SRAM U6 to +5V (pin 24, the Vcc input) using Technique 3.

Alternate Faults: Shorting any other data pin (pins 11-19) on SRAM U6 to +5V is allowed.

4.1.2.1.4 PROM (280)

a) Data Pin Stuck at Low

Primary Fault: Short data pin 15 of U7 to 0.0V (pin 14) using Technique 1.

Alternate Faults: Shorting any data pin (pins 11-13, 15-19) of U7 to 0.0V is allowed.

b) Open Input (address) Pin

Primary Fault: Remove address pin 23 on U7 using Technique 6.

Alternate Fault: Removing any address pin (pins 1-10, 21, 23-26) on U7 is allowed

4.1.2.2 MEMORY BOARD

Faults will be inserted on Memory Board #2, which is located in slot 5 of the PDIU.

4.1.2.2.1 EEPROM Memory Device (310)

a) Multiple Open Pins

Primary Fault: Remove EEPROM memory device, U21 using Technique 5.

Alternate Faults. Removal of another EEPROM Device (U10) from the memory board is allowed.

b) Data Pin Stuck at Low

Primary Fault: Short data pin 13 to ground (pin 14) on EEPROM memory device U21 using Technique 1.

Alternate Faults: Shorting of any other data pin (pins 11-13, 15-19) on either socketed EEPROM memory device (U10 or U21) is allowed.

c) Address Pin Stuck at High.

Primary Fault: Short address pin 2 to +5V (pin 28) on EEPROM memory device U21 using Technique 3.

Alternate Faults: Shorting of any other address pin (pins 1-10, 21, 23-26) on either socketed EEPROM memory device (U10 or U21) is allowed.

4.1.2.2.2 Station Bus (330)

a) Open Data Pin between the Data Transceiver and the On-board Circuits

Primary Fault: Remove data pin 15 on U33 using Technique 6.

Alternate Fault: Removal any other data pin of data transceiver U34 connected to on-board circuits is allowed.

b) Short Data Pins on the Station Bus Backplane

Primary Fault: Short data pins D6 and D7 (C24 and C25) on slot 5 backplane using Technique 7.

Alternate Faults: Shorting any other data (C18-C33), address (B18-B36) or control (A18-21, A24, A28, C10-11) connector pins on the slot 5 backplane is allowed.

c) Output Pin Stuck at High on the Address Latch

Address Line is Stuck at High.

Primary Fault: Short output pin 2 to +5V (pin 20) on address latch U36 using Technique 3.

Alternate Faults: Shorting of any other output pin (pins 2 - 9) on address latch U36 is allowed.

4.1.2.3 I/O BOARD

The I/O board is located in slot 2 of the PDIU.

4.1.2.3.1 Programmable Counter/Timer (410)

a) Multiple Open Pins

Primary Fault: Remove component U5 using Technique 5.

Alternate Fault: There are no alternate faults for this step.

b) Input Pin Stuck at High

Primary Fault: Remove pin 1 (CLK0 data input) of U5 and apply +5V (pin 24) to the vacant pin socket.

Alternate Fault: Removal of clock output pin (10, 13, 17) of U5 and application +5V to the vacant pin socket is allowed.

c) Input Pin Stuck at Low

Primary Fault: Short timer input data pin 1 on U5 to 0.0V (pin 12) using Technique 1.

Alternate Faults: Shorting timer input data pin (1 - 8) to 0.0V is allowed.

4.1.2.3.2 Station Bus (430)

a) Short between Data Pins on the Control Register (for BIT)

Primary Fault: Short data pins 13 and 14 together on U23 using Technique 7.

Alternate Faults: Shorting any two data pins (11 - 18) together on U23 is allowed.

4.1.2.3.3 BIT Lamp Driver/Status Indicator (440)

a) Input Pin Open

Primary Fault: Remove data input pin 18 of status register U23 using Technique 6.

Alternate Faults: Removing of data input pin 18 of status register U23 and shorting it to +0V using Technique 2.

b) Output Pin Stuck at Low

Primary Fault: Remove data output pin 19 of status register U23 and apply 0V to the vacant socket.

Alternate Faults: Removal of data input pin 19 of status register U23 and the application of +5V.

4.1.2.3.4 Vcc Monitor (450)

a) Open Pin

Primary Fault: Disconnect DGND_SENSE (A9) on the I/O board edge connector using Technique 8.

Alternate Faults: Shorting the +5V_SENSE input to 0V using Technique 1 is allowed.

4.2 FAULT EMULATION DEMONSTRATIONS

Fault emulation is defined as a fault that is inserted via software. Once inserted, this fault manifests the physical behavior of an actual fault. There are two techniques whereby faults will be inserted into the PDIU via the in-circuit emulator (I2ICE):

A) Modify the expected value of the test to yield an "error" when the test is executed under fully operational conditions. Test conditions: before the test is performed, modify the expected values. Execute the test; a "failure" should be detected by either PUCT or the OD tests. Stop test, correct the expected value, and retest the PDIU.

B) At pre-determined breakpoints, the execution of the test is halted, erroneous data is loaded into the microprocessor's registers (or other addressable/programmable devices such as the RAM, EEPROM, or programmable counter/timer), and the execution of the test is resumed. This process is performed by application software that inserts these faults.

4.2.1 Fault-Emulation Demonstration Procedure Template.

It is assumed that the PDIU has undergone board test and has successfully demonstrated its self-diagnostics as described in Section 3. To perform in-circuit emulation, the 80186 microprocessor on the PDIU's processor board must removed (note, it is socketed) and the emulator probe placed in the vacant socket. It is via this probe that the emulator can control the system operation of the PDIU and directly read addressable memory/register contents.

A. PRETEST INPUTS

1) Verify that the PDIU power switch (S1) is in the OFF position. Verify that the POWER and BIT lamps are not illuminated.

2) Power up the PDIU by placing the power switch in the ON position. PUCT/OD tests will be performed automatically as specified in subsection 3.2. Verify that the POWER and BIT lamps are illuminated upon power up. Verify that the BIT lamp is extinguished within approximately five seconds after power up, ensuring that the BIT lamp is functional and that the PDIU's processor and I/O boards are operational. Also verify that no error messages are displayed on the DCU. If any error message is reported, then the PDIU is deemed faulty and will be repaired. If a demonstration step was performed immediately before this step, then it will be deemed invalid (see Part E for further explanation).

B. FAULT EMULATION

1) Use the I2ICE to load fault emulation software into the PDIU. For technique A, this software will activate breakpoints in the test software and will prompt the operator to modify the expected values used by this software. For technique B, this software will

activate breakpoints in the PUCT and OD software and will insert faults (e.g., erroneous data, read from incorrect memory locations, etc.).

2) Observe and respond to any messages requiring operator input.

C. DETECTION OF KNOWN FAULT–TEST OF SELF-DIAGNOSTICS

1) Initiate PUCT/OD tests.

   a. Verify that the POWER and BIT lamps are illuminated.

   b. Observe and record BIT lamp status after approximately 5 seconds upon initiating PUCT and whether any error message is displayed on the DCU.

2) Interrogate the EEPROM for BIT status (optional). This will indicate which test detected the fault.

D. REMOVAL OF KNOWN FAULT AND VERIFICATION OF ITS REMOVAL

1) Use the DCU to reload fault free software into the PDIU. Wait for transfer complete message to appear on the display of the I2ICE.

2) Turn off power to the PDIU by placing the power switch in the OFF position. Verify that the POWER and BIT lamps are extinguished.

3) Power up PDIU. PUCT/OD tests will be performed automatically. Verify that the POWER and BIT lamps are illuminated that the fault has been removed. Verify that the BIT lamp is extinguished within approximately 5 seconds upon power up and no error messages are displayed on the DCU). This step ensures that no other faults have been accidentally inserted.

E. DEMONSTRATION GRADING CRITERIA

1) During Part A, if PUCT or OD tests detect and report any errors, then the PDIU is deemed faulty, and it will be repaired. When the PDIU has been repaired and returned to a fully operational state, the complete demonstration step will be repeated.

2) If after the inserted fault has been removed and another fault is detected upon retest (Part D), then the demonstration step is deemed invalid and must be repeated after the PDIU is repaired. This additional/unintentional fault(s) may have been introduced during the fault insertion or fault removal process, and therefore it is impossible to ascertain whether the particular demonstration was valid. It is immaterial whether PUCT or the OD tests detected a fault during Part C, because they could have detected the unintentional fault.

3) If the inserted fault was <u>not</u> detected and reported to the operator during Part C, then the demonstration step is deemed a failure. This failing result will be reported in the demonstration report.

4) If either PUCT or OD tests detected a fault during Part C and no faults were detected after the fault was removed (during Part D), then the demonstration step is deemed a success. This result will be reported in Section 5.

4.2.2 Fault Classes and Primary and Alternate Faults To Be Emulated

This subsection indicates the fault classes whose failure is to be emulated in the demonstration. Primary faults are those that will be emulated by the contractor, while alternate faults are available for emulation at Government request. The fault class numbers correspond to those shown in Table 3.

4.2.2.1 PROCESSOR BOARD

4.2.2.1.1 Microprocessor (210)

a) Register Failure

Primary Fault: During processor register test activate breakpoint 5 using Technique B. Prompt the operator to change the current contents of register AX.

Alternate Faults: Repeat the fault emulation, and allow the operator to insert any other four-digit hexadecimal number.

b) Addressing Mode Failure/Data Fault

Primary Fault: During processor indirect addressing mode test activate breakpoint 11 using Technique B. Prompt the operator to change the value pointed to by register DS.

Alternate Faults: Repeat the fault emulation, and allow the operator to insert any other four digit hexadecimal number.

c) Addressing Mode Failure/Address Line or Register Fault

Primary Fault: During processor indirect addressing mode test activate breakpoint 11 using Technique B. Prompt the operator to change the value of register SI.

Alternate Faults: Repeat the fault emulation, and allow the operator to insert any other four digit hexadecimal number.

d) Flag/Jump Command Failure

Primary Fault: During processor test of the Jump if Zero Flag, activate breakpoint 31 using Technique B. Prompt the operator to replace the current contents of register SP (0001H) to 0000H, whereby resetting the flag (emulates data bit 1 stuck at 0).

Alternate Faults: Activation of breakpoint 30 and alteration of the contents of SI register to 0110H (the expected value for the comparison) using technique A is allowed.

e) Instruction Set Failure [could also be use to emulate fault in processor board PROM (280)]

Primary Fault: During processor test of the ADD instruction, activate breakpoint 57 and change the contents of AX register to 5885H to yield a result different than expected using technique A.

Alternate Fault: Perform test as above, but alter AX to any number other than 5555H.

f) Processor Internal Timer Failure

Primary Fault: During the processor test of its own internal timers, use technique A to modify the expected value of the test. This change should cause the detection of a counting error.

Alternate Fault: Repeat the fault emulation with any other number.

4.2.2.1.2 SRAM (260)

a) Data Fault

Primary Fault: During the RAM test activate breakpoint 24 using Technique B. Prompt the operator to replace the current contents of any RAM location.

Alternate Faults: Activation of the same breakpoint and alteration of the memory contents to any other value.

b) Address Fault

Primary Fault: During the RAM test activate breakpoint 24 using technique B. Prompt the operator to replace the value of register BP, containing the next RAM location to be tested. Resume test.

Alternate Fault: Repeat same test, but changing the contents to a different value.

c) Leaking Memory Fault

Primary Fault: During the RAM test activate breakpoint 24 using technique B. Change the value in the next location to be examined by writing to two locations instead of one. Resume test.

Alternate Fault: Repeat same test at a different location in RAM.

4.2.2.1.3 1553 SRAM (270)

Repeat the fault emulations described in subsection 4.2.2.1.2, but for the 1553 SRAM locations.

4.2.2.2 MEMORY BOARD

4.2.2.2.1 EEPROM Memory Device (310)

a) Data Fault

Primary Fault: Interrupt the processing at the point where the ROM checksum is checked, and alter the previously stored checksum.

Alternate Fault: Repeat the emulated fault, except enter a different number.

b) Address Fault

Primary Fault: Interrupt the processing at the point where the ROM checksum is about to be calculated, and alter the calculation's start point.

Alternate Faults: Repeat the emulated fault using a different erroneous start point.

4.2.2.3 I/O BOARD

4.2.2.3.1 Programmable Counter/Timer (410)

a) Data Fault

Primary Fault1: Interrupt testing of I/O Timer 0 using Technique B. Initialize counter 0 to return Binary Coded Decimal instead of Binary.

Alternate Fault1: Repeat the fault emulation on another timer.

Primary Fault2: Interrupt testing of I/O Timer 2 using Technique B. Change the value returned by counter 2.

Alternate Fault2: Repeat the fault emulation with a different changed value.

b) Control Register Fault

Primary Fault: Interrupt testing of I/O Timer 1 using Technique B. Read counter 2 instead of counter 1.

Alternate Fault: Repeat, but read counter 0.

4.3 FAULT CLASSES NOT DEMONSTRATED

This section includes only those fault classes that are tested by the PDIU's self-diagnostics (see Table 3) but are not included in the demonstration.

Processor Board

Local Bus (220)

Buffer/Transceiver Control Circuitry (240)

1553 Remote Terminal Interface (2A0)

Memory Board

Decode and Control (320)

I/O Board

Decode and Control (420)

A/D Converter (460)

SP/MUX Board

Since there are no test requirements for this board, none of its electronics are tested.

5. DEMONSTRATION RESULTS

The demonstration was conducted on 23 February 1990 with a Government representative present, and on 26 February 1990 with only ALPHATECH and GE-ASD personnel attending. A total of 44 primary and alternate faults were demonstrated over the two days. No false alarms were encountered in the demonstration.

All 17 primary emulation faults and four alternate emulation faults were correctly detected by the PDIU. The alternate emulation faults chosen were 4.2.2.1.1b, 4.2.2.1.2a, 4.2.2.1.3b, and 4.2.2.3.1a2. A no-fail condition was also demonstrated via the emulation procedure, with no failure registered by the PDIU.

All 19 primary insertion faults and four alternate insertion faults were demonstrated. The alternate insertion faults chosen were 4.1.2.1.1a, 4.1.2.1.2a, 4.1.2.1.3a, and 4.1.2.1.4a. All four alternate faults were correctly detected by the PDIU. Of the 19 primary inserted faults, only 16 were correctly detected by the PDIU. These three insertions were each repeated at least once, with identical results. Primary fault 4.1.2.3.1b, an input pin stuck high on the I/O Board Programmable Counter/Timer (410), was detected but not reported, i.e., the internal fault register was correctly set but the BIT lamp did not stay lit. Primary faults 4.1.2.3.2a and 4.1.2.3.3a were not detected at all. The former was a short between data pins on the control register for BIT on the I/O Board Station Bus (430), while the latter was an input pin open on the I/O Board Lamp Driver/Status Indicator (440).

Resources did not permit investigating the causes of these errors. The fact that all errors occurred on the I/O Board certainly provides focus for any subsequent analysis. The MATE Guidelines [7] indicate that for three errors out of 44 random, independent trials, there is 90 percent confidence that FFD is at least 85 percent or, equivalently, there is 70 percent confidence that FFD is at least 90 percent. The Test Effectiveness Analysis Report [3] indicates that the

estimated FFD was 82 percent if test overlap was accounted for, and 88 percent if it was not. These figures appear to be consistent.

Because this not an actual acceptance demonstration but an illustrative example, no plans for correcting or mitigating these diagnostic errors are required.

APPENDIX A

DEFINITIONS

Fault	The condition where one or more functions on the PDIU processor, memory, or I/O board fails to perform as designed.
Diagnostic Error	The condition where the PDIU self-test fails to detect and report a failure, or when it produces a false alarm.
Inserted Fault	In general, this term refers to either hardware- or software-inserted faults. Occasionally we use this term in context in a more restricted sense to refer only to a fault that is physically inserted (via hardware) and can be verified by observation.
Emulated Fault	A fault that is inserted via software using an in-circuit emulator (ICE). Once inserted, such a fault must manifest the physical behavior of an actual fault.
Test	In this document, a PDIU self-test: the Power-Up Confidence Test (PUCT), the On-Demand/On Event tests, and the Continuous Monitoring mode.

APPENDIX B

HARDWARE FAULT INSERTION TECHNIQUES

Power must be turned off to the PDIU before any fault can be inserted (although shorting pins together may be inserted after initialization, if it is more convenient). To insert a fault, first determine on which board of the PDIU the given fault is to be inserted, and carefully remove this board from the PDIU. Once the board is out the fault may be inserted, and the board put back into the PDIU. At this point the PDIU can be safely turned on and the system is initialized. Faults to be inserted into the PDIU (see subsection 4.1.2) generally fall into 10 categories, and the steps for inserting each are described below.

1) Short a pin to 0.0 volts (ground).

   This fault consists of using a shorting probe lead to connect a given component pin to ground. Connect one end of the lead to ground, and connect or hold in place the other end to its respective pin. When manually holding or connecting the probe be sure that the probe is not touching any pins other than the given one, because grounding certain pins and voltages could cause damage to the affected component. The faults listed have been selected to prevent damage to any part of the PDIU.

2) Remove a pin and short it to 0.0 volts.

   This is similar to the above, except that the specific pin to be grounded must be "removed" from the board. This fault has only been selected for those components that are socketed; i.e., the component is connected to the board via a socket rather than soldered directly to the board. Thus, no unsoldering of components is necessary, as this could cause damage to the component. To insert this fault pry the chip out of the socket and carefully bend the appropriate pin upwards about 90 degrees. Carefully replace the component with the bent pin out of the socket. It may be necessary to use an insulator (small piece of paper) to ensure that the bent pin makes no electrical contact with its socket. From here follow the directions for shorting a pin to ground to complete the fault insertion.

3) Short a pin to +5.0 volts.

   This is identical to shorting a pin to ground, except that the probe is connected to a +5.0 V voltage source. The board's +5.0 V supply may be used for this +5.0 V source.

4) Remove a pin and short it to +5.0 volts.

This is a combination of removing a pin (see type 2) and shorting to +5.0 volts (see type 3 above), so the steps already listed may be employed here.

5) Remove a component from the circuit board.

   This is self-explanatory. The component must be removed before power up to the test set. Only socketed components may be selected for this fault, again to minimize the possible damage from unsoldering.

6) Remove a pin.

   This refers to removing a pin from a socketed component as described in 2 above without connecting it to ground or +5.0 volts.

7) Short two pins together.

   Connect two pins together with a wire jumper.

8) Disconnect certain board edge connector pin(s).

   The board pins refer to the row of metal pins on the bottom edge of each circuit board, which make the connections to the rest of the PDIU. To "disconnect" any of these pins, remove the requested circuit board (make sure that the power is off) and find the appropriate pin(s) along the bottom edge. Once the pin has been located, disconnect the corresponding wire that is connected to the board edge connector. A board extender may be necessary to implement this fault.

9) Short pins together on a connector.

   This refers to shorting the appropriate pins on either the 1553 or RS-422 connector. A jumper wire may be used on the connector to provide the short.

10) Short board pins together.

    Similarly to 9 above, a jumper wire may be use to short pins located on the board connector.

APPENDIX C

FAILURE RATE OF PDIU FAULT INSERTIONS

C.1 INTRODUCTION

In this appendix we calculate the failure rate contribution of the various individual primary and alternate fault insertions discussed in Section 4. The procedure is as follows. First, we calculate the relative contribution of the particular hardware or software insertion to the overall failure rate of its fault class, based on the number of components, pins, etc. in each fault class and engineering judgement regarding the relative contribution of the failure mechanism represented by the insertion to the total component failure rate. Next, we multiply the sum of these relative contributions for each fault class by the failure rate for the fault class (shown in Table 3). Finally, we add these fault class figures. We will now proceed by demonstrated fault class, where the fault class number is given in parentheses after the fault class name).

C.2 MICROPROCESSOR (210)

Register Failure Emulation

We estimate that register faults account for 5 percent of the total microprocessor faults. Since there are 16 registers, the single primary register fault insertion yields a relative contribution of .0032. (Note: we will assume that all primary data is accurate to 4 significant figures, and maintain that significance on derived quantities throughout.)

Addressing Mode Failure/Data Fault Emulation

We estimate that this fault mode accounts for 5 percent of the total microprocessor faults. Since there are 16 bits and 8 addressing modes, the insertion of both a primary and secondary fault yields a relative contribution of .0008.

Addressing Mode Failure/Address Line or Register Fault Emulation

We estimate that this fault mode accounts for 1 percent of the total microprocessor faults. Since there are 20 address lines, this single primary insertion yields a relative contribution of .0005

Flag/Jump Command Failure Emulation

We estimate that this fault mode accounts for 1 percent of the total microprocessor faults. There are 92 instructions, and 30 of them have 4 flags each. We calculate the relative contribution of this single primary insertion as (30/92)(1/4)(.01)=.0008.

Instruction Set Failure Emulation

We estimate that this fault mode accounts for 1 percent of the total microprocessor faults. Since there are 92 instructions, this single primary insertion yields a relative contribution of .0001.

Internal Timer Failure Emulation

We estimate that this fault mode accounts for .05 percent of the total microprocessor faults. Since there are three internal timers, this single primary insertion yields a relative contribution of .0002.

Summary

From the above calculations, we see that for the microprocessor we have demonstrated examples of fault modes that account for 13.05 percent of the total microprocessor failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for .56 percent of the total microprocessor failures. Using the smaller figure, and the microprocessor failure rate of 10.5 FPMH, the microprocessor insertions account for 0.0588 FPMH.

C.3 STATION BUS (230)

Multiple Open Data Bus Transceiver Pin Insertion

We estimate that this fault mode accounts for 20 percent of the total station bus faults. Since there are six transceivers, and both a primary and alternate fault of this type are inserted, this single insertion yields a relative contribution of .0667.

Adjacent Address Bus Transceiver Pin Short Insertion

We estimate that this fault mode accounts for 30 percent of the total station bus faults. Since there are six transceivers, and we are shorting two out of 9 pins, this yields a relative contribution of (.3)(2! 7!/9!)(1/6) = .0014.

Summary

From the above calculations, we see that for the station bus we have demonstrated examples of fault modes that account for 50 percent of the total station bus failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 6.81 percent of the total station bus failures. Using the smaller figure, and the station bus failure rate of 4.8 FPMH, the station bus insertions account for 0.3269 FPMH.

C.4 SRAM (260)

Short Adjacent Address Pin Insertion

We estimate that this fault mode accounts for 30 percent of the total SRAM faults. Since we are shorting two out of 12 address lines, and both a primary and alternate fault of this type are inserted, this yields a relative contribution of (.3)(2! 10!/12!)(2) = .0091.

Data Fault Emulation

We estimate that this fault mode accounts for 40 percent of the SRAM faults. Since there are 16 data pins, and both a primary and alternate fault of this type are inserted, this yields a relative contribution of (.4)(1/16)(2) = .05.

Address Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total SRAM faults (the same figure used in the insertion discussed above). Since there are 12 address lines, this single primary insertion yields a relative contribution of (.3)(1/12) = .0250.

Memory Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total SRAM faults. Since all of the memory locations are checked by the diagnostics, this single primary insertion yields a relative contribution of .3.

Summary

From the above calculations, we see that for the SRAM we have demonstrated examples of fault modes that account for 100 percent of the total SRAM failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 38.41 percent of the total SRAM failures. Using the smaller figure, and the SRAM failure rate of 12.8 FPMH, the SRAM insertions account for 4.916 FPMH.

C.5 1553 SRAM (270)

Data Pin Stuck-At Insertion

We estimate that this fault mode accounts for 40 percent of the total 1553 SRAM faults. Since there are 9 data pins, and both a primary and alternate fault are inserted, this yields a relative contribution of .0889.

Data Fault Emulation

We estimate that this fault mode accounts for 40 percent of the total 1553 SRAM faults (the same figure as used immediately above). Since there are 9 data pins, this single primary insertion yields a relative contribution of .0444.

Address Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total 1553 SRAM faults. Since there are 12 address lines, and both a primary and alternate fault are inserted, this yields a relative contribution of (.3)(1/12)(2) = .05.

Memory Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total 1553 SRAM faults. Since all of the memory locations are checked by the checksum diagnostics, this single primary insertion yields a relative contribution of .3.

Summary

From the above calculations, we see that for the 1553 SRAM we have demonstrated examples of fault modes that account for 100 percent of the total 1553 SRAM failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 48.33 percent of the total 1553 SRAM failures. Using the smaller figure, and the 1553 SRAM failure rate of 0.8 FPMH, the 1553 SRAM insertions account for .387 FPMH.

C.6 PROM (280)

Data Pin Stuck-At Insertion

We estimate that this fault mode accounts for 40 percent of the total PROM faults. Since there are 8 data pins, and both a primary and alternate fault are inserted, this yields a relative contribution of (.4)(1/8)(2) = .1.

Open Address Pin Insertion

We estimate that this fault mode accounts for 30 percent of the total PROM faults. Since there are 15 address pins, this single primary insertion yields a relative contribution of .02.

Summary

From the above calculations, we see that for the PROM we have demonstrated examples of fault modes that account for 70 percent of the total PROM failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 12 percent of the total PROM failures. Using the smaller figure, and the PROM failure rate of 6.4 FPMH, the PROM insertions account for .768 FPMH.

C.7 EEPROM (310)

Multiple Open Fault Insertion

We estimate that this fault mode accounts for 20 percent of the total EEPROM faults. Since there are 24 devices, this single primary insertion yields a relative contribution of (.2)(1/24) = .0083.

Data Pin Stuck-At Insertion

We estimate that this fault mode accounts for 20 percent of the total EEPROM faults. Since there are 24 devices and 8 data pins, this single primary insertion yields a relative contribution of (.2)(1/8)(1/24) = .0010.

Address Pin Stuck-At Insertion

We estimate that this fault mode accounts for 30 percent of the total EEPROM faults. Since there are 15 address lines, this single primary insertion yields a relative contribution of (.3)(1/15) = .02.

Data Fault/Corrupted Memory Emulation

We estimate that this fault mode accounts for 30 percent of the total EEPROM faults. Since each memory location is checked by the checksum diagnostics, this single primary insertion yields a relative contribution of .3.

Address Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total EEPROM faults (the same figure used for the address pin insertion discussed above). Since there are 15 address lines, this single primary insertion yields a relative contribution of (.3)(1/15) = .02.

Summary

From the above calculations, we see that for the EEPROM we have demonstrated examples of fault modes that account for 100 percent of the total EEPROM failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 34.93 percent of the total EEPROM failures. Using the smaller figure, and the total EEPROM failure rate (for two boards) of 96 FPMH, the EEPROM insertions account for 33.53 FPMH.

C.8 STATION BUS (330)

Open Data Pin Insertion

We estimate that this fault mode accounts for 20 percent of the total station bus faults. Since there are 16 data lines, this single primary insertion yields a relative contribution of (.2)(1/16) = .0125.

Short Adjacent Data Pin Insertion

We estimate that this fault mode accounts for 20 percent of the total EEPROM faults. Since there are 16 data lines, this single primary insertion yields a relative contribution of (.3)(2! 14!/16!) = .0025.

Output Stuck-At Fault Insertion

We estimate that this fault mode accounts for 20 percent of the total EEPROM faults. Since there are 8 output pins, this single primary insertion yields a relative contribution of (.2)(1/8) = .025.

Summary

From the above calculations, we see that for the station bus we have demonstrated examples of fault modes that account for 60 percent of the total station bus failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 4 percent of the total station bus failures. Using the smaller figure, and the total station bus failure rate (for two boards) of 2.4 FPMH, the station bus insertions account for .096 FPMH.

C.9 PROGRAMMABLE COUNTER/TIMER (410)

Multiple Open Pin Insertion

We estimate that this fault mode accounts for 20 percent of the total programmable counter/timer (PCT) faults. This single primary insertion yields a relative contribution of .20.

Input Pin Stuck-At Insertions

We estimate that this fault mode accounts for 40 percent of the total PCT faults. Since there are 8 pins, and there is a primary insertion high and a different primary insertion low, this yields a relative contribution of (.4)(1/8)(2) = .10.

Data Fault Emulations

We estimate that this fault mode accounts for 10 percent of the total PCT faults. Since there are 3 timers, each with 8 bits, and two primary and one alternate faults are inserted, this yields a relative contribution of (.1)(1/3)(1/8)(3) = .0125.

Control Register Fault Emulation

We estimate that this fault mode accounts for 30 percent of the total PCT faults. Since there are 16 bits in the register, this single primary insertion yields a relative contribution of (.3)(1/16) = .0188.

Summary

From the above calculations, we see that for the PCT we have demonstrated examples of fault modes that account for 100 percent of the total PCT failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for 33.13 percent of the total PCT failures. Using the smaller figure, and the PCT failure rate of 2.0 FPMH, the PCT insertions account for .663 FPMH.

C.10 STATION BUS (430)

Control Register Data Pin Short Insertion

We estimate that this fault mode accounts for 20 percent of the total PCT faults. Since there are 16 pins, this single primary insertion yields a relative contribution of (.2)(2! 14!/16!) = .0017.

Summary

From the above calculations, we see that for the station bus we have demonstrated examples of fault modes that account for 20 percent of the total station bus failures. However, in terms of the contribution of the specific demonstrated faults, and not the fault mode classes they represent, we have demonstrated faults that account for .17 percent of the total station bus failures. Using the smaller figure, and the station bus failure rate of 1.2 FPMH, the station bus insertions account for .002 FPMH.

C.11 BIT LAMP DRIVER (440)

Open Input Pin Insertion

We estimate that this fault mode accounts for 25 percent of the total BIT lamp driver faults. Since there is only a single pin, this single primary insertion yields a relative contribution of .25.

Output Pin Stuck-At Insertion

We estimate that this fault mode accounts for 25 percent of the total BIT lamp driver faults. Since there is only a single pin, this single primary insertion yields a relative contribution of .25.

Summary

From the above calculations, we see that for the BIT lamp driver we have demonstrated examples of fault modes that account for 50 percent of the total BIT lamp driver failures. In terms of the contribution of the specific demonstrated faults, we have also demonstrated faults that account for 50 percent of the total BIT lamp driver failures. Using this figure, and the BIT lamp driver failure rate of 0.2 FPMH, the BIT lamp driver insertions account for .10 FPMH.

C.12 VCC MONITOR (450)

Open Pin Insertion

We estimate that this fault mode accounts for 25 percent of the total Vcc monitor faults. Since there is only a single ground line, this single primary insertion yields a relative contribution of .25.

Summary

From the above calculations, we see that for the Vcc monitor we have demonstrated examples of fault modes that account for 25 percent of the total Vcc monitor failures. In terms of the contribution of the specific demonstrated faults, we have also demonstrated faults that account for 25 percent of the total Vcc monitor failures. Using this figure, and the Vcc monitor failure rate of 2.0 FPMH, the Vcc monitor insertions account for .50 FPMH.

C.13 SUMMARY OF FAILURE RATE DEMONSTRATED

Table 5 indicates a summary of the failure demonstrated by the primary and alternate faults demonstrated. The total failure rate demonstrated is 41.35 FPMH. Comparing this figure with the failure rate for the PDIU-detectable fault classes of 201.7 FPMH, gives a percentage of failure rate demonstrated by hardware and software insertions of 20.50 percent.

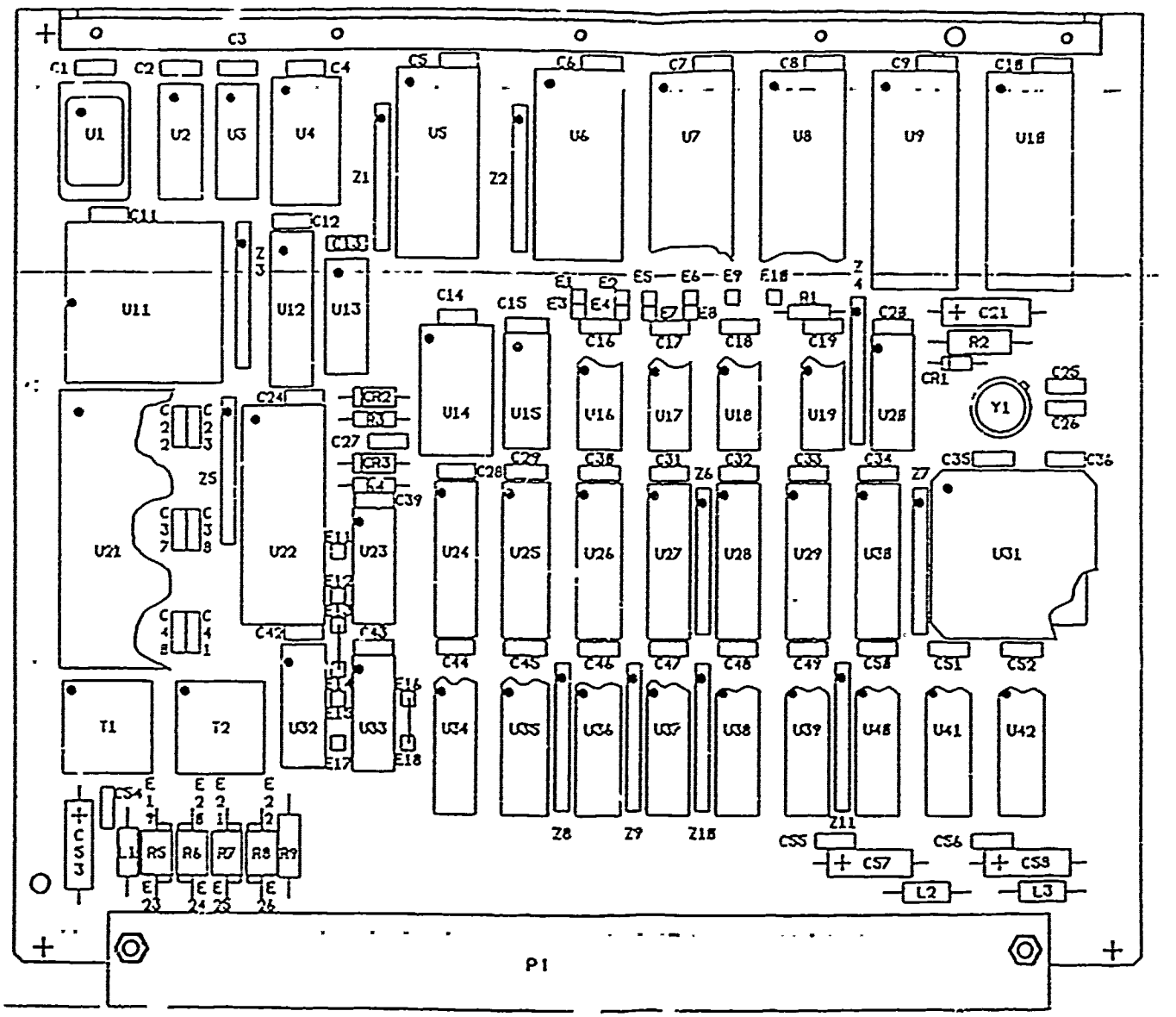
TABLE 5. DEMONSTRATED FAILURE RATE SUMMARY

Fault Class Number	Fault Class Name	Demonstrated Failure Rate (FPMH)
210	Microprocessor	.0588
230	Station Bus	.3269
260	SRAM	4.916
270	1553 SRAM	.387
280	PROM	.768
310	EEPROM	33.53
330	Station Bus	.096
410	Programmable Counter/Timer	.663
430	Station Bus	.002
440	BIT Lamp Driver	.10
450	Vcc Monitor	.50
TOTAL		41.35

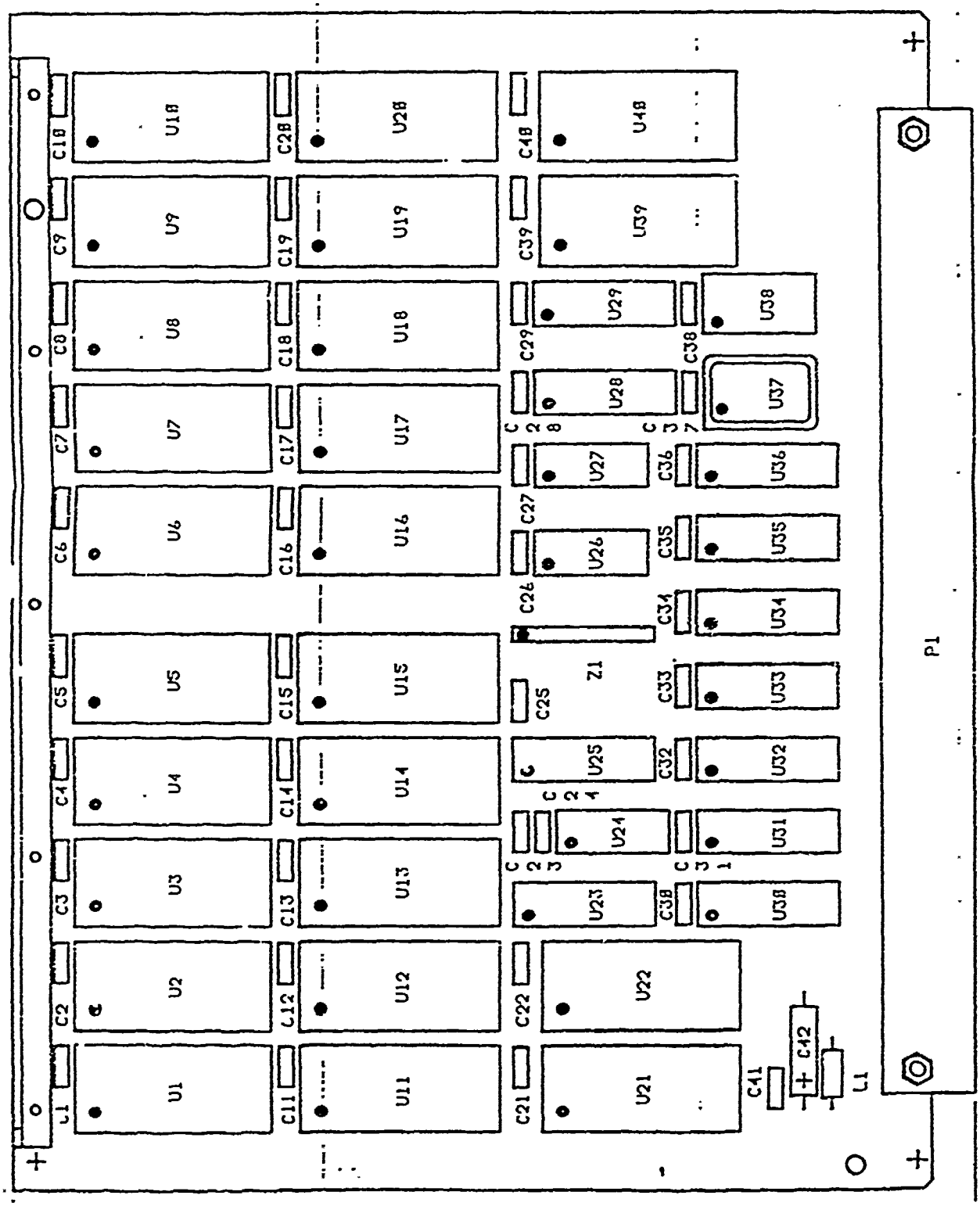
APPENDIX D

PDIU CARD ASSEMBLY DRAWINGS

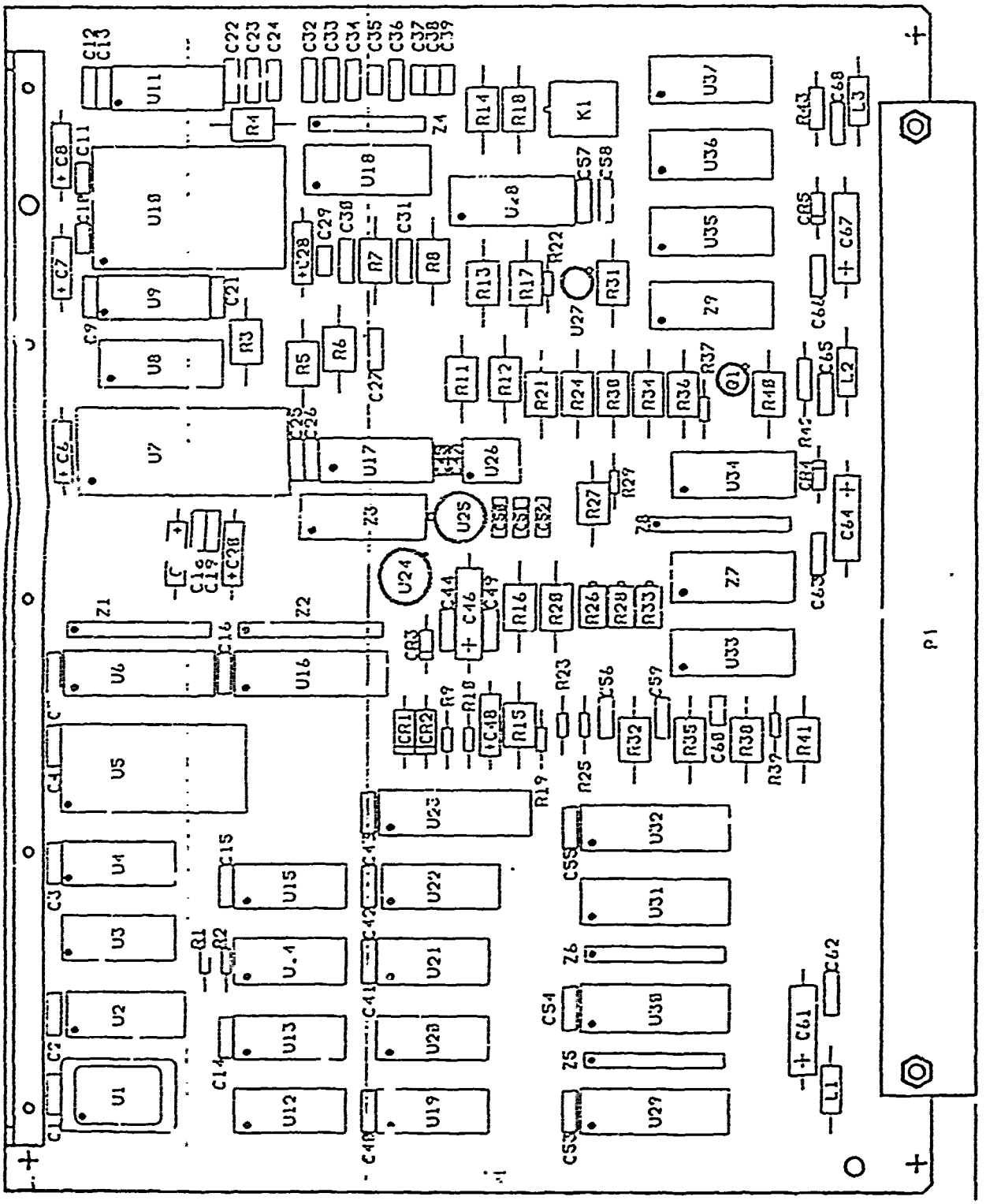
D.1 PROCESSOR BOARD



D.2 MEMORY BOARD



D.3 I/O BOARD



A.4 SUMMARY AND CONCLUSIONS

In this appendix we applied the methodology developed earlier to the validation and demonstration of the self-test feature of the Prognostic Diagnostic Interface Unit. We developed a Test Effectiveness Analysis Report and a Demonstration Plan, using the guidelines of Sections 9 and 7, respectively. We conducted a one and one-half day demonstration in which 21 faults were emulated and 23 faults were physically inserted. All emulated faults were correctly detected, but diagnostic errors were made on three physical fault insertions on the I/O board: two inserted faults were not detected at all, and one was detected but not reported. For three errors out of 44 random, independent trials, there is 90 percent confidence that FFD is at least 85 percent or, equivalently, there is 70 percent confidence that FFD is at least 90 percent. The Test Effectiveness Analysis Report (subsection A.2) indicates that the estimated FFD was 85 percent if test overlap was accounted for, and 90 percent if it was not. These figures appear to be consistent.

The fault emulation process went quite well. Close interaction with the ICE operator gave good visibility into the nature of the emulation errors. The emulation of a no-fault condition, and the resulting proper system response, together with the emulation of observer-selected faults, gave supporting confidence to what is by its very nature a fairly "invisible" process.

On the other hand, the physical fault insertion process took much longer than originally envisioned. This increased time was required primarily to reload memory after most fault insertions, since the majority of these insertions resulted in the PDIU microprocessor's writing over various memory segments. This effect of fault insertions can occur for any system having a microprocessor, and should be explicitly accommodated in the demonstration planning process.

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.