12

RADC-TR-86-241
Final Technical Report
February 1987

BUILT-IN-TEST VERIFICATION TECHNIQUES

Boeing Aerospace Company

Jeffrey H. Albert, Mike J. Partridge and Richard J. Spillman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC 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.

RADC-TR-86-241 has been reviewed and is approved for publication.

APPROVED: Thomas L Fennell

THOMAS L. FENNELL
Project Engineer

APPROVED: Tuthill

W. S. TUTHILL, Colonel, USAF
Director of Reliability & Compatibility

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBER) 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 requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
N/A

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
N/A

5. MONITORING ORGANIZATION REPORT NUMBER(S)
RADC-TR-86-241

6a. NAME OF PERFORMING ORGANIZATION
Boeing Aerospace Company

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
Rome Air Development Center (RBER)

6c. ADDRESS (City, State, and ZIP Code)
P. O. Box 3999
Seattle WA 98124

7b. ADDRESS (City, State, and ZIP Code)
Griffiss AFB NY 13441-5700

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Rome Air Development Center

8b. OFFICE SYMBOL (If applicable)
RBER

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F30602-84-C-0021

8c. ADDRESS (City, State, and ZIP Code)
Griffiss AFB NY 13441-5700

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
62702F	2338	02	1J

11. TITLE (Include Security Classification)
BUILT-IN-TEST VERIFICATION TECHNIQUES

12. PERSONAL AUTHOR(S)
Jeffrey H. Albert, Mike J. Partridge, Richard J. Spillman (Spillman Research Associates)

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM Jan 84 TO Jan 86

14. DATE OF REPORT (Year, Month, Day)
February 1987

15. PAGE COUNT
250

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	05	
14	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Built-In-Test, Built-In-Test Verification,
BIT, Simulation
BIT Verification Verification

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report documents the effort to develop practical verification methodologies for the accurate and economical test and demonstration of Built-In-Test (BIT) subsystems and circuitry. These methodologies are to be applicable to both formal verification of a system for demonstration of BIT effectiveness (test system verification), and under operational conditions, to verify that the BIT is functioning as designed (test system condition assessment).

Currently BIT is verified by analysis of the design or by demonstration of a limited set of simulated faults. Verification of BIT capability by design analysis for verification purposes is not currently recognized as a reliable means of verification. Use of simulated set of faults has also proven to have low effectiveness for evaluation of BIT capabilities, as well as being costly. BIT on equipment in the field can seldom be tested to discover failures in the BIT circuitry, which results in gradual degradation of BIT effectiveness.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Thomas L. Fennell

22b. TELEPHONE (Include Area Code)
(315) 330-3476

22c. OFFICE SYMBOL
RADC (RBER)

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

[illegible]

PREFACE

This report documents the results of the effort for the Rome Air Development Center Contract F30602-84-C-0021, BIT Verification Techniques.

The work was accomplished by the Engineering Technology Organization of Boeing Aerospace Company under the management of T. A. Nicolino, with support provided by Dr. Richard Spillman of Spillman Research Associates. The principal investigators were Mike Partridge and subsequently Jeffrey Albert.

The contract was administered by the Rome Air Development Center. Technical and administrative support was provided by Thomas Fennell. Management was provided by Jerome Klion (RADC/RBET).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

This report documents the results of a two year effort to develop techniques for Built-In Test (BIT) verification. The objective of the contract was to develop specifications and technical details for practical verification methodologies for the accurate and economical test and demonstration of built-in test. This included both Test System Verification (TSV), to verify that BIT is designed to meet its performance requirements, and Test System Condition Assessment (TSCA), to verify during operation that BIT is performing as designed. Current test system verification methodologies have proven to be inadequate and costly for effective evaluation of BIT capability. As a consequence, these factors have contributed to questionable BIT performance in the field.

This contract has addressed potential improvements in TSV techniques, to provide the designer with better tools to design and integrate BIT into a system. It has also addressed development of TSCA techniques, to provide assurance to operators and maintenance personnel that the BIT is operating correctly.

Task 1 investigated current military electronic design technology and projected trends for the near future to determine their impact on BIT design. The following design trends were examined to determine their impact on BIT design:

Computer-Aided Engineering
Artificial Intelligence
Modular Avionics Packaging
Pave Pillar
Integrated Diagnostics
Architectures
Data Buses

General trends in electronic component technology were also examined, with particular emphasis on Very High Speed Integrated Circuits (VHSIC). The conclusions of this task were that no radical changes in technology would occur before 1990, and that the current trends toward increased integrated circuit functionality and increased performance would continue. As integrated circuit technology advances result in increased performance capabilities, the requirement for advanced BIT design at the integrated circuit level should result in more effective BIT. One possible result could be development of BIT chip sets incorporating some smart BIT concepts.

Task 2 surveyed current and proposed TSV/TSCA techniques by reviewing literature and making personal contacts in industry, military and academia. The following techniques were identified:

TEST SYSTEM VERIFICATION
- Figures of Merit
- Failure Modes and Effects Analysis
- Simulation
- Statistical

TEST SYSTEM CONDITION ASSESSMENT
- Self-Checking Circuits
- Fault Insertion

Descriptions of these techniques were prepared and are included in the report. Within each of these categories, specific implementations are numerous and vary greatly. This indicates a need for standardized effective verification techniques.

In Task 3, potential improvements to the techniques identified in Task 2 were investigated. TSV improvements assessed included candidates for new figures of merit, use of test repetition methods for assessing intermittent faults, extensions to failure modes and effects analysis methods and advanced simulation concepts. The investigation of TSCA improvements led to the development of a new concept, Overlapping BIT, which is described in section 7.1.

Task 4 consisted of the evaluation of techniques and potential improvements. An initial filter screened out techniques that did not meet the requirements of being practical, economical and accurate. The primary reasons for elimination at this stage were very limited applicability or high complexity. The remaining techniques were evaluated against several criteria, some of which were qualitative and some which were quantitative. The evaluation resulted in three techniques being selected as the most promising candidates for improvement. These were behavioral simulation for TSV, and overlapping BIT and fault insertion for TSCA.

Task 5 involved further development of the techniques selected during the evaluation task. Behavioral level simulation was developed for TSV, with overlapping BIT and fault insertion developed for TSCA. After this additional development, these techniques continued to show promise for developing into effective verification capabilities.

Investigation into the use of behavioral level simulation for TSV led to the conclusion that the technique shows promise but is not ready today. Improvements in simulation software and in fault modeling, plus increased availabilty of powerful computers, will be necessary before simulation use for TSV in complex, modern systems can be practical on a fairly universal basis. Most of the necessary improvements will evolve naturally over the next 3-5 years, but some stimulus is necessary in the areas of concurrent fault simulation and the relationship of functional fault models to physical faults. A specification for recommended futher development in this area was written as part of Task 5.

The TSCA techniques, overlapping BIT and fault insertion, are applicable in their current form now. For each technique, a control system architecture was developed, and requirements and limitations were identified. For overlapping BIT, several applications (to data buses, memories, analog interfaces and Hamming code extensions) were identified and explored and a specification for its use was developed. For fault insertion, a new device to insert faults was developed. This device uses less hardware and permits insertion of a greater variety of faults than other devices found in the literature. A number of fault insertion output processing concepts were documented, along with their relative strengths in different application contexts. Also, since overlapping BIT and fault insertion are applicable in different situations and they have different strengths and weaknesses, guidelines for their use were developed.

An additional noteworthy finding of this study was that agreement on standardization of methodologies is at least as necessary as finding better techniques. Standardization would eliminate use of the ad hoc methods frequently used now and would make it unnecessary to spend time selecting a method for each new program. It would also result in the application of BIT verification methods being better understood and accepted by users. Promoting user acceptance of integrated approaches to BIT would be in itself a significant achievement.

1.0 INTRODUCTION

This report documents the results of a two year effort to develop practical, effectiv~ methodologies to verify that Built-In Test (BIT) designs meet their requirements (test system verification) and, in operation, perform as designed (test system condition assessment).

1.1 BACKGROUND

Maintenance of weapon systems is becoming an increasingly important consideration in weapon system development. Improvement in the maintenance capability of a weapon system greatly reduces the total life cycle cost of the system since the cost of maintenance is a significant portion of that cost. Improved maintenance also increases availability which, in turn, reduces he number of systems that need to be acquired, lowering the acquisition cost of the weapon system.

With the implementation of MIL-STD-2165, Testability Program for Electronics Systems and Equipments, it will become essential to have the tools to predict and measure the various testability requirements. This standard specifies a program to incorporate testability disciplines into programs from concept exploration phase through production and deployment. Elements from the system level to the integrated circuit component level are affected. Key to successful application of this standard is specification, prediction, measurement and verification of the testability evaluation parameters (e.g., fault detection rate, fault isolation rate and false alarm rate). Experience has shown that adequate verification techniques for these parameters do not currently exist. New techniques will likely use computer-aided tools for verification and assessment methodology application. This would provide standardized, accurate verification at lower cost and make verification techniques available to design engineers at all phases of development.

Numerous techniques have been used for verification of BIT systems. These include manual analysis, computer analysis by modeling and simulated fault insertion, simulated fault insertion in actual hardware and use of real faults by collecting faulty components. Each of these techniques has differing costs associated with their use and the effectiveness of each varies. In general, the greater the cost, the greater the effectiveness. The BIT performance in the field has not, however, been as good as the verification techniques have predicted.

Advances in electronic design technology and new BIT developments will place additional burdens on the verification process. In particular, advances in Very Large Scale Integrated (VLSI) circuit technology present special BIT verification problems. In VLSI devices there are more locations where failures can occur (e.g. due to increased gate count), there are additional failure modes, and the circuit description and failure mechanisms are often not known to the subsystem and system designers.

1.2 OBJECTIVE

The objective of the BIT Verification Techniques effort is to develop the specifications and technical details for practical verification methodologies for the accurate and economical test and demonstration of built-in test. This applies both to Test System Verification (TSV), to verify that BIT is designed to meet its performance requirements, and Test System Condition Assessment (TSCA), to verify during operation that BIT is performing as designed.

The methods selected for development must be practical in that they are generally applicable to a wide range of electronic and BIT system designs and usable by the majority of industry and military users without substantial capital investment. They must verify parameters that are definable in a procurement specification. They must also be accurate so that the verification results reflect what is expected in operation. Finally, the techniques must be economical to use and not require investments in advanced computers and simulation capability beyond the means of most companies.

1.3 PROGRAM PLAN

The BIT Verification Techniques program is structured around five tasks as illustrated in figure 1-1.

Task 1 identified near-term trends in military electronics design technology and investigated their impact on BIT verification. The trends were used to determine where improvements are needed.

In parallel, task 2 surveyed literature as well as government, industry and educational sources to identify TSV and TSCA techniques in use, under development, or proposed for development. Descriptions of these techniques were documented and evaluated for possible improvements as part of task 3.

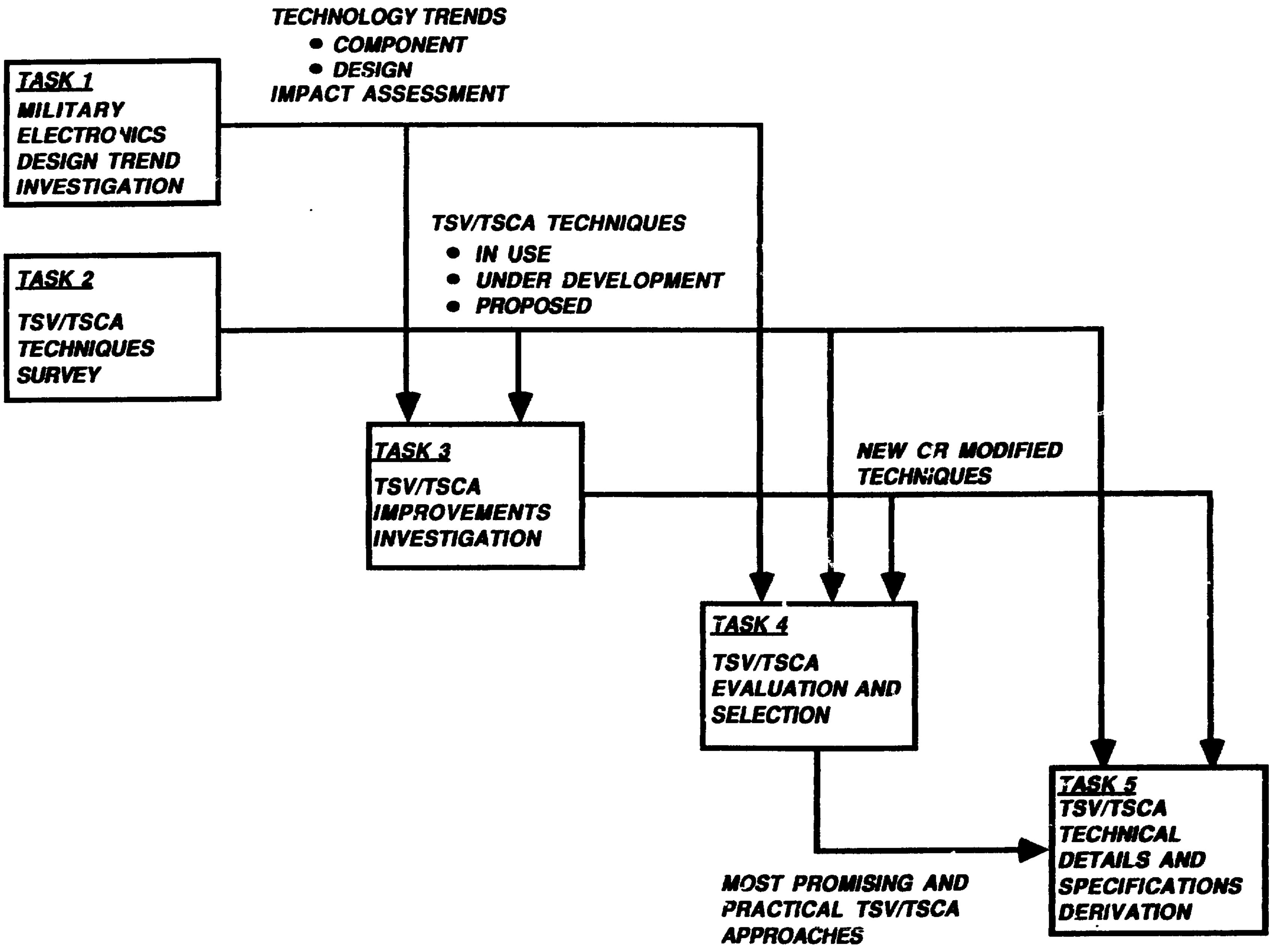

TASK 1
MILITARY ELECTRO\ICS DESIGN TREND INVESTIGATION
TECHNOLOGY TRENDS
• COMPONENT
• DESIGN
IMPACT ASSESSMENT
TASK 2
TSV/TSCA TECHNIQUES SURVEY
TSV/TSCA TECHNIQUES
• IN USE
• UNDER DEVELOPMENT
• PROPOSED
TASK 3
TSV/TSCA IMPROVEMENTS INVESTIGATION
NEW CR MODIFIED TECHNIQUES
TASK 4
TSV/TSCA EVALUATION AND SELECTION
MOST PROMISING AND PRACTICAL TSV/TSCA APPROACHES
TASK 5
TSV/TSCA TECHNICAL DETAILS AND SPECIFICATIONS DERIVATION

FIGURE 1-1 BIT VERIFICATION PROGRAM TASK FLOW

In task 3, the TSV and TSCA techniques identified in task 2 were evaluated to determine if improvements to these techniques needed to be made or could be made to incorporate advances in BIT or electronics technology. Potentially useful new approaches were also identified. The output of task 3 includes the description of TSV and TSCA technique improvements and descriptions of potentially useful new techniques.

In task 4, the evaluation criteria were selected and each TSV and TSCA technique was analyzed with respect to those criteria. The technology trend and impact information from task 1 was used to help develop and weight the criteria. The most promising TSV and TSCA techniques were selected based on the criteria and the evaluations.

In task 5 the recommended TSV and TSCA techniques were developed to provide further technical detail and to derive the necessary specifications to implement the techniques.

1.4 REPORT ORGANIZATION

This report is organized around the individual tasks as indicated in table 1-1. Section 2 contains the results of the military electronic design trends investigation (task 1). Section 3 contains the results of the BIT verification techniques survey (task 2) and includes descriptions of the various BIT verification techniques. Section 4 discusses the TSV and TSCA techniques improvements which were investigated and analyzed as part of Task 3. Section 5 presents the results of the evaluation process (Task 4). Section 6 and 7 describe the work on further development of the techniques selected for improvements. Also included is an annotated bibliography of related resource material. A summary of the VHSIC phase 1 integrated circuits and their BIT implementations are included in appendix A.

		Report
Task #	**Task Title**	**Section**
1	Military Electronics Design Investigation	2
2	TSV/TSCA Techniques Survey	3
3	TSV/TSCA Improvements Investigation	4
4	TSV/TSCA Evaluation and Selection	5
5	TSV/TSCA Technical Details and Specifications Derivation	6, 7

Table 1-1. Report Organizaton

2.0 MILITARY ELECTRONIC DESIGN TRENDS

The objective of this task was to investigate current and projected near-future military electronic design technology and trends which can affect the character and characteristics of BIT systems and, hence, test system verification and test system condition assessment. In establishing the time period of interest for the trend assessment, it was desirable to concentrate on the few years when the results of this study would be expected to be implemented. Since the final report is planned for publication in mid-1986 and full implementation would take 1 to 2 years, the BIT verification techniques developed as a result of this study would be implemented in 1988 to 1990. Therefore, the time period used for investigation of trends typically extended to just beyond 1990.

This task was structured to examine trends in three categories as illustrated in figure 2-1. The first category, design technologies, consists of a variety of design thrusts, programs and technology areas that may have a significant impact on future electronics designs. The second category covers basic component technologies. The third is the military's Very High Speed Integrated Circuit (VHSIC) program which is not only changing the state-of-the-art of component technology, but also utilizes elements of some of the design technologies.

2.1 DESIGN TECHNOLOGY TRENDS

There are a number of important design technologies that will impact military electronic designs in the near

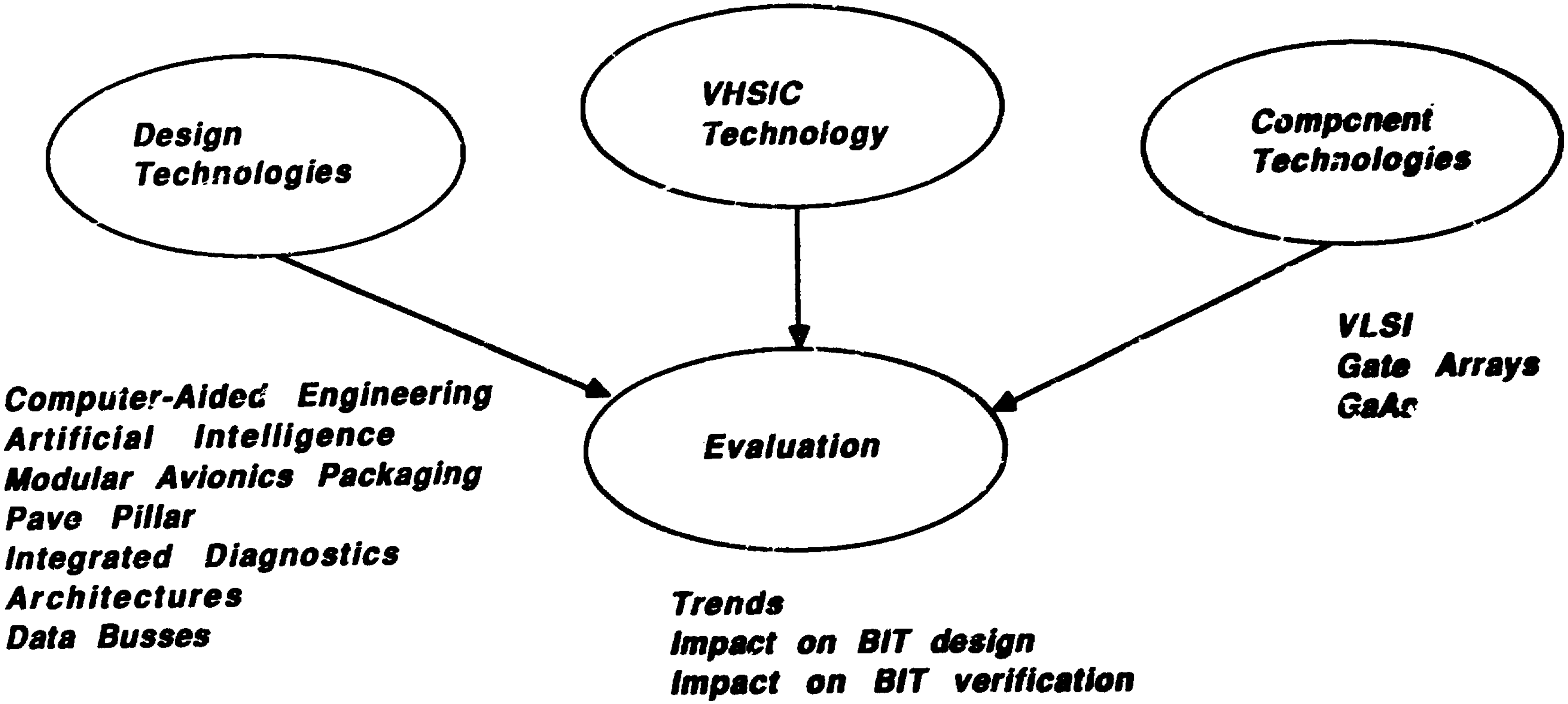


Figure 2-1 Military Electronics Design Trends

future. Those identified for this study are:

a. Computer-Aided Engineering
b. Artificial Intelligence
c. Modular Avionics Packaging
d. Pave Pillar
e. Integrated Diagnostics
f. Architectures
g. Data Buses

Each of these will be examined and evaluated as to its impact on built-in test design in the following sections. Conclusions for the various design technologies may be contradictory since they are drawn only from evaluation of the individual technology, but these will be resolved in a discussion of the overall impact of these design trends in section 2.4.

2.1.1 Computer-Aided Engineering

The use of computers in the electronics design process has increased beyond the Computer-Aided Drafting capability. Computer-Aided Engineering (CAE) provides the designer with a computer design assistant to do bookkeeping type tasks, verify that designs conform to design rules, check for errors and manipulate data, artwork or machine control information. There are also computer tools that simulate designs to verify operation before implementation.

The use of computer-aided engineering is increasing significantly, especially for the design of integrated circuits. CAE is currently being used for Very Large Scale Integrated (VLSI) circuit design, including VHSIC and gate array design as well as the traditional roles of printed wiring board design and assembly wiring generation.

The use of computer-aided engineering tools to perform design for testability tasks is lagging, but is currently being addressed. Some integrated circuit design systems provide the capability for automatic incorporation of test circuits. This is usually in the form of a set/scan technique such as the system reported in reference (1). For integrated circuit design, fault simulators exist that can be used to verify the built-in test or generate the set of test vectors used by automatic test equipment. These simulators can also be used for subsystem design when gate-level models of its components are available.

The future should provide for better computer-aided engineering through incorporation of testability tools, particularly at the subsystem and system levels. References (2) and (3) recommend development of Computer-Aided Design for Testability tools. Following those recommendations would provide for consideration of testability in the early stages of design, resulting in the development of more testable circuits, subsystems and systems. One of the difficulties to overcome in adopting these recommendations is to effectively model VLSI devices used in the design of these subsystems and systems.

To do this, the capability tc accurately model systems at a level higher than at the gate level must be developed. One strong reason for this is that gate level descriptions of most VLSI devices are proprietary and not available to the subsystem designer. A second reason is that even when gate level models are available, computers would not be able to handle simulation models of that complexity. For example, if a subsystem contained 100 VLSI devices of complexity ranging from 10,000 gates to 100,000 gates, the simulation of the subsystem would have to be capable of handling 1 million to 10 million gates. Even if the computer on which the simulation ran could handle a model that large, the execution time would be prohibitive. Modeling devices and subsystems at a higher level would help overcome these problems.

As a result of the predicted increase in the use of computer-aided engineering systems that incorporate testability tools, the built-in test capability of future systems will be improved. This improvement will be in the form of better fault coverage and automatic incorporation of standardized BIT (e.g. set/scan registers).

2.1.2 Artificial Intelligence

Developments in Artificial Intelligence (AI) are rapidly finding their way into practical applications in industry, and efforts are currently underway to develop military applications. The field of artificial intelligence generally includes natural language processing, robotics, machine vision, expert systems and other related fields.

Expert systems have already been used effectively in industry for diagnosing electronic systems, examples of which are cited in reference (3). Reference (3) evaluated the possible applications of artificial intelligence to testability and found several to be particularly cost effective. As a result, a practical evolutionary development program based on that work was recommended.

The recommended program is based on government support for the development of basic tools and application independent rule bases for two primary efforts. These would be expert systems hosted on engineering workstations, which are becoming standard industry tools for a broad range of engineering tasks. The first effort would be a computer-aided design for testability system. This would give design engineers access to the testability engineer's expertise early in the design cycle at all levels of circuit, subsystem and system design. The second would be a maintenance expert design system which would permit easy development of diagnostic expert systems either for organizational level maintenance or for use on automatic test equipment. It would contain a set of metarules (application independent rules) representing general diagnostic strategy upon which the user would develop application specific rules. The capability of developing self-improving diagnostics is seen as a later evolutionary step. Both of these applications of artificial intelligence are expected to greatly improve the testability of future military electronic systems through supporting improved design approaches. In particular, they will make the built-in test more effective and the design less prone to containing hidden design errors.

There is, however, one area of concern related to application of artificial intelligence in military systems. That is, the validation of expert systems embedded in the electronics, as an integral part of the built-in test such as that under study in the Smart BIT effort (4). Smart BIT would use an expert system rule base to examine test data and filter out false alarms. Some work has been accomplished in verifying the design of expert systems (5) but no work addressing monitoring of expert systems during field operation was uncovered. BIT has not been developed for expert systems, but BIT for the hardware hosting the expert system can be implemented independently from the expert system application.

For expert systems incorporated as part of BIT, the design can be verified using the same techniques developed for other expert systems (5). Monitoring the operation of an embedded expert system for BIT will involve testing both the hardware and the software. The hardware may be conventional von Neumann computers, LISt Processing (LISP) language machines or, ultimately, parallel inference processors. Testing of conventional computers will be accomplished the same as it is currently. It is anticipated that special inference processors will use BIT techniques similar to the best of what is in use now for production systems or those for near-term new VLSI devices (e.g. VHSIC). In either case, the testing of the hardware is not influenced by the fact that the software implements an expert system. The software can incorporate tests for gross operation but detailed testing of software operation will not be feasible. Software errors will need to be corrected as part of design verification.

The risk associated with verification of a BIT expert system and the monitoring of its operation during use will be a key consideration in the development of embedded expert systems for BIT. It is unlikely that expert systems will be embedded as part of a subsystem's BIT before 1990. The initial impact of artificial intelligence on BIT design will be its use in computer-aided engineering systems as described in section 2.1.1.

2.1.3 Modular Avionics Packaging

The Navy's Modular Avionics Packaging (MAP) effort was initiated in the mid 1970's to standardize avionics packaging. The motivation was to reduce life cycle cost of weapon systems by using standard modules on a variety of programs. The original thrust of the program was to specify standard modules (circuit cards) for Air Transportation Rack (ATR) style boxes. With the rapid increase in electronics density, the emphasis shifted to an integrated rack concept. The rack would contain collections of standard modules (circuit cards), without individual boxes enclosing functions. This could reduce weight and volume requirements by 30 to 50 percent (6). The modules would be interconnected to implement functions and the functions would be interconnected through the rack wiring. Power and cooling would be provided as part of the rack design.

The integrated rack concept continues to run into technical problems with thermal management, electrical interference between the interconnections, and exposure of the rack and the modules to the environment during maintenance. However it offers enough benefits that development of the concept continues. For example, in the Pave Pillar program (see section 2.1.4), the concept is being evaluated for application on tactical fighters. As part of this program, ARINC Research Corporation is preparing a military standard for standard size line replaceable avionics modules. These would take advantage of surface mount technology and VHSIC components to achieve high density.

The MAP concept could potentially make BIT more complicated or require more BIT since failures would need to be isolated to a module rather than a box. However, as circuit density increases, more complete functions will be implemented on a single module, reducing the need for additional isolation capability. The effectiveness of BIT will be improved with implementation of the ARINC military standard (to be approved around 1988) since it will incorporate design for testability guidelines and BIT design requirements. This standard will address testing at all levels, and include preferred methods for BIT, use of a standard test bus, standard system interfaces, autonomous module checking and module storage of BIT maintenance data.

2.1.4 Pave Pillar

Pave Pillar is an effort sponsored by the Avionics Laboratory of Air Force Wright Aeronautical Laboratory (AFWAL) to define and demonstrate the avionics system for the 1990's tactical fighter. Program emphasis is two-fold. Increased performance, to counter the ever-increasing threat, is to be achieved through greater integration of aircraft systems with defensive and offensive avionics systems. Increased availability is to be achieved through incorporation of fault tolerance and reduction of maintenance requirements. Testability issues have been considered throughout the concept definition phase. The following are summaries of features of the Boeing Pave Pillar concept that may impact future implementations of built-in test.

Generally, the architecture consists of several groups of processing elements interconnected by multiple, redundant, high speed, serial data buses. Nondigital information is converted to digital form at the front-end of the system and digital processing is used to the maximum extent possible. Functions are distributed among the various computing elements which are replicated where necessary to meet fault tolerance requirements. The groups (mission management, vehicle management, etc.) are loosely coupled via serial data buses or control elements. The serial data buses will use a new standard high speed bus (>20MHz), and the interconnections will be fiber optic links. The computing elements will make maximum use of VHSIC devices. Common modules will be used whenever possible, using software to tailor their functions. Certain hardware elements will require unique modules. For example, the cockpit displays will be unique because of special form and function requirements. The total system is expected to consist of approximately 250 modules of about 40 different types.

Detection of failures within data processing computing elements will make use of BIT incorporated in the VHSIC devices, but primary testing will be accomplished by duplication of functions and synchronized comparison of the outputs. In the signal processing elements, a data flow architecture will probably be implemented and testing will be accomplished by injecting test data vectors in the data stream and testing for errors in the data as it is processed. When a failure is detected, the module will either report its failure or shut down, depending on the type of failure. The failed module's task will then be executed by a spare module. BIT hardware, other than that incorporated in the VHSIC devices will be kept at a minimum and system level test functions will be implemented in software.

Each computing element will be packaged on a single card module. A data processing module will include duplicate elements for processing (CPU), memory, unique Input/Output (I/O), and serial bus I/O. Elements are duplicated to provide for fault detection. Each operation is performed by each member of the duplicate hardware set and the results are compared. Disagreement indicates the presence of a fault. This isolates the fault to the module level because all functions are on the same card.

An additional feature is that there are only a few pins (for power, serial bus I/O and unique I/O signals) on each card module. This significantly reduces failures due to interconnections and makes isolation to the module more straightforward. The modules will be installed in an integrated rack (an adaption of the Modular Avionics Packaging concept) that contains sufficient power and cooling for up to 40 modules. The rack also provides for interconnection between modules within the rack and fiber optic interfaces to other racks and devices.

Testing of the avionics was considered early in the concept development. This resulted in a system concept that reduced typical testing problems. The problem of interconnection failure is reduced by putting entire functions on single cards and using serial communications buses. The problem of module isolation is addressed by putting whole functions on cards and duplicating the ciruitry to provide fault detection. System failure rates are reduced by use of serial buses, and interconnection faults can be detected and isolated through the bus protocol. Built-in test design is simplified due to use of duplication within the module for the data processing modules, and injection of test vectors in the signal processing modules.

The trend in BIT design, as indicated by the Pave Pillar approach, is toward simple duplication of the circuitry. This is made possible by advances in circuit technology. It not only provides for effective fault detection, but also provides effective test system condition assessment since a failure in either half of the duplicated circuit is also a failure of the BIT and is detected.

2.1.5 Integrated Diagnostics

The integrated diagnostics effort was initiated by the Department of Defense in the early 1980's in response to diagnostic problems encountered in maintaining modern electronic equipment. The problem is that specifications only called for automatic diagnosis of a majority of system failures. This is accomplished using BIT or Automatic Test Equipment (ATE). The remaining failures are generally ignored. Therefore, when a particular problem exceeded the automatic diagnostic capability, the maintenance technician was left with only a schematic and an illustrated parts breakdown, but no diagnostic guides. This is an especially critical situation, because problems that exceed the BIT or ATE capability usually tend to be the more complicated ones.

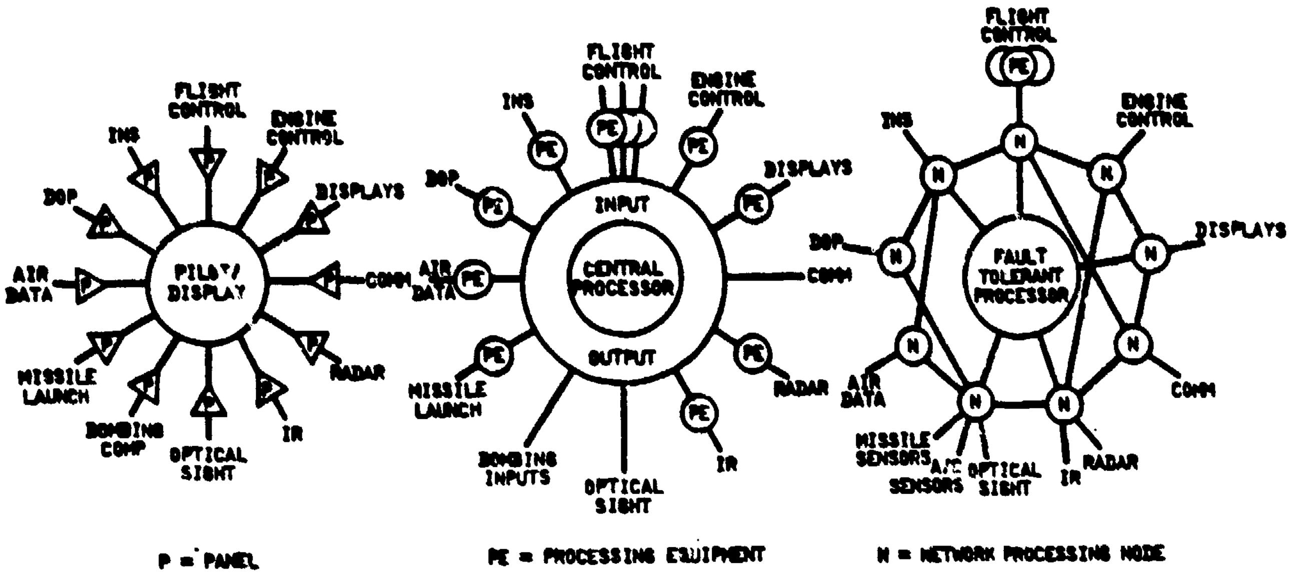
The objective of the integrated diagnostics effort is to develop standards and technology to provide an integrated mix of BIT, ATE, maintenance aids and manpower management to provide unambiguous diagnostics for all failures expected to occur. The National Security Industrial Association (NSIA) has formed an integrated diagnostics working group to provide guidance to the Department of Defense. The working group is to recommend technology development, demonstration projects, and policies and standards for implementation of integrated diagnostic concepts.

The impact of integrated diagnostics on BIT design is uncertain at this time. More sophisticated BIT with better diagnostic capability may be required. On the other hand, it may turn out that a better interface between the maintenance technician and BIT is more important than better built-in diagnostic capability. The improved interface could allow the technician to access test data for analysis, as needed, rather than provide him with an end conclusion from preprocessed data. This interface needs to be able to provide more information and better ways of requesting and displaying data than currently available. Use of artificial intelligence (e.g., expert systems) is a potential solution to this problem.

2.1.6 Architectures

Architectures of military electronics have a significant impact on the design of electronic subsystems and hence the built-in test. As illustrated in figure 2-2, the architecture of avionics systems is changing with the trend toward distributed systems and digital implementations. In early systems, all information was displayed to the pilot who functioned as the system integrator. Most modern systems utilize a central digital computer to integrate and distribute information. The pilot utilizes integrated data such as navigation data integrated from a number of sources. Future systems, many now in concept development, will consist of highly distributed hardware and software. The hardware will be distributed based on the sources of data and locations and requirements for controls. The software also will be distributed through the hardware elements. This allows for even greater integration of functions thereby increasing pilot effectiveness. It was estimated that in 1978, the military avionics inventory was 90% analog and 10% digital (7). Systems currently being implemented are more than half digital, with the specific amount depending on the type of system. It is predicted that by 1998 avionics implementations will be 90% digital and 10% analog (7).

BIT will also have to be distributed with the functions, but system level BIT control is still necessary and will have to use system resources, i.e. buses, processors and mass memory. More systems will incorporate fault tolerance, and the BIT design will have to be compatible with fault tolerance concepts. This means that more of the BIT will need to be concurrent. It will have to operate continually during operation of the subsystem or be interleaved with other operations to detect failures in time for the fault tolerant hardware and software to take



1940 - 60

ANALOG

- WIRED PROGRAMS
- DEDICATED ANALOG PROCESSORS
- INTEGRATION THROUGH PILOT/DISPLAYS
- NO REDUNDANCY
- LITTLE FAULT TOLERANCE
- NO DYNAMIC RECONFIGURATION CAPABILITY
- DISCRETE AND SSI HARDWARE

1960 - 80

CENTRAL DIGITAL

- STORED PROGRAM
- CENTRAL PROCESSOR(S)
- COMMUNICATION THRU I/O INTEGRATION THROUGH CENTRAL PROCESSOR/STORED PROGRAM
- SOME REDUNDANCY
- SOME FAULT TOLERANCE
- NO DYNAMIC RECONFIGURATION CAPABILITY
- MSI AND LSI HARDWARE

1980 - 2000

DISTRIBUTED DIGITAL

- DISTRIBUTED HIERARCHICAL STORED PROGRAM
- REDUNDANT CENTRAL PROCESSOR(S)
- DISTRIBUTED DEDICATED FUNCTIONAL PROCESSORS
- COMMUNICATION THRU BUS NETWORK
- LARGE SCALE USE OF REDUNDANCY
- FAULT TOLERANCE AND DYNAMIC RECONFIGURATION
- VHSIC HARDWARE

Figure 2-2 Avionics System Architectures (7)

corrective action. BIT will become more critical and will receive systems level attention early in the design process. Existing BIT techniques are adequate to meet requirements of distributed systems.

2.1.7 Data Buses

Future systems, because of their distributed nature, will continue to make use of the MIL-STD-1553B data bus or its derivatives. The trend will be toward higher speeds and the use of fiber optic transmission media. The bus control and terminal interfaces, except for transformers and fiber optic transceivers, will be reduced to single integrated circuits. The effect of serial data buses and single integrated circuit interfaces on BIT is that isolation to the failed Line Replaceable Unit (LRU) becomes simpler and faults (especially intermittent faults) due to interconnections are significantly reduced. There is, however, a need for reliable BIT as part of the bus interface components in order to properly diagnose bus related failures. No new BIT techniques are required, however the need to incorporate BIT in bus interface components must be addressed during system design.

2.2 COMPONENT TECHNOLOGY TRENDS

Trends in electronic component technology were examined to determine impact on built-in test (BIT) design through 1990. This was accomplished by reviewing electronic component technology trend information from in-house sources and from recent electronics design trade publications. The discussion of results addresses the forces driving electronic component development, component parameters of interest, and anticipated trends. It includes an assessment of the impact of these trends on BIT design.

2.2.1 Driving Forces

A significant factor in the component technology trend assessment is the trend toward digital implementations of electronic systems. The continuing shift from predominantly analog designs to predominantly digital designs is the result of the trend in architectures as discussed in section 2.1.6 and advances in large scale integration of digital circuits. These advances are resulting in dramatic increases in capability, as well as decreases in cost and power requirements of integrated circuits. This results in increased capability for implementation of operational requirements and for inclusion of more sophisticated BIT capability.

The majority of digital computing applications over the past ten years have used general purpose von Neumann data processors, particularly microprocessors. This is continuing, but interest is increasing in non-von Neumann processors for a variety of signal processing functions. It is in this area that the majority of the displacement of previously analog implementations will occur. Table 2-1 illustrates the variety of digital functions in military electronic systems (8).

In addition to the expansion of applications for digital electronics, processing requirements are increasing significantly. Table 2-2 shows the expected growth in throughput requirements for digital processors in several applications by 1990 (8).

Because of the trend toward the use of digital electronics and the significant increases in capability required for

Table 2-1 Digital Functions in Military Electronic Systems (8)

Military Systems / Digital Functions Used	Communications	Navigation	Command & Control	Passive Radar	Active Radar	Infrared	Video Scan	ECM/ECCM
Analog to Digital Conv.	x	x		x	x	x	x	x
IF Band Manipulation	x			x	x		x	x
Data Reordering				x	x		x	x
Correlation	x			x	x	x	x	x
Filtering	x			x	x	x	x	x
Adaptive Filtering	x	x			x			x
Detection and Integration	x	x	x	x	x	x	x	
Display Formatting		x	x	x	x	x	x	x
Data File Maintenance			x		x		x	x
Logic and Calculation	x	x	x	x	x	x	x	x

Table 2-2 Expected Growth in Digital Throughput Requirements for Various Defense Systems (8)

Digital Processor Function	Computation Rate Required (MIPS)	
	Now	1990
System Management and Control	0.1-1.0	10-100
Radar	1-10	100-500
EO and IR Imaging Systems	10-20	200-2,000
Broadband Secure Communications	10-30	500-2,000
Electronic Warfare	25-100	1,000-10,000

the future, the component technology assessment is concentrated on digital components. Trends for analog components are following those of digital components (i.e. greater performance and larger scale integration). However, no resulting changes to BIT techniques are required based on current indications.

2.2.2 Component Parameters

The trends in electronic component technology were analyzed in terms of key component parameters. This was because most trending information is presented in terms of these parameters and most of the parameters relate directly to the primary performance of the devices. The following discussion of each of the key parameters includes the significance of the parameter, the trends in general and the relationship to other parameters.

Chip size - relates to the functional capability of the device. That is, for a given feature size, the larger the chip, the more there is on it and the more it can do. Chip size has been increasing through successive levels of integration (Small Scale Integration (SSI), Medium Scale Integration (MSI), Large Scale Integration (LSI), Very Large Scale Integration (VLSI)) and will continue to increase for the foreseeable future as the industry moves towards wafer scale integration.

Feature size - relates to the functional capability of the device, in that the smaller the features are, the more that can be put on a chip. Feature size also has been decreasing and will continue to decrease, well through the 1990 time period. Ultimately, feature size is limited by the physics of the materials, but that limitation will not be reached until well after 1990.

Active elements per device - relates to the functional capability of the device and reflects the combined effects of the first two parameters. As chip size increases the number of active elements per device increases. For a given size, smaller feature size allows more active elements per device. This parameter is most often used to project trends in integrated circuit technology.

Performance - is a direct measure of device capability that depends on the type of device. The most common are propagation delay for logic gates, access time for memories and operations or instructions per second (usually in thousands or millions) for processors. Performance over a class of devices depends primarily on the specific technology, that is the materials and manufacturing processes used (e.g. bipolar is faster than Complimentary Metal-Oxide Semiconductor [CMOS]). For a given technology, however, performance improvements can result from changes in the manufacturing processes, especially those which permit reductions in feature size. The trends then are toward generally increasing performance with significant increases resulting from new technologies.

Power - is of concern for applications with a limited power source or where heat dissipation is difficult. As with performance, power depends primarily on the specific device technology. In general there is a trade-off between power and performance in selecting the device technology. Devices with greater performance require more power. Within a given technology, the more active elements there are in a device, the more power it requires.

I/O pins availability - continues to be a limiting factor in VLSI designs. Increases in functional capability and device flexibility require more I/O pins. Also, it is desirable to add extra pins for test accessibility. Until recently, most integrated circuits were limited to 40 pins, with a few going to 48 or 64. Now packages are being introduced with more pins, especially leadless chip carriers (up to 164 pins) and pin grid array packages (up to 240 pins) (9).

2.2.3 Trends

The previous sections presented a general indication of the expected trends for each of the major parameters. Figure 2-3 shows specific trends predicted for the number of active elements per device and the factors contributing to it. It shows the increase in die size and the decrease in feature size through 1990. These result in the number of active elements per device shown by the curves. The curves are for Random Access Memory (RAM) devices and Microprocessor Units (MPU), the two most common and most important digital integrated circuit types. Memory devices have the greater number of active elements for a given chip size because they have regular layout patterns and relatively simple interconnections.

The curves are generated from information from reference (8), and the data for wafer size, die size and feature size are from reference (10). Projections from other sources show slightly later availability for production devices of a given capability, so figure 2-3 should be considered as representing the earliest availability of a device (i.e. preproduction). Differences in projections between sources reflect both levels of optimism and differences in timing between working laboratory versions, initial samples, commercial production and military production of integrated circuits. The specific availability of a device is not as important as the

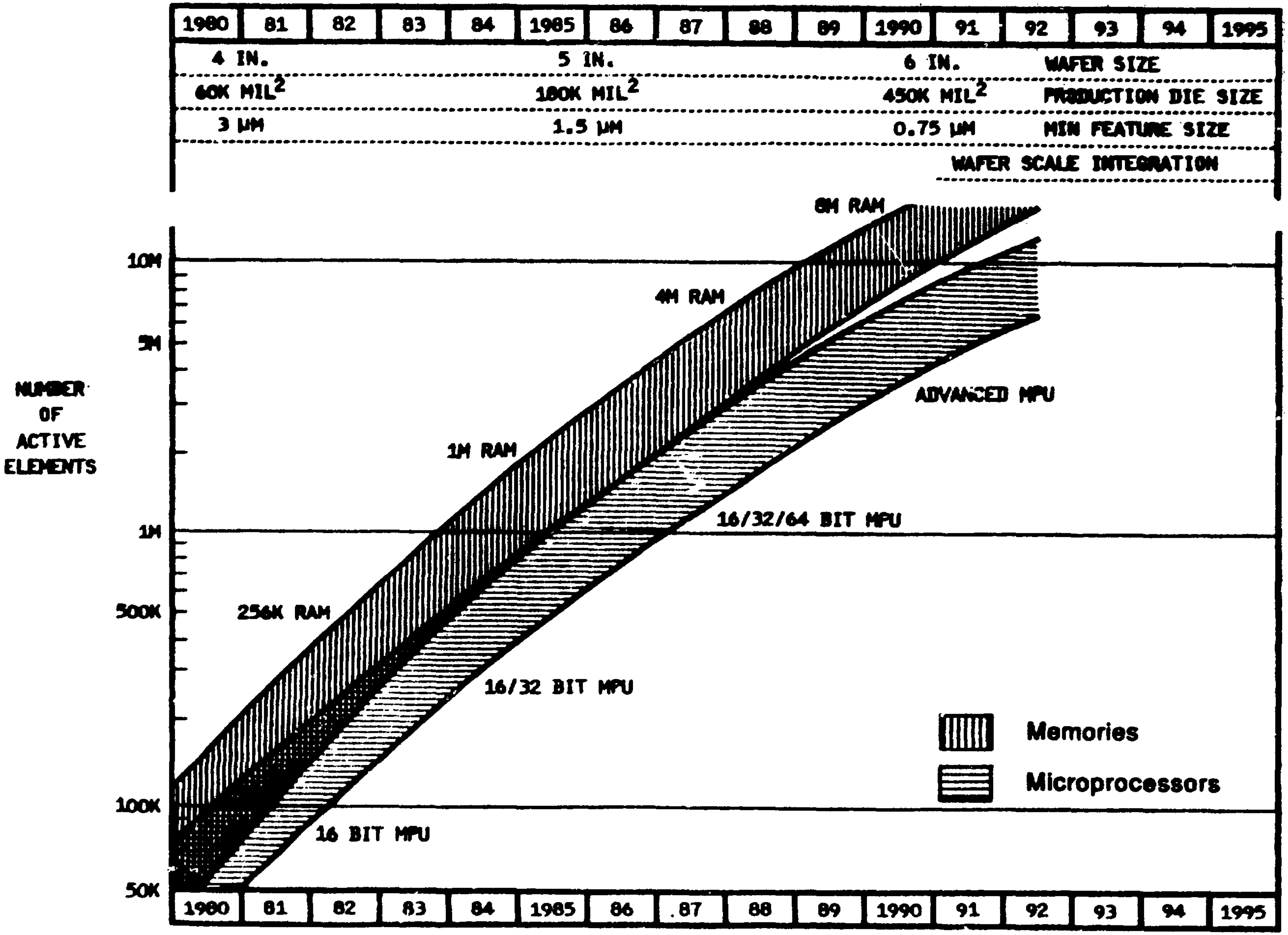


Figure 2-3 Device Trends (8), (10)

indication that the growth in device complexity will continue doubling approximately every 2 years through 1990.

Similar trends can be found in the performance parameters. Figure 2-4 shows projected semiconductor memory access time for N-channel Metal-Oxide Semiconductor (NMOS) and bipolar technologies (12). These indicate continuing improvement through 1990, and similar improvement can be expected for gate propagation delay. The performance parameter receiving the most attention, however, is microprocessor performance. Figure 2-5 shows the expected improvement in processing speed of 16 bit micro-processors through 1990.

As was previously indicated, the power required per gate in a device is primarily determined by technology, with a trade to be made between power required and performance as shown in figure 2-6. The exception is gallium arsenide (GaAs) which will offer increased switching speeds for power dissipation comparable to NMOS. Although production of GaAs devices is now beginning, there will not be substantial application of the technology in digital systems before 1990. VLSI devices of GaAs will not be available before then and use of GaAs will be limited to special applications with unique requirements, due to the cost of the devices.

It should also be noted that speed improvements in the silicon technologies at the same power dissipation will continue to be made through 1990 as a result of improvement in manufacturing processes, especially those that permit smaller more accurate features.

Power dissipation for integrated circuits is increasing due to the increasing number of devices on the chip. This is causing concern in packaging of the integrated circuits

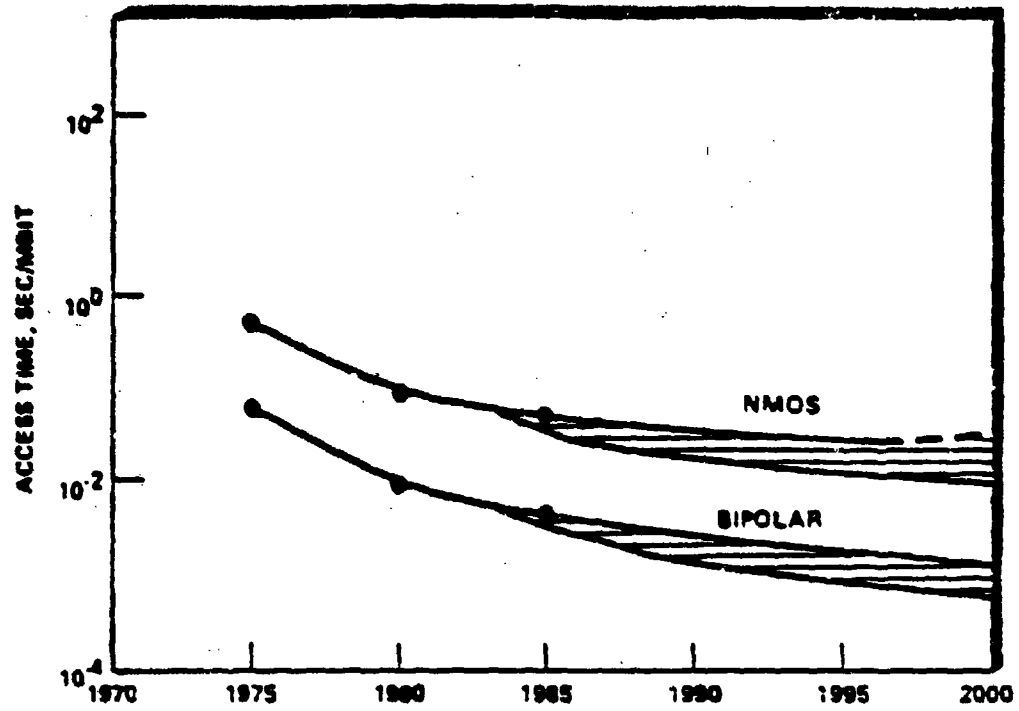
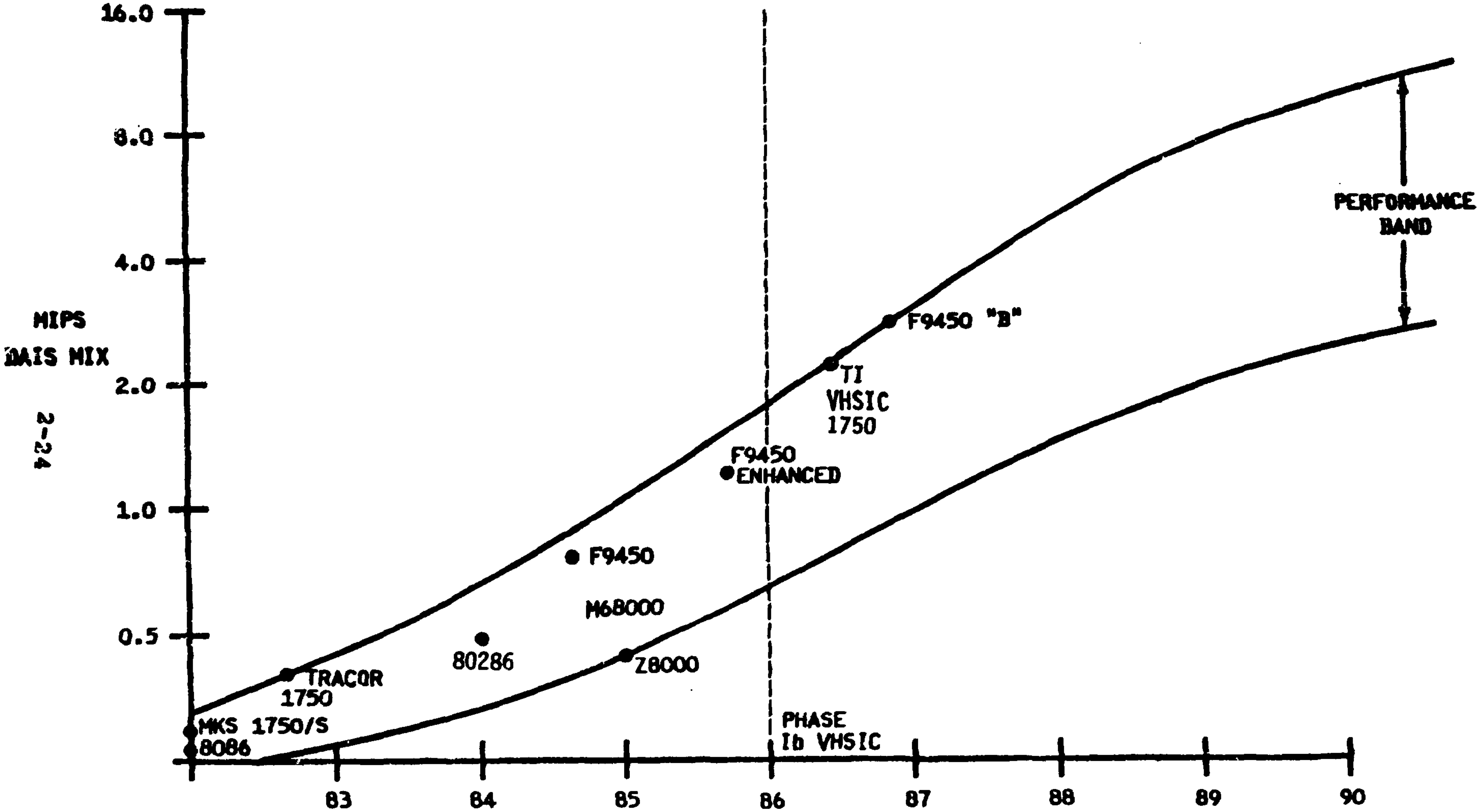


Figure 2-4 Speed of Semiconductor Memory (12)



(FPU) - WITH A FLOATING POINT UNIT

Figure 2-5 16-BIT Computer Capabilities (CHIP)

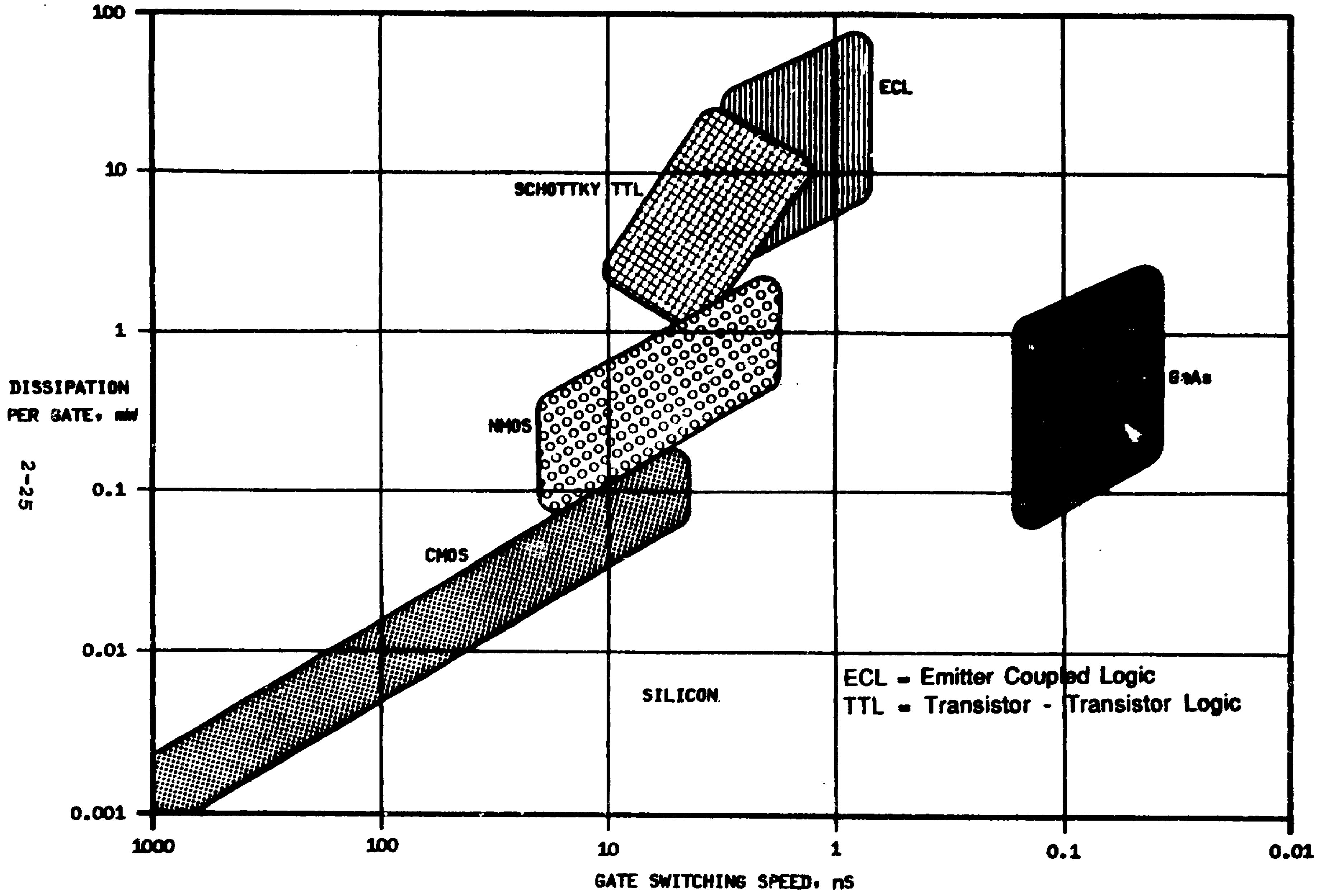


Figure 2-6 Semiconductor Speed - Power Performance

and in the design of circuit cards. For example, it often involves careful placement of the heat generating components closer to the cooling source (thermal management program).

2.2.4 Trend Summary

The above subsections summarize trends expected for the most important digital electronic component characteristics. The significance of the data for this study is not that a specific level of capability is available at a particular time, but the trend that is shown. From the information collected, the trends from the past five years will continue through 1990 with no radical changes in technology. That is, there will be increasing application of microprocessors in military electronic design, and the processing power of microprocessors and the density of memories will continue to increase, allowing processing capability per unit volume to increase correspondingly. Similar trends are expected for analog components.

2.3 VHSIC TECHNOLOGY

The Very High Speed Integrated Circuit (VHSIC) development effort was initiated by the Department of Defense to push the state-of-the-art of military grade integrated circuits. The emphasis of the program is to increase operating speeds primarily through reductions in feature size and to increase functionality by increasing the scale of integration (more devices per chip).

The VHSIC program is important to the future of military electronics design for several reasons. First, it is anticipated that the technology, if not the specific devices developed, will be used in a significant portion of new military electronics design. The program is also developing the tools to permit easier integrated circuit design which should lessen the cost impact and make custom designs more readily available. An important aspect of the VHSIC program is the mandated consideration of design for testability. The program requires part of the chip area be devoted to built-in test for the chip, but did not specify how much should be reserved or what level of performance should be achieved.

The appendix contains a summary of the various Phase I VHSIC efforts, including a description of the built-in test approach chosen for each of the devices. The most common BIT technique was built-in signature analysis. Hughes, International Business Machines (IBM), Texas Instruments and Honeywell all chose it as their primary technique. TRW used set/scan registers for loading test data and reading test results. Westinghouse partitioned their chips into test cells for which all of the inputs and outputs could be controlled. In some cases the manufacturers appeared to use BIT primarily to simplify manufacturing tests. Some of the chips have limited on-line test capabilities for their parts, but all need to be taken off-line for the thorough tests. With the possible exception of Honeywell's, the chips may be tested by an external processor in the system. Some of the processor elements may be tested as part of a regular timed operation. The Hughes parts require an external processor to compare the test results with known good results. Several of the other companies have the correct result of a test built in to the VHSIC chip (probably in ROM) and perform the comparison on-chip.

The only BIT verification done on any of the programs seemed to be fault simulation to determine fault coverage. Hughes does use self-checking comparators for some of the tests on the Encoder/Decoder chip, but that is the only on-line verification technique described. Most of the chips have not yet been fault simulated. The fault coverage predictions in those cases are consistently higher than for those cases in which fault simulation has been completed.

From the higher level, the system must initiate self testing of the circuits either on a time available or scheduled basis. Contractors who designed large portions of the systems in which the chips were to be used were able to include the capability to test the system operation (e.g. signals passing from chip to chip) as well as chip operation. But for the most part, the VHSIC parts will only respond with information concerning their own health.

The VHSIC devices have incorporated built-in test, but the resources required to exercise that capability when the devices are incorporated into a system have not been developed. These will need to be defined as part of the VHSIC technology insertion programs for the VHSIC built-in test to become effective.

2.4 IMPACT ON BIT DESIGN

There is increasing emphasis on design for testability and design of built-in test for military electronics. This is because of the high cost of maintaining equipment and an increasing need for fault tolerant systems. Currently, about one third of the life cycle cost of a weapon system is expended on maintenance labor. This is likely to increase due to the increasing complexity of electronic equipment and decreasing skills of maintenance technicians unless effective test capability can be incorporated into electronic equipment. Even with more effective BIT, more skilled technicians will be required to interact with the BIT. BIT should be designed to interface with the technician in a way that will allow him to improve his diagnostic skills. Current interest in fault tolerant design is a result of the need for high system reliability to meet current and projected availability requirements. An essential element of fault tolerance is some form of accurate built-in test (including replication of functional elements and voting). For these reasons, design for testability and, in particular, built-in test are being considered early in the concept definition phases of programs. The design for testability process may be formalized with the release of the proposed MIL-STD-2165, Testability Program for Electronic Systems and Equipment. This would help establish testability as a design discipline similar to reliability and maintainability.

The design technologies all indicate an increased emphasis in design for testability and the tools to provide it in the design cycle. As such, BIT will become more effective (greater fault coverage and less susceptibility to false alarms) by becoming more sophisticated. The only indication contrary to the trend of increasing BIT sophistication is the Pave Pillar approach, which distributes functions among replicated computing elements along with multiple, redundant, high speed, serial data buses (see section 2.1.4). This is however, an application in a highly fault tolerant, highly integrated system and not representative of all military electronics.

The concern for testability has not been fully reflected in the design of integrated circuits. Newer, complex, off-the-shelf integrated circuits have incorporated built-in test primarily for testing during manufacturing not for use in application of the devices. There are some exceptions such as parity checks or coding incorporated in some memories, timing or error condition checks in some processing devices and functional tests in some peripheral devices (e.g. loop-back test for I/O devices). For the most part, built-in test is (and will continue to be) implemented at the card, box and system level. BIT may be incorporated in some semicustom devices (gate arrays) since some CAD systems have the capability to automatically incorporate BIT, usually a form of scan technique.

Comprehensive implementation of BIT at the integrated circuit level is part of the VHSIC program. However, before this capability can be used in a subsystem design, BIT processor elements for control and monitoring of results need to be developed. This would involve standardization of chip-level BIT interfaces. In the interim, because of the high level of integration of components used in designs, BIT design will be predominantly functional rather than component oriented. That is, tests will be made to see that functions (arithmetic, I/O) are performed rather than testing to verify that individual gates are operational. Currently this is often implemented as an allocation of part of the processing capability of the equipment. With increasing processing power available in microprocessors, more capability is being allocated to BIT. This will permit incorporation of smart BIT concepts, including (1) more storage of failure data, (2) evaluation of environmental and operational conditions, (3) filtering of transient faults and false alarms, and (4) isolating intermittent faults to the faulty component. This, in turn, may lead to incorporation of a separate BIT processor in equipment designs and ultimately, perhaps, to a BIT processor integrated circuit that includes these capabilities.

3.0 TSV/TSCA TECHNIQUES

This section reports the results of a survey of the technical literature related to BIT verification techniques. The survey included articles from the following journals:

a. IEEE Transactions on Computers,

b. IEEE Transactions on Reliability,

c. Journal of Digital Systems,

d. Computers & Electrical Engineering,

e. IEEE Transactions on Circuits & Systems,

The survey also covered the proceedings of the following conferences:

a. International Symposium on Fault-Tolerant Computing

b. Reliability & Maintainability Symposium

c. International Test Conference

d. Autotestcon

e. Compcom

as well as several national library data base searches. Overall, the review yielded a surprisingly small number of references to BIT verification.

In addition to current journals, personal contacts at major United States universities were made to identify any BIT verification work which was unreported or in progress. A major finding of these contacts was that no BIT verification studies were currently included in university research. However, several university efforts are underway to develop new BIT systems, and while they are not working in the area of BIT verification, researchers at these universities expressed great interest in the results of this BIT verification study. Another finding of these contacts is the conclusion that none of the new BIT systems under development will cause a radical change in the approach to BIT verification.

Personal contacts also were made with industry and military organizations to determine what BIT verification techniques are actually in-use or under development for military electronics (and some commercial electronics) that may not have been described in publications. These contacts indicate widespread use of Failure Modes and Effects Analysis (FMEA) and simulation (both modeling and physical fault insertion) for test system verification, but little attention being paid to test system condition assessment. The FMEA and simulation techniques vary from very casual to very detailed use depending on the type of program, the company, the personnel and the type of equipment being designed.

Figure 3-1 shows a break-down of the major BIT verification techniques found in the literature search. A summary description of each technique is also provided in this section.

3.1 TSV TECHNIQUES

Test System Verification (TSV) involves those BIT verification approaches for both the evaluation of BIT systems during engineering development and for qualification of BIT systems to verify compliance with the testability requirements. They are formally applied at the system and subsystem level but may be used at all levels of development. There were four major types of TSV techniques reported in the literature and confirmed by the personal contacts: figures-of-merit, FMEA, simulation, and statistical.

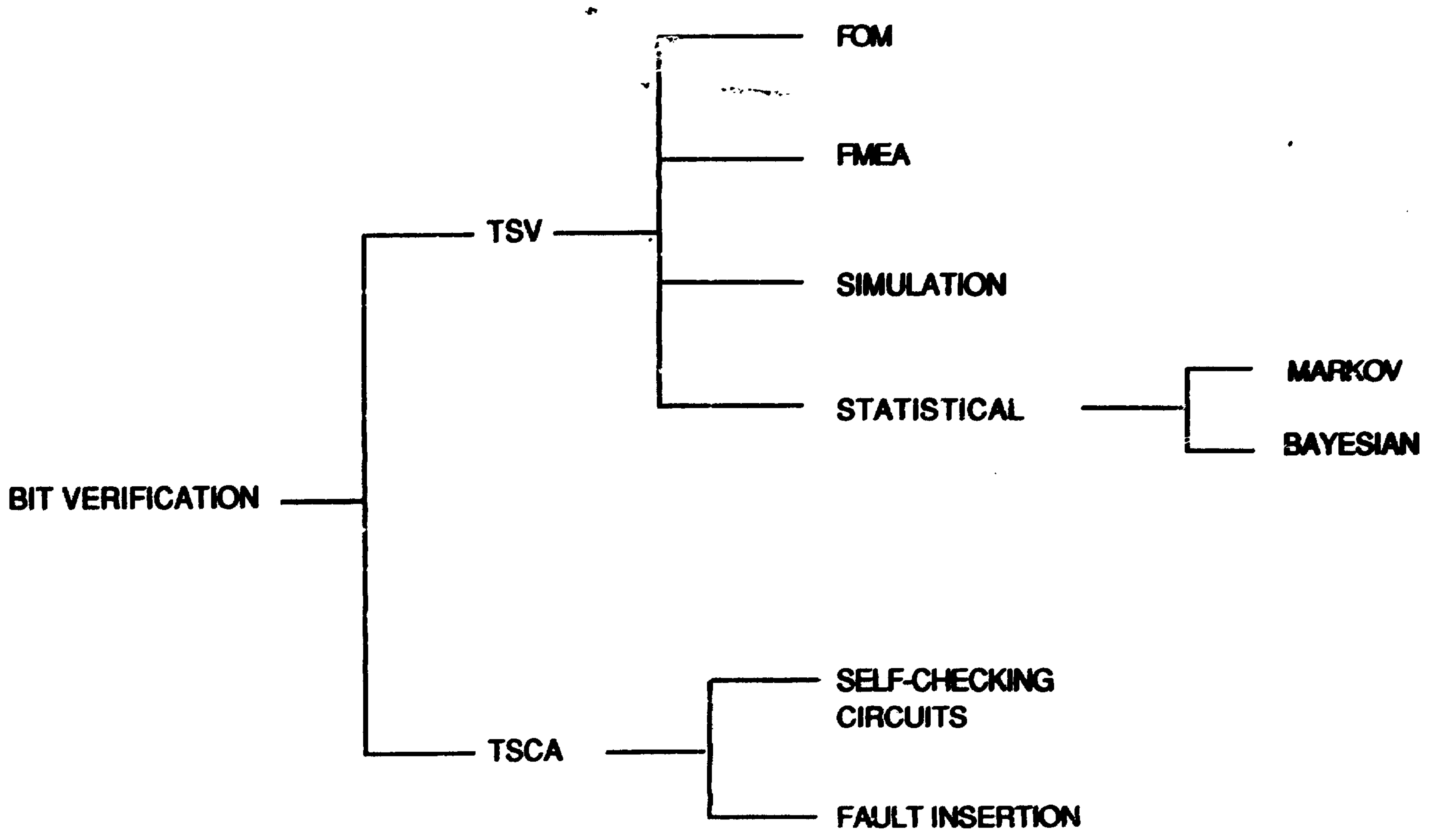


Figure 3-1 BIT Verification Methods

3.1.1 Figures of Merit

A common approach to evaluation of BIT systems involves the determination of the value of certain BIT Figures-of-Merit (FOM). This technique applies in the TSV environment. A 1979 RADC report (13) provides an excellent review of the various FOM's available. There has been little change since the report was published. The report isolated 16 different, but in some cases related, BIT FOM's:

a. Fraction of Faults Detected (FFD)
b. Fraction of False Alarms (FFA)
c. Fraction of False Status Indications (FFSI)
d. Mean Fault Detection Time
e. Frequency of BIT Executions
f. Fault Isolation Resolution
g. Test Thoroughness
h. Fraction of Faults Isolated
i. Mean Fault Isolation Time
j. Maintenance Personnel Skill Level
k. BIT Reliability
l. BIT Maintainability
m. BIT Availability
n. System Maintainability
o. System Availability
p. Mean BIT Running Time

The evaluation of these sixteen possible FOM's included in the RADC report indicated that BIT availability, BIT mean time to repair, fraction of faults detected, and fault isolation resolution scored high on the evaluation criteria of translatability, trackability, demonstratability, applicability, and uniqueness. Two of these FOM's (fraction of faults detected and fault isolation resolution) have been used extensively as TSV measures, but they suffer from the problem of requiring either a significant amount of operational data or use of complex simulations to compute the FOM's. This makes them difficult to calculate with a high degree of accuracy. The 1979 RADC report recommends verification of the FOM's by field data collection or by demonstration as part of the maintainability demonstration, except for FOM's, such as time to test, which are verified by direct measurement. It dces not address how the test results are obtained, but discusses at length the statistical analysis for selecting sample sizes and for assuring demonstrated results verify the specified FOM. These statistical analyses can be used as part of a test system verification technique.

3.1.2 Failure Modes and Effects Analysis

Failure Modes and Effects Analysis (FMEA) is widely applied to Test System Verification. Tasar and Ohlef (14) suggested the use of a statistical FMEA to determine the fault coverage of a self testing program. They assumed a single stuck-at fault model. After performing a standard FMEA using the gate level model of the entire system (the operating circuit and its BIT systems), they calculate a reliability table which shows the probability of failure of each test point. A criticality table which contains the probability of undetectable failures is then constructed. Using these two tables they calculate an overall BIT coverage value. The method they propose requires prior knowledge of the probability of all possible failures.

Kreuze (15) proposed an FMEA-derived BIT analysis scheme which will:

a. Ensure that all known major failure modes have been evaluated and are detectable and isolatable by BIT.

b. Provide a preliminary estimate of BIT fault detection and isolation capability.

c. Establish second level BIT hardware design requirements as the BIT sequence is detailed.

d. Establish guidelines for BIT software development

e. Define operational Line Replaceable Unit (LRU) interfacing requirements to facilitate BIT.

The flow chart which Kreuze uses to illustrate his FMEA derived BIT analysis is shown in figure 3-2. He offers a method to calculate the capability of the BIT system to meet design requirements using:

% faults detectable = (ffr/sfr) X 100%

% faults isolatable = (ifr/sfr) X 100%

where

ffr = sum of fault failure rates detected by BIT

ifr = sum of fault failure rates isolatable by BIT

sfr = sum of all failure rates

Kreuze illustrates the application of his technique by applying it to a typical digital Automatic Flight Control System (AFCS) servo loop.

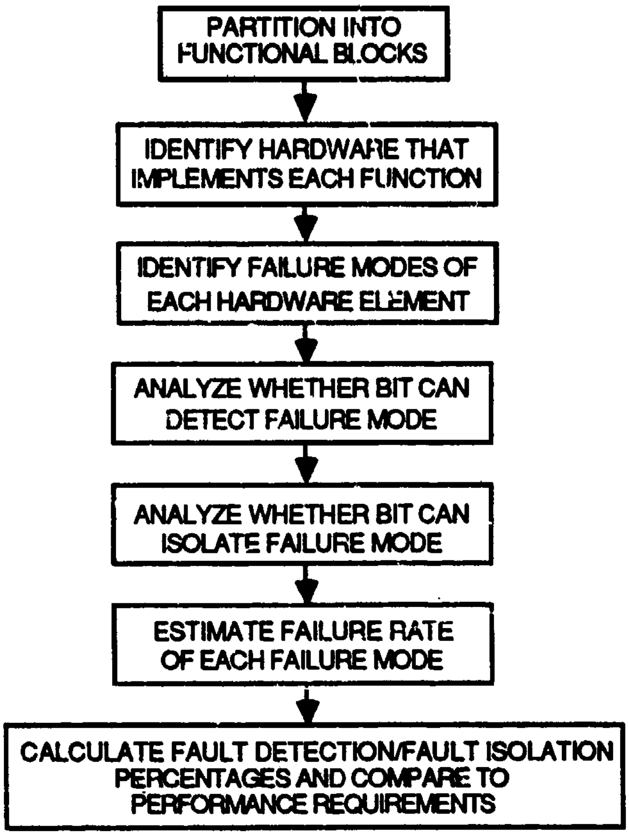

PARTITION INTO FUNCTIONAL BLOCKS
IDENTIFY HARDWARE THAT IMPLEMENTS EACH FUNCTION
IDENTIFY FAILURE MODES OF EACH HARDWARE ELEMENT
ANALYZE WHETHER BIT CAN DETECT FAILURE MODE
ANALYZE WHETHER BIT CAN ISOLATE FAILURE MODE
ESTIMATE FAILURE RATE OF EACH FAILURE MODE
CALCULATE FAULT DETECTION/FAULT ISOLATION PERCENTAGES AND COMPARE TO PERFORMANCE REQUIREMENTS

Figure 3-2 FMEA - Derived BIT Analysis (15)

Collett & Bachant (16) like Kreuze, propose a Failure Modes, Effects, and Criticality Analysis (FMECA) for BIT system verification. Using a functional level FMECA, they suggest the implementation of the flow chart shown in figure 3-3. The BIT plan is generated from the system level failure modes analysis and then the entire system including the BIT circuit is subjected to a FMECA. They applied this method to a number of designs at GTE Systems, and it produced design changes that improved the ability to meet BIT requirements.

3.1.3 Simulation

Another approach to Test System Verification (TSV) of BIT systems which, in our estimation, is used frequently yet is rarely reported in the literature in this context, is the use of simulation tools. The system operation, including BIT, is modeled and simulated under various fault conditions to determine its performance characteristics. Benowitz, Calhoun and Lee (17) estimated the effectiveness of BIT systems at Hughes Aircraft using the Hughes SATGEN (Simulation and Test Generation) program. They modeled the operational system with its BIT and simulated all single stuck-at-1 and stuck-at-0 faults. Using this method they were able to determine how many system faults the BIT system would detect. In one circuit, for example, using simulation they were able to determine that the BIT system could detect 89% of the simulated faults. Bastian, Hochwald and Suzuki (18) utilized a similar technique to evaluate BIT performance. However, they decided that rather than simulating all possible faults to determine the BIT fault coverage they would generate a sample of possible faults and simulate this smaller set. The size of the sample set of faults was determined to be 25 in order to be 95%

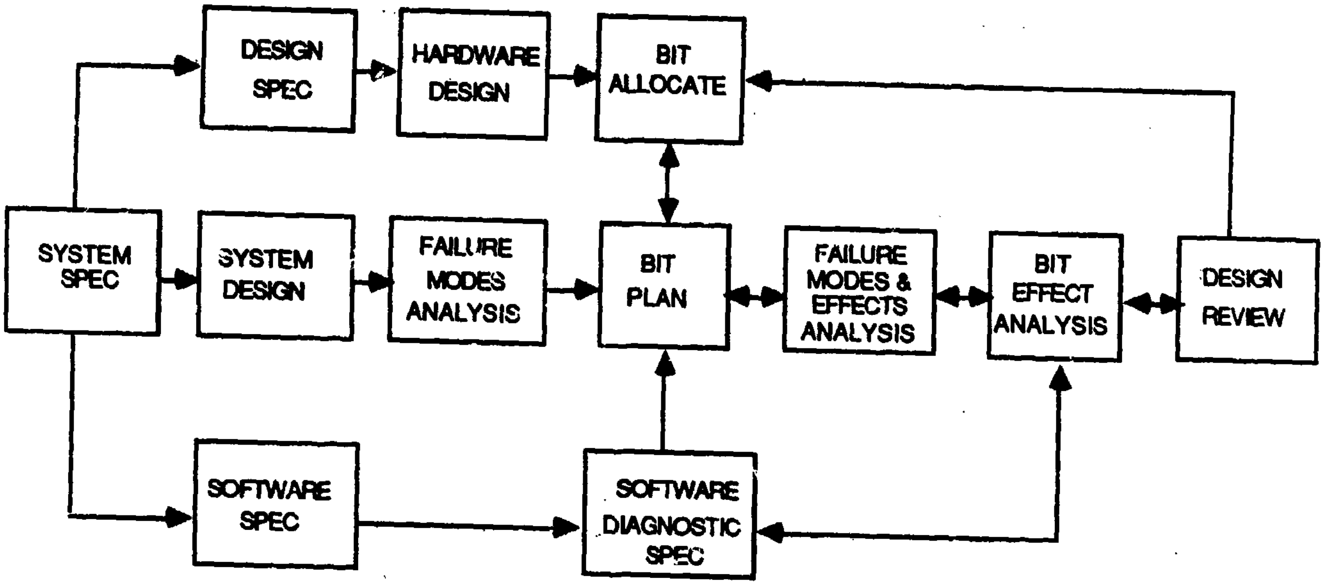


Figure 3-3 FMECA steps with BIT Evaluation (16)

confident that the BIT system could detect 90% of the system faults. This method significantly reduced the simulation time.

While fault simulation is a straightforward approach to validating the fault detection capabilities of a BIT system, it involves considerable time and expense to simulate all possible faults. At this time, it is probably the most common TSV approach. A recent improvement in this technique, as reported by Bastian et al, seems to overcome the simulation time expenses. However, the assumption made in their statistical analysis (that the number of faults detected is binomially distributed) has not been completely demonstrated as valid or fully justified.

3.1.4 Statistical

Another method of analyzing BIT performance involves the application of statistical techniques. Most approaches reported in the literature are designed to answer questions such as:

a. What is the probability of a BIT system generating a false alarm?

b. What is the probability of a BIT system missing a fault?

c. What should be the time between BIT checks?

Two statistical techniques found in the literature are described below.

3.1.4.1 Bayesian Analysis

One technique found in the literature is the application of Bayes' formula. J. G. Malcolm (19) (20) has developed a BIT evaluation approach using Bayes' formula. His application of the formula to BIT evaluation is based on the following model of the BIT testing process:

a. A system has two possible states--faulty or not faulty

b. A BIT test has two possible states--positive (indicates a failure) or a negative (no failure)

As a result there are four possible combined categories as expressed in the following test truth table:

Test Result / System Condition	Positive	Negative
Faulty	Valid Result	Missed Fault
Not Faulty	False Alarm	Valid Result

A Bayesian analysis of this truth table involves the calculation of the probabilities of an event belonging to each of the categories given certain basic information.

For example, to calculate the probability $P(F|T^+)$ that the system is faulty given a positive BIT test result (that is, BIT claims it has detected a system fault), we would use Bayes' formula as follows:

$$P(F|T^+) = \frac{P(T^+|F) \cdot P(F)}{P(T^+|F) \cdot P(F) + P(T^+|\overline{F}) \cdot P(\overline{F})}$$

where:

$P(T^+|F)$ = probability of a positive test result given a fault

$P(F)$ = probability of a fault

$P(T^+|\overline{F})$ = probability of a positive test result given no fault (false alarm)

$P(\overline{F})$ = probability of no fault

Also, the probability of a missed alarm, $P(F|T^-)$, (that is, system is faulty but the test is negative) is given by:

$$P(F|T^-) = \frac{P(T^-|F) \cdot P(F)}{P(T^-|F) \cdot P(F) + P(T^-|\overline{F}) \cdot P(\overline{F})}$$

where:

$P(T^-|F)$ = probability of a negative test given a fault

$P(T^-|\bar{F})$ = probability of a negative test given no fault

To use the above formulas, we need to know the probability of a positive BIT test result given that the system is faulty and the "a priori" probability of a system failure.

For example, assume that a BIT system is installed in a circuit and it has been determined that the circuit has a probability of failure of 1% ($P(F) = .01$). It has also been determined that the BIT system has a 95% fault coverage ($P(T^+|F) = .95$) and has a specificity (probability of false alarm) of 1% ($P(T^+ | \bar{F}) = .01$). Then the probability that the system has actually failed given a BIT alarm is:

$$P(F|T^+) = \frac{(.95)\ (.01)}{(.95)\ (.01) + (.01)\ (.99)}$$

$$= .4896$$

This illustrates a surprising result that Malcolm (19), (20) discusses at some length. That is, that a BIT system with a high degree of fault coverage may still produce a high percentage of incorrect results. A look at the effect on the BIT false alarm rate of improving the reliability of the system is shown in figure 3-4. As figure 3-4 shows, the more reliable the system, P(F) approaches 0, the lower the probability that the system is faulty when BIT indicates a system failure. Therefore, the effectiveness of BIT depends not only on coverage but also false alarm rate and the reliability of the system.

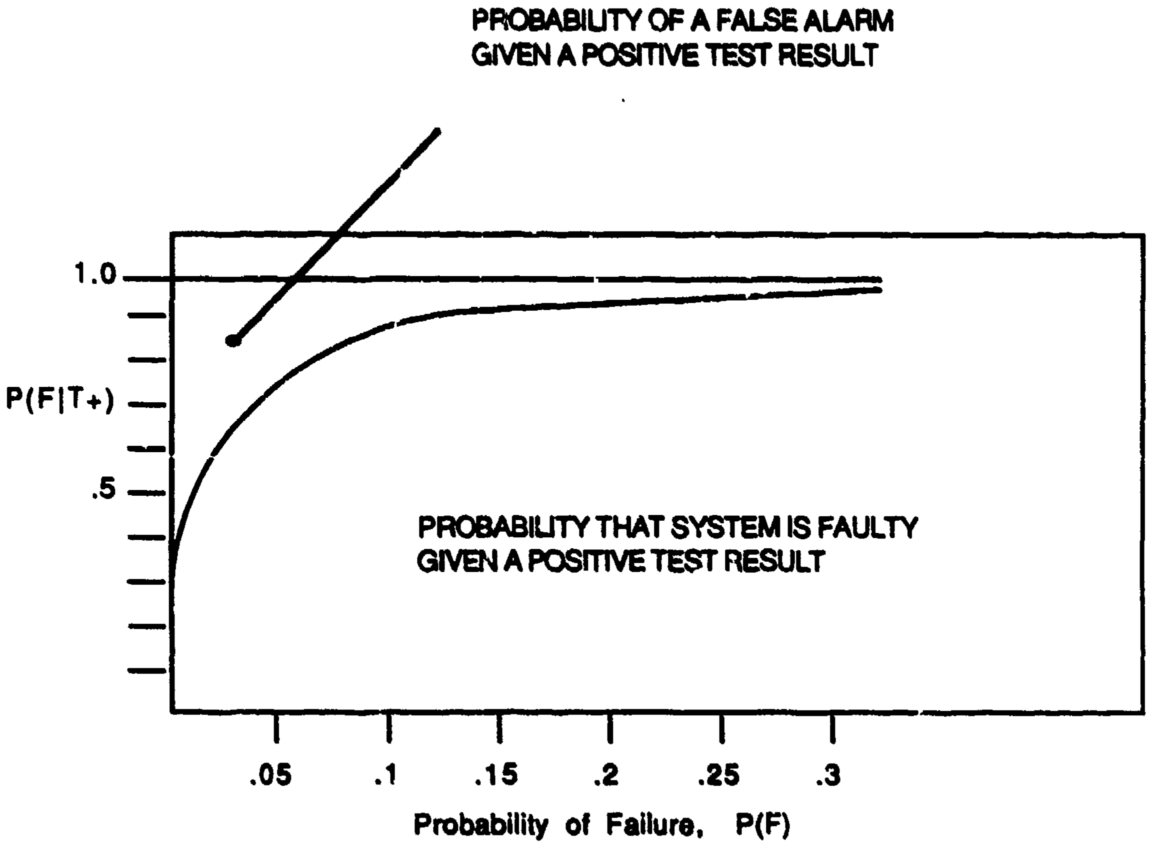


Figure 3-4 Effect of Reliability on BIT Performance

The other side of this Bayesian analysis is what is the probability of a system being faulty and the BIT failing to detect the error? Assuming the same values as in the first example the result is:

$$P(F|T^{-}) = \frac{(.05)\ (.01)}{(.05)\ (.01) + (.99)\ (.99)}$$

$$= \quad .00051$$

3.1.4.2 Markov Models

Another statistical approach to TSV is the use of a Markov model to analyze the performance of a BIT system. Markov models view the system as a series of states where a state describes everything we need to know about the system at any instant. The behavior of the system is modeled as a series of transitions between states where the Markovian assumption is that the probability of making a transition to any state in the system depends only on the presently occupied state. A good introduction to Markov models can be found in the book by Ronald Howard (21).

Capt. Gleason proposed a measure of BIT performance called BIT accuracy (22) based on a Markov model of BIT system operation. His approach could lead to a TSV measure. He assumes that the system with BIT is in one of four states:

S_0: System functional; No false alarm

S_1: System functional; False alarm

S_2: System nonfunctional; Detected failure

S_3: System nonfunctional; Undetected failure

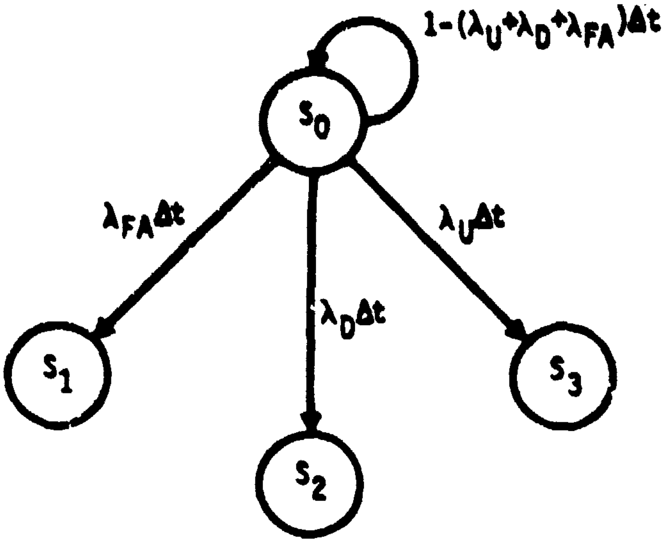
The transition between the states are represented by the probabilities:

$\lambda_{FA}\Delta t$ = probability of a false alarm in time period Δt

$\lambda_{D}\Delta t$ = probability of a detected failure in time period Δt

$\lambda_{U}\Delta t$ = probability of an undetected failure in time period Δt

The Markov model for this system is expressed in the state diagram and transition matrix shown in figure 3-5. The probabilities of the system occupying each state S_0 to S_3 as a function of time are derived from this model and are given by:



$$
\begin{array}{c|cccc|}
 & S_0 & S_1 & S_2 & S_3 \\
\hline
S_0 & 1-(\lambda_U+\lambda_D+\lambda_{FA})\Delta t & \lambda_{FA}\Delta t & \lambda_D\Delta t & \lambda_U\Delta t \\
S_1 & 0 & 1 & 0 & 0 \\
S_2 & 0 & 0 & 1 & 0 \\
S_3 & 0 & 0 & 0 & 1
\end{array}
$$

Figure 3-5 State Diagram and Transition Matrix for BIT Operation Model (22)

$$P_o(t) = e^{-(\lambda_U + \lambda_D + \lambda_{FA})t}$$

$$P_1(t) = \frac{\lambda_{FA}}{\lambda_U + \lambda_D + \lambda_{FA}} \left[1 - e^{-(\lambda_U + \lambda_D + \lambda_{FA})t}\right]$$

$$P_2(t) = \frac{\lambda_D}{\lambda_U + \lambda_D + \lambda_{FA}} \left[1 - e^{-(\lambda_U + \lambda_D + \lambda_{FA})t}\right]$$

$$P_3(t) = \frac{\lambda_U}{\lambda_U + \lambda_D + \lambda_{FA}} \left[1 - e^{-(\lambda_U + \lambda_D + \lambda_{FA})t}\right]$$

Gleason suggests that BIT accuracy is given by:

$$P_A(t) = P_o(t) + P_2(t)$$

$$= e^{-(\lambda_U + \lambda_D + \lambda_{FA})t} + \frac{\lambda_D}{\lambda_U + \lambda_D + \lambda_{FA}} \left[1 - e^{-(\lambda_U + \lambda_D + \lambda_{FA})t}\right]$$

That is, BIT is operating correctly when the system is functional and there are no BIT false alarms or when the BIT correctly identifies the failure of the system. The other states, false alarm or undetected failure, indicated that the BIT system has failed.

This Markov model assumes knowledge of λ_{FA}, λ_D and λ_U This information could be either based on experimental results or derived from some other TSV technique. Therefore, the Markov Model serves only as an analytical extension of other possible TSV techniques.

3.2 TSCA TECHNIQUES

Test System Condition Assessment (TSCA) involves determining the health of the BIT system under operating conditions and prior to maintenance. It is required at the circuit card, line replaceable unit and subsystem levels to support depot level testing and at the subsystem and system levels to support organizational level testing. In addition, the necessary TSCA capability must be provided at each level of hardware implementation to support the next higher level of hardware assembly. The focus of this study is toward TSCA implemented at the subsystem and system levels. The TSCA problem is different from TSV in that measurement of the testability parameters is no longer required. In this case, the interest is in determining if the BIT has sustained a failure affecting its performance. There were two major approaches identified in the literature search: the use of self-checking checkers and fault insertion techniques.

3.2.1 Fault Insertion

A form of BIT self-verification involves the addition of a forced failure mode into the system design. That is, the BIT system generates simulated failures in the system. Then the BIT goes through its standard check-out routine. If it detects the simulated failure then the BIT is assumed to be operating correctly. If it fails to detect the simulated failure then the BIT has failed. This technique is best suited for TSCA since it can be utilized for self-verification of equipment in the field. Often, the forced failures are simulated in software. However, Ramirez (23) has suggested a simple hardware mechanism for injecting the failure directly into the system. Such a mechanism is shown in figure 3-6. It involves a simple flip-flop which forces the multiplexer to select the stuck-at input condition. The BIT then runs its normal test sequence and if it fails to detect the forced failure it flags itself as "failed." These flip-flop/multiplexer units could be installed at several locations in the circuit.

Similar implementations of forced failures are reported elsewhere. For example. Siewiorek and Swarz (24) report forced error conditions in the CPU and forced parity errors in the memory of the DEC VAX 11/780 to test the BIT. They also report fault injection capability, implemented in both hardware and software in the Sperry Univac 1100/60.

3.2.2 Self-Checking Checkers

Another method of TSCA for BIT involves the development of Self-Checking BIT systems. Breuer (25) defines a totally self-checking circuit as a circuit which is both fault secure and self testing. This is accomplished by incorporating additional logic to encode the operation and, if necessary to decode the result. The following definition of a fault secure circuit applies to circuits for which a set of allowable outputs can be specified. An output not in that set is termed invalid. For any input, only one output from the allowable set is the correct

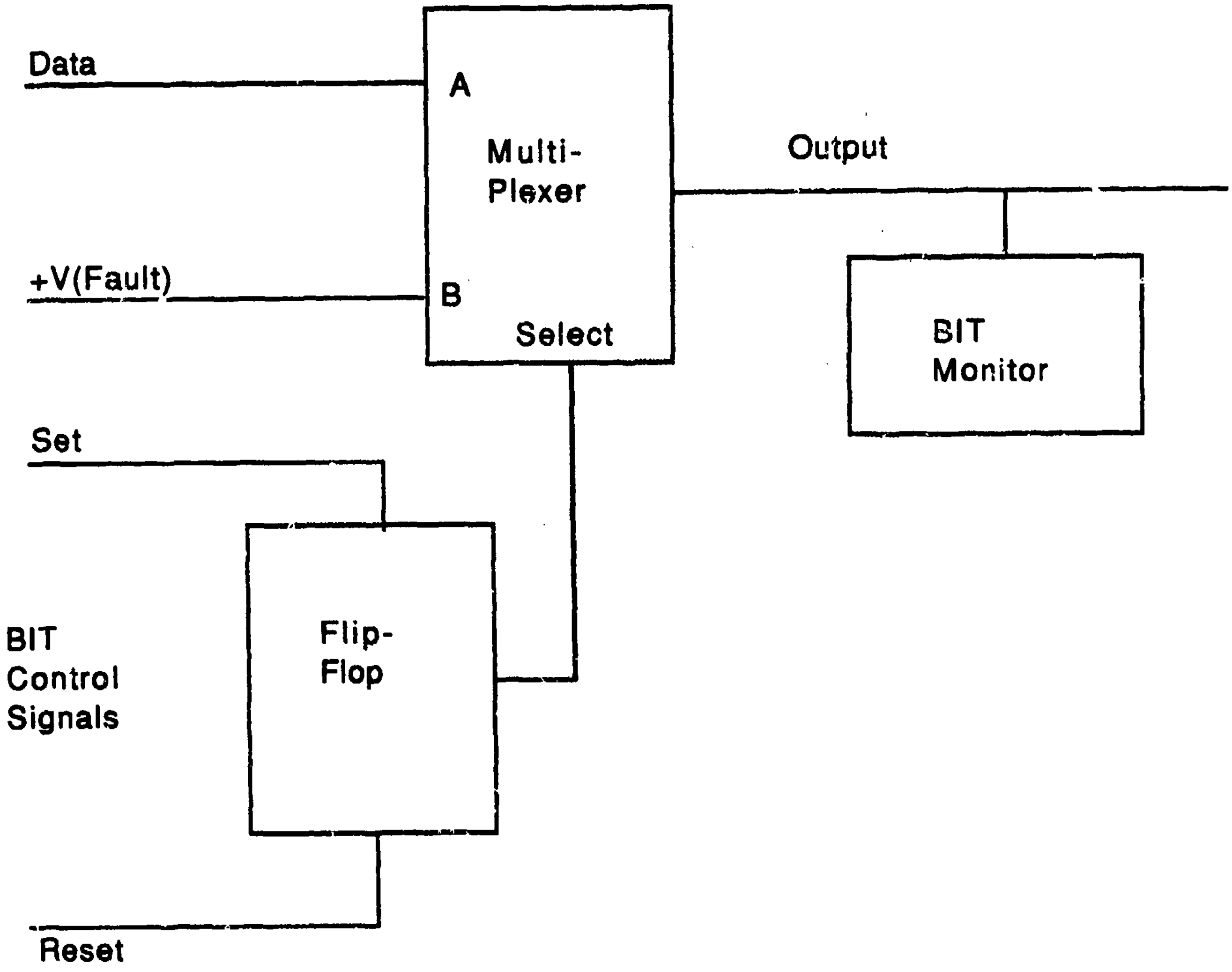


Figure 3-6 Forced Failure Mechanism (23)

output. A circuit is fault secure if, for any fault and any allowable input, the output is either (1) the correct output (if the fault does not affect the specific input) or (2) an invalid output. That is, a fault will never produce an output that is an allowable output but is, in fact, the wrong answer for the given input. A circuit is self testing if for any fault there exists an allowable code input which detects the fault. As the definition indicates, totally self-checking checkers use coded data so that an error forces a detectable, invalid word at the output. A totally self-checking BIT system which failed would produce only detectable invalid words at the output, hence, its condition could always be determined by the nature of its output.

There are a variety of codes in use in self testing circuits. The most common and well known are the parity type codes. Other codes include residue codes and group codes. A review of the literature indicated a large amount of theoretical work is currently being undertaken to develop new coding systems for self-checking checkers. In addition, there is some work reported in the application of the self-checking approach to BIT. For example, Fujiwara (26) proposes a new kind of error checking scheme for use as a combinational circuit BIT system called a group-parity prediction (GPP) checker. The system is designed to monitor a multiple output combinational logic circuit by partitioning the output into several groups and calculating the parity of each group. The calculated parities are compared to the parity predicted from the inputs. It has been shown that this method offers a high degree of fault coverage for combinational logic circuits. In addition to detecting failures in the monitored circuit, the GPP checker is also self-testing with respect to any single internal failure. Fujiwara's example system demonstrated a 98.1%

self-failure coverage and a 95.9% monitored system fault coverage for an Arithmetic Logic Unit (ALU) Controller.

Hughes, McCluskey & Lu (27) have developed a totally self-checking comparator circuit which could be useful in a BIT system that involved the monitoring of duplicated circuits. Such a BIT approach would signal an error when the duplicated circuits generated different outputs. A totally self-checking comparator would also monitor itself for internal errors and report them to the user.

Sievers and Avizienis (28) have developed a method to design totally self-checking functions in CMOS and NMOS arrays called general logic structures (GLS). A GLS is a two dimensional array much like a Programmed Logic Array (PLA) only the GLS uses a k-out-of-2k code to achieve the totally self-checking feature. They found that the GLS NMOS array was totally self-checking, but the self-checking attribute of the GLS CMOS array was sensitive to the input patterns.

3.2.3 Other Techniques

While not directly reported in the literature, personal discussions with engineers working in the area of BIT design has lead to the identification of two additional approaches to test system condition assessment. One method involves BIT systems where a single processing element is dedicated to a BIT function. Often such a processing element will have self-checking software which performs a diagnostic run when the BIT is powered up. An example application is a microprocessor implementation of BIT which acquires test data and evaluates the results to determine the presence of failures. The self-checking program then would perform a checksum test of the memory containing the BIT programs and a test of read/write memory by writing test patterns into memory then reading and verifying the test patterns. These tests verify the integrity of the BIT function (i.e. the BIT programs) and the operation of the microprocessor. Such an approach provides a check of the BIT status prior to BIT usage. This approach is limited to systems with BIT implemented as a separate processing element.

Another method for TSCA is a simple fail-safe approach. The output of the BIT system is initialized to a "failure detected" message prior to testing. The BIT system must successfully complete testing then take a positive action to change the "failure detected" message to a "no failure" message. The result is a limited fail safe capability in that in most cases of BIT failure, BIT will not be successfully completed, the message will not be reset, and the user will be notified of the failure. Of course, the user will not be able to determine from the message if the failure was in the BIT or the unit under test. However, it does reduce the problem of BIT failing in a mode that always indicates a "no failure" condition, so that when a unit under test failure occurs, BIT does not indicate the failure. An example is memory tested by a parity circuit where the result is stored in a flip-flop. At the start of a test the flip-flop is reset to zero and on successful test is set to one. If it doesn't get set to one, then either the memory is failed or the parity checking circuit is not working. This approach is applicable to any type of BIT system where the results are stored in a flip-flop, a register or a memory location.

4.0 TSV/TSCA IMPROVEMENTS INVESTIGATION

This section reports several suggested improvements to current BIT verification techniques identified in the survey. After an initial review, three of the improvements were selected for further development. The details of these developments are reported in sections 6 and 7. The specific methods covered in this section are summarized in Table 4-1.

4.1 TSV TECHNIQUES

Additional FOM's can be derived by combining existing FOM's in various ways. An illustration based on Gleason's Markov method is given below. However, major drawback to FOM's in general is that no easy way is known for obtaining accurate values for the parameters used to calculate them. Until more progress in this direction is made, it is futile to pursue FOM's any further.

FMEA and simulation techniques are also candidates for improvement. Possible directions for development work to improve these techniques are presented in the following sections.

4.1.1 New Figures of Merit

A BIT Figure-of-Merit could be used to provide a measure of predicted BIT performance. As indicated in section 3.1.1, FOMs have been developed for use as TSV measures. This research effort briefly considered the possibility of modifying several of these FOMs in order to produce a new TSV technique. For example, the basic Markov model suggested by Gleason (22) (see section 3) was analyzed further to produce several new FOMs relating to various aspects of BIT performance.

	METHOD	PROCEDURE	ACTION	REASON
TSV	New FOM	Extend Gleason's Model BIT Performance to extract a BIT FOM	Decided not to Develop	FOM's depend on parameters which must be estimated. Accurate estimation procedures are not available
	Test Repetition Methods	Extend Spillman's model of Intermittent Faults to assess their impact on BIT performance	Decided not to develop	Intermittent faults should be considered only after methods for hard faults are developed
	Extensions of FMEA	Apply Automated FMEA techniques to BIT evaluation	Decided not to develop	More work needs to be done on improving standard FMEA first
	Advanced Simulations	Develop functional level Simulation of BIT	Developed & reported in section 6	Behavioral level simulation tools are available
TSCA	Overlapping BIT	Develop multiple BIT systems which will provide self-checking capabilities to BIT	Developed & reported in Section 7	The theory of self-diagnostic systems could be used to develop Overlapping BIT circuits
	Fuzzy Pattern Recognition	Develop a Pattern Recognition Algorithm which will allow for automatic BIT evaluation	Decided not to develop	Pattern Recognition approach to this problem not well understood at this time
	Fault Insertion	Develop a new fault insertion device	Developed & reported in section 7	Fault Insertion provides a general and simple approach to TSCA

Table 4-1
Possible BIT Verification Techniques

One possible new FOM based on Gleason's model was suggested by the work of Spillman (29). Gleason's model of BIT systems contains four states, two representing correctly operating BIT and two representing fault states. Spillman's new FOM represents the average amount of time the operating BIT spends in the two correct states. It is determined by first calculating the mean holding time for each of the two BIT correct states and then averaging the two values. The mean holding time is the average amount of time that the system is in any given state. It is given by:

$$T_i = \frac{1}{w(i)} \ln \left[\frac{\overline{w(i)}}{w(i)} \right]$$

where T_i = mean holding time

$w_i = \sum_{j \neq i}$ (probability of transition from state i to state j)

$\overline{w}_i = \sum_{j \neq i}$ (probability of transition from state j to state i)

The naw FOM is the average of the mean holding times for the two correct states S_0 and S_2 in Gleason's model. It is defined as

$$A = (T_0 + T_2)/2$$

$$= \frac{1}{2} \left[\frac{1}{w_0} \ln \left(\frac{\overline{w_0}}{w_0} \right) + \frac{1}{w_2} \ln \left(\frac{\overline{w_2}}{w_2} \right) \right]$$

For example, consider a system in which the probability of system failure over an hour period is .01, the BIT coverage is .99, the probability of false alarm is .01 and the system is always repaired if a failure is detected. Figure 4-1 shows the state diagram and transition matrix for this example.

In this case,

$$w_o = \lambda_{FA} + \lambda_D + \lambda_U = .01 + .0099 + .0001 = .02$$

$$\bar{w}_o = 1$$

$$w_2 = 1$$

$$\bar{w}_2 = \lambda_D = .0099$$

Then $A = 1/2 \left[\frac{1}{.02} \ln \left(\frac{1}{.02} \right) + \ln (.0099)\right]$

≈ 95.5 hours

As with the discussion of Markov models in section 3.4, the probability of fault detection and the probability of a false alarm need to be determined by other means.

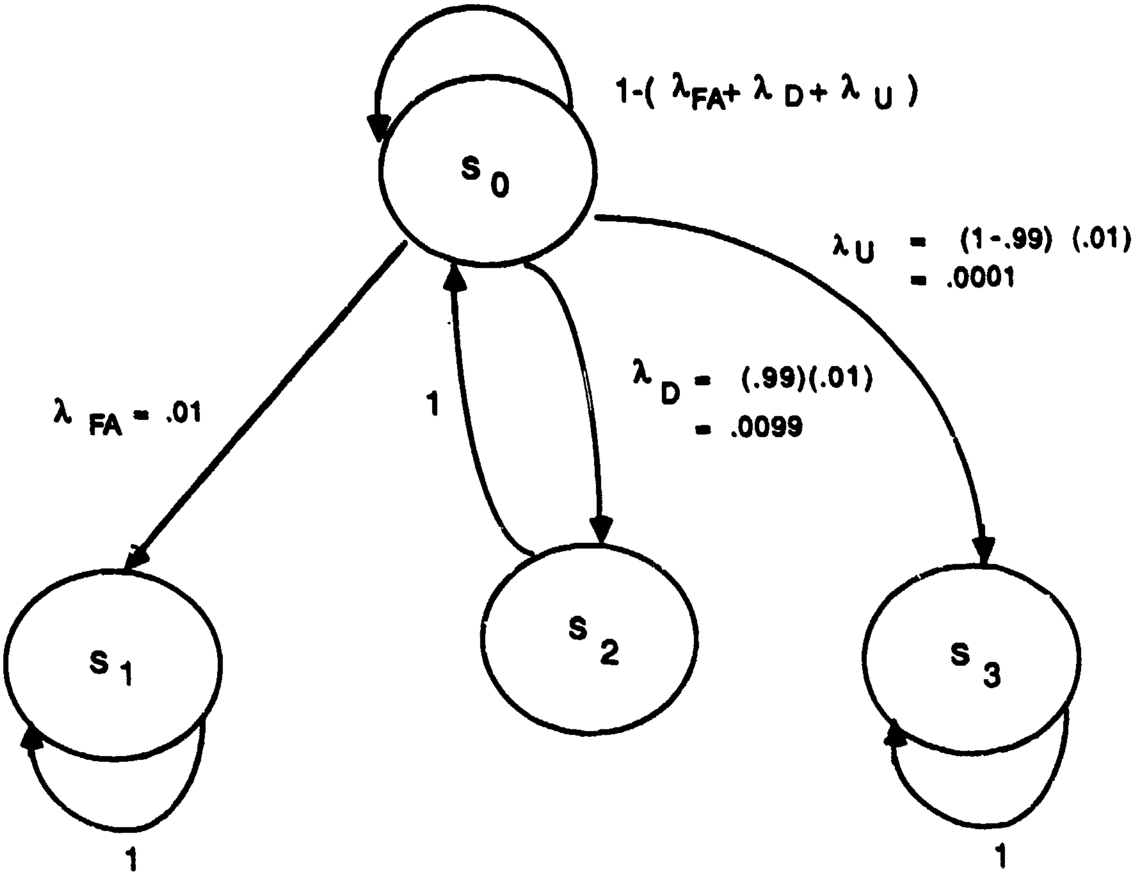
Beaudry (30) has also suggested some performance related FOMs that may be adapted for Gleason's Markov model. They include:

(1) Computational Reliability

(2) Computational Availability

(3) Mean Computations Before Failure

Computational reliability, R(t,T), is the probability that the system will correctly execute a task of length T started at time t. Computational availability is the average number of computations the system will correctly execute in a given time interval. The mean computations before failure, MCBF, is the average amount of computation available on the system before a failure. In terms of a BIT TSV technique, these



$$\text{Transition Matrix} = \begin{pmatrix} 1-(\lambda_{FA}+\lambda_D+\lambda_U) & \lambda_{FA} & \lambda_D & \lambda_U \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4-1 State Diagram and Transition Matrix for Example of Modified Gleason FOM

three measures could be interperted as BIT reliability, BIT availability, and the mean number of correct BIT decisions before a failure. All three of these measures emphasize the performance of a BIT system over time rather than the presence or absence of errors as in Gleason's model.

After a preliminary investigation it was determined that a BIT Figure-of-Merit approach would not provide an effective TSV technique. Before Gleason's model of BIT performance can be used to generate an FOM, the exact values of the transition rates between the four BIT states must be known. There are no known methods to accurately calculate these rates. They can be estimated to questionable degrees of accuracy from operational data. However, such estimates require long-term observation of the working system. Since the FOM would be used as TSV measure, the working system may not be available and if it is, the observation time may be excessive. As a result, further investigation of BIT FOMs as possible TSV measures was discontinued.

4.1.2 Test Repetition Methods

If the goal of a BIT system is to detect intermittent faults, it is necessary to allow the BIT system to apply its test repetitively so that the BIT will be operating when the intermittent fault is active. The problem from a TSV perspective is to verify that the BIT does repeat itself frequently enough to detect intermittent faults.

Spillman (40) has derived a formula describing fault behavior, from which the test repetition rate to detect intermittent faults can be calculated. The formula is based on the following two assumptions:

(1) **Well-Behaved Faults**
The assumption is made that the transition from a fault-free state to a faulty state and the transitions between faulty states occur instantaneously.

(2) **Signal-Independent Faults**
It is assumed that the occurrence of an intermittent fault does not depend on the nature of the signals in the circuit.

Both of these assumptions are commonly made to simplify the study of intermittent faults and they both seem reasonable in light of the observed behavior of intermittent faults. If intermittent faults occur at rate λ and disappear at rate μ, the probability of a transition from m to n "active" intermittent faults is given by equation (1). N is the maximum number of multiple "active" intermittent faults allowed in the system.

$$P_{nm}(t) = \frac{\alpha^{n-N}(\lambda-\mu)}{l\,(1-\alpha^{-N-1})} + \frac{2\alpha^{(n-m)/2}}{N+1}\left\{\sum_{j=1}^{N}\left[\frac{e^{\beta}}{1-2\alpha^{1\backslash 2}\cos(j\pi/[N+1)+\alpha}\right]A_j(\alpha)B_j(\alpha)\right\} \quad (1)$$

where $\alpha = \lambda/\mu$,

λ = the birth rate for intermittent faults;

μ = the death rate for intermittent faults;

$\beta = -(\lambda+\mu)t + 2t(\lambda\mu)^{1/2}\cos(j\pi/[N+1])$.

$$A_j(\alpha) = \sin(mj\pi/[N+1]) - \alpha^{1/2}\sin(j(m+1)\pi/[N+1])$$

$$B_j(\alpha) = \sin(nj\pi/[N+1]) - \alpha^{1/2}\sin(j(n+1)\pi/[N+1])$$

When $t \to \infty$, equation (1) becomes:

$$P_{nm} = P_m = \frac{(\lambda-\mu)\,\alpha^{n-N}}{\lambda(1-\alpha^{-N-1})} \qquad (2)$$

which represents the probability that the system evolves from having m active faults at some time to a steady state condition with n active faults. These expressions reduce to the single fault case found in the literature when N = 1.

The test repetition rate, K, is given by equation (3):

$$k = \frac{\log(1-CL_n)}{\log\left[\frac{1-[2-P_{nn}(t')]P_n}{1-P_n}\right]} \qquad (3)$$

where CL_n is the confidence level for the decision "the circuit is fault-free" if it passes all K test applications, and t' is the time between test applications. The derivation of equation (3) can be found in reference (43). For example, an engineer testing a circuit for intermittent faults may want to be 99% confident ($CL_n = .99$) of detecting a single fault in a triple fault environment with $\lambda = .008$ and $\mu = 1.0$. First, equation (2) would be used to calculate $P_1 = .00794$. If the test is applied one second apart, $P_{11}(1) = .36851$ from equation (1). The test repetition rate is found from (3):

$$k = \frac{\log(.01)}{\log(.99495)}$$

$$= 909.61$$

The engineer would apply each test in the test set to the circuit 910 times. If a fault is not detected then he would be 99% sure that the circuit is free of intermittent faults.

The test repetition rates produced by the model could serve as a guide for determining BIT performance under intermittent fault conditions. However, TSV approaches for permanent faults need to be better understood before exploring intermittent faults, so this concept was not pursued further.

4.1.3 Extensions of FMEA

The most popular Test System Verification method is Failure Modes and Effects Analysis. However, its use for TSV suffers from lack of standardized approaches and lack of accuracy. RADC's Automated Advanced Matrix FMEA effort has attempted to develop a system for automated FMEA with the objective of standardizing the technique and reducing the cost through automation. There is still need for additional work in the BIT evaluation portions of the Automated FMEA. Better identification of failure modes and their effects is still needed, especially for VLSI and VHSIC devices. After a preliminary investigation, it was determined that FMEA approaches need additional improvements before they can be adapted to a TSV techrique.

4.1.4 Advanced Simulations

Simulation techniques have the greatest potential to provide accurate and economical test system verification. Simulation has been used in the past for TSV and continues to be used extensively in the design of integrated circuits. However, its use for TSV at the subsystem level has become impractical because of the increasing use of VLSI devices. Past simulation techniques used gate level modeling, but for many VLSI parts the gate level models are not available. Also the complexity of today's and future subsystems would make gate level simulation impractical. A significant amount of computing resources would be required to test a reasonable fault population. Suppliers of simulation hardware and software have been addressing these problems, because the size of VLSI devices is putting greater demands on simulation capability. Advances are needed to increase the size of the system that can be modeled and to speed up the simulation. These include methods of simulating the independent effects of many faults at the same time, methods of removing equivalent faults from the fault set and the use of special hardware accelerators. These hardware accelerators are special-purpose computers designed to execute logic simulations and can increase execution speed by several orders of magnitude. A summary of the current state of the art of logic and fault simulators is given in references (31) and (33).

The above advances in simulation technology do not however address the problems of simulation at subsystem or circuit module level. The most significant problem is the enormous size of gate level descriptions of LSI and VLSI devices.

Two recent advances in simulation technology have potential for resolving these problems. The first is the use of actual hardware as part of the simulation and the second is the use of behavioral simulation.

4.1.4.1 Hardware Simulation

This approach models portions of the circuit using gate level simulation techniques and uses real hardware for the remaining portions. The motivation for this is to be able to simulate a subsystem or circuit module that consists of a combination of custom logic and off-the-shelf LSI devices. A diagram of this type of simulation system is shown in figure

4-2. The simulation host interfaces to a complex hardware adapter and the hardware adapter allows the hardware devices to interface to the rest of the simulation. The simulation system specifies whether each pin operates as an input or output or both. As the simulation progresses, signals are passed to the hardware devices and are received from the hardware devices. When performing fault simulation to determine test coverage, faults are inserted in the logic level simulation portion in the conventional manner, but faults for the LSI and VLSI devices can only be inserted at the interface pins of the hardware adaptor. Hence only faults at the I/O pins of the devices can be simulated not faults internal to the device.

This permits simulation of a subsystem which includes the use of LSI and VLSI devices to be developed with substantially less effort than would be required if the LSI and VLSI devices were modeled at the gate level.

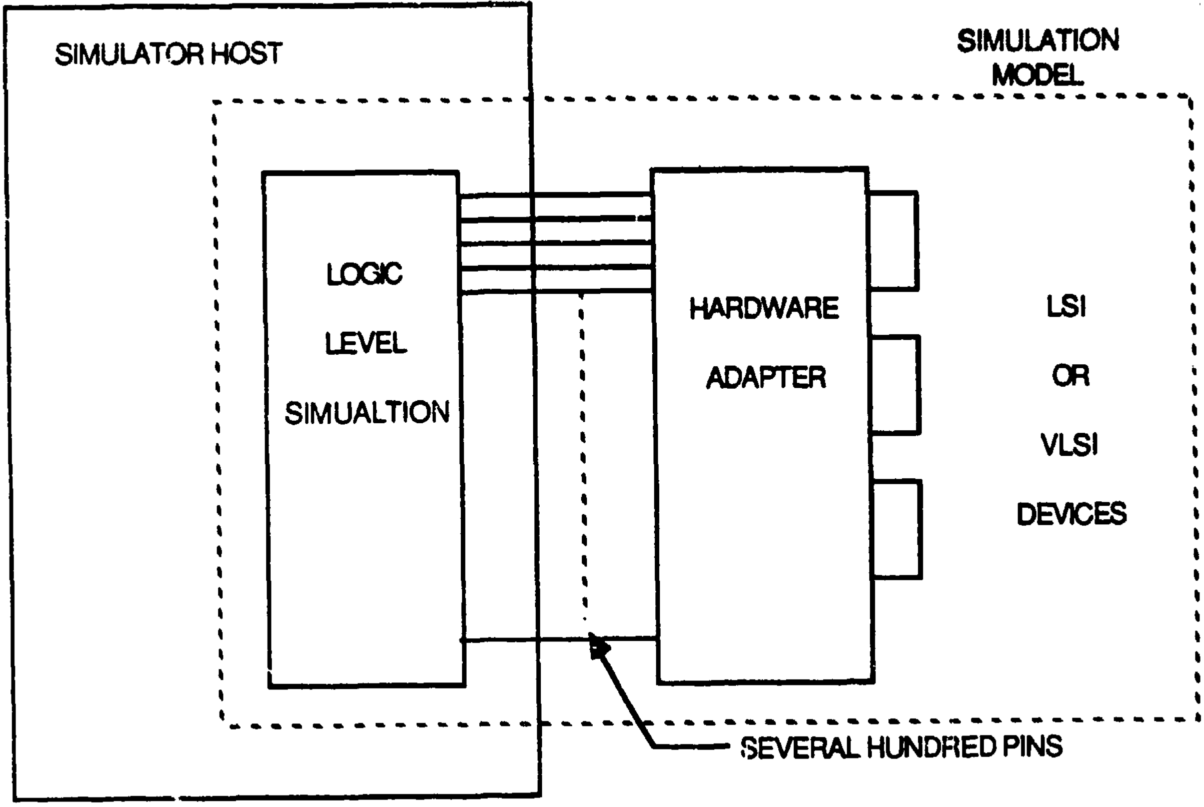


Figure 4-2 Simulation Testbed Incorporating Hardware Elements

4.1.4.2 Behavioral Level Simulation

There has been recent interest in development of higher level modeling capability for use in the design of LSI and VLSI devices. This is because IC designers need a tool to easily model a complex circuit early in the design cycle. Modeling at a higher level allows the designer to make architecture trades, check performance and permit parallel development of associated hardware and software without modeling at the detailed design level. Figure 4-3 illustrates the levels at which electronic systems can be represented and described. It also shows samples of simulation tools applicable to the various levels. There has been recent interest in developing behavioral level simulators for the design of LSI and VLSI devices. (34) (35).

Suppliers of simulation systems are developing behavioral level simulators to respond to the need. Most provide a special purpose modeling language based on popular general purpose programming languages such as C or Pascal, with an increasing interest in Ada. The behavior model languages need flexibility, but have to be structured for use by designers who are not expert programmers or simulation engineers. These behavioral level models can be integrated with gate-level models for a complete model of a subsystem.

There is additional need for improvements to behavioral simulation languages to address test system verification needs. First, continued work is needed in developing a structure that is easily usable by design and systems engineers. Second, a capability for modeling failure modes (no longer definable at the gate level) in a realistic manner is needed. This aids in understanding how various classes of faults affect the behavior of a component, subsystem or system.

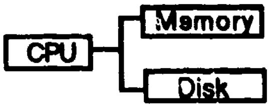
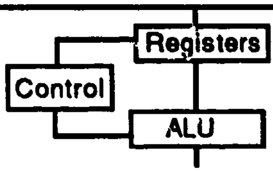
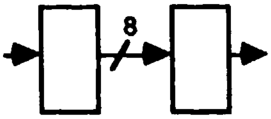
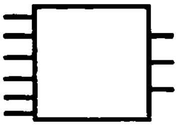
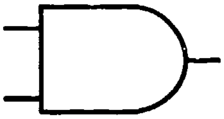
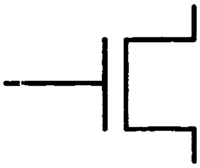
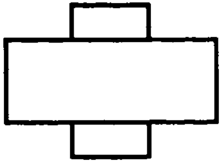
Level	Representation	Description Domain	Representative Tools
System		Major Hardware and Software Interfaces	SIMULA, SIMSCRIPT
Archi-tecture		Instruction Level	
Register		Bit-level Transfers	Behavioral Simulators
Function		Boolean Equations	
Logic		NAND, NOR Gates	TEGAS, LASAR
Circuit		Electrical Network	SPICE
Device		Diffusion Characteristics	SUPREM, SUPRA

FIGURE 4-3 General Levels of Simulation (35)

Related to the development of behavioral level simulators is development of Hardware Description Languages (HDL). HDL's are often based on general purpose languages such as C, Pascal or Ada and are used to specify the design of hardware. Key elements are: (1) specification at hierarchical levels (2) specification independent of implementation (3) libraries of part descriptions (4) companion simulators. As with behavioral level simulators, a capability for modeling faults needs to be developed for HDL's.

As part of task 5, new simulation techniques were investigated. These included modeling devices and their failures at a functional level higher than the gate level. The results of this work are reported in section 6.

4.2 TSCA TECHNIQUES

This section identifies some possible new TSCA techniques which have not been considered in the literature and provides a short description of each.

4.2.1 Overlapping BIT

One possible method of monitoring BIT performance in the field which does not have all the overhead of a simple duplication technique would be to divide a circuit into overlapping subsets and provide BIT for each subset. The result would be similar to a duplication approach in that a single BIT error message would have to be confirmed by other BIT circuitry monitoring the subset. If there is an inconsistent BIT response then there is a failure in one of the BIT systems. This method was investigated as part of task 5. A full report on the use of this approach is in section 7.

4.2.2 Fuzzy Pattern Recognition Applications

Another method of TSCA which has not be reported in the literature involves the application of pattern recognition algorithms to classify the output data of the BIT system and the circuit under test to determine overall system performance. The speculation was that by examining both the BIT output and several parameters of the circuit under test, it may be possible to classify the BIT performance. Development of this approach would require the identification of relevant BIT and system parameters to monitor as well as generating test cases from which a classifier could be constructed. Once completed the method could supply both BIT information and data on the monitored parameters, which could be used to determine both system performance and BIT performance.

It was determined during this study that (1) this approach required a complete understanding of all the operating parameters of the specific BIT system as well as those of the unit-under-test, preventing development of a general approach, and (2) the computation time for the algorithms was excessive. It was therefore not studied in further detail.

4.2.3 Fault Insertion

Fault insertion techniques for BIT evaluation have been suggested in the literature. However, the methods have not been fully developed and rely on the use of a large number of multiplexers which must be added to the design of the unit-under-test. A new simpler fault insertion device was developed and a complete fault insertion mechanism for TSCA was designed. A full report on this approach is contained in section 7.

5.0 TSV/TSCA Evaluation and Selection

This section provides the results of the TSV/TSCA evaluation and selection task. The original plan for this task was to select appropriate evaluation criteria and evaluate each of the TSV and TSCA techniques against the criteria using an evaluation matrix. Because of the differences of the various techniques and the interrelationship of the evaluation criteria, the approach for this task was modified as described in this section.

5.1 Rejected Techniques

Over the course of the contract, several new techniques considered for TSV or TSCA were rejected for further development or evolved into other techniques. The following is a list of these techniques and an explanation of why effort on them was discontinued.

a. p-t Diagnosability Models (TSCA) - Investigation into the application of p-t diagnosability models evolved into development of the new TSCA technique called overlapping BIT.

b. Self-checking Checkers (TSCA) - A variety of self-checking checker approaches were reviewed as part of task 2 and task 3. It was determined that it would take a considerable effort to develop any new self-checking checker system and it would be applicable only to a narrow class of circuits.

c. Birth-Death Models (TSV) - This statistical procedure was proposed for analysis of intermittent faults.

Work on evaluating BIT performance with respect to intermittent faults was de-emphasized (based on customer direction) in favor of the primary performance parameters of fault detection and fault isolation.

d. Fault-Insertion Circuits for TSV - Incorporation of circuitry to automatically insert faults for TSV analysis in a manner similar to fault-insertion circuits for TSCA was investigated. Although it had the prospect of providing the same technique for both TSV and TSCA, the faults inserted for TSV could not adequately represent the fault population without substantial hardware overhead penalty.

e. Automatic Test Pattern Generators (TSV) - Automatic test pattern generators produce fault coverage measures based on the test patterns generated. They work well for combinational circuits but their applicability to sequential circuits is limited. They can only be used for subsystems containing VLSI devices when detailed gate level models of the devices are available.

f. Test Repetition Methods (TSV) - These are primarily useful for detecting intermittent faults. They were not pursued because work on intermittent faults was de-emphasized based on customer direction.

g. Computational Reliability, Computational Availability and Mean Computations Before Failure (TSV) - Development of these new FOM's based on Markov Models was discontinued based on customer direction.

h. Pattern Recognition for TSCA - This evolved into the Pattern Recognition TSV method for analog circuits.

i. Symbolic Execution (TSV) - This method is similar to simulation approaches. However, at its present stage of development it requires extensive modeling of the system to be evaluated in software. It has been determined that the software development would be expensive and of limited usefulness.

j. Statistical Design Verification (TSV) - Since complicated mathematical concepts are involved, it is questionable whether the method can be used on a practical level. Extensive work would be required to make this method applicable to the problem of BIT verification.

k. Improved FMEA (TSV) - Improvements to FMEA to increase the accuracy of BIT evaluations involve better characterization of integrated circuit failure modes and effects. Effort to pursue this would be far more than required for improved simulation capability, and the end result would not be as accurate as with simulation.

5.2 Evaluation Criteria

The first step in the evaluation process was to select the evaluation criteria. This was done by generating a list of possible criteria, then reducing the list to those most relevant to the selection. The criteria were then grouped as either qualitative or quantitative and whether they are relevant to TSV or TSCA. Table 5-1 lists the evaluation criteria and indicates their grouping. Note that most of the criteria are common to both TSV and TSCA, but there are a few that are applicable only to one or the other.

TABLE 5-1 Evaluation Criteria

	TSV	TSCA
QUALITATIVE	**Benefits/advantages**	**Benefits/advantages**
	Limitations/disadvantages	**Limitations/disadvantages**
	Type of BIT	**Type of BIT**
	State of development	**State of development**
	Development risk	**Development risk**
	Level of applicability	**Level of applicability**
	Skill level requirements	**Skill level requirements**
	Technology impact	**Technology impact**
	Facility impact	
QUANTITATIVE	**Development & acquisition costs**	**Development & acquisition costs**
	Utilization costs	**Impact on operational system**
	Confidence level	**Fault coverage**

The following is a brief description of each of the criteria.

Qualitative Criteria

a. Benefits/advantages: strong points of the TSV or TSCA methodology.

b. Limitations/disadvantages: weak points of the TSV or TSCA methodology.

c. Type of BIT: kinds of BIT approach(es) the identified TSV or TSCA methodology will serve.

d. State of development: what needs to be done to fully develop the approach.

e. Development risk: level of risk involved in the development of the new TSV or TSCA approach.

f. Level of applicability: hardware implementation level and maintenance level at which the TSV or TSCA approach is useful.

g. Skill level requirements: for TSV, the skill requirements of the engineers and analysts using the system; for TSCA, the training/skill requirements of maintenance personnel using the system.

h. Technology impact: the economics and practicality of implementing a TSV/TSCA approach, based on technology trends.

i. Facility impact: special facilities required for TSV approaches.

Quantitative Criteria

a. Development and acquisition costs: cost to fully develop the identified TSV or TSCA approach; likely acquisition cost impact.

b. Utilization costs for TSV approaches: cost to a development program to use the TSV approach as a verification tool.

c. Impact on the operational system: how the TSCA system will change the weight, complexity, volume or other design characteristics of the operational system.

d. Confidence level (TSV): degree of confidence that the verification results will match the actual performance.

e. Fault coverage (TSCA): The probability that a failure in a BIT system will be detected.

5.3 Evaluation

TSV and TSCA technique options remaining after the initial filtering are shown in Table 5-2.

The final evaluation started by assessing each technique with respect to the individual evaluation criteria. TSV and TSCA techniques were examined separately. Tables 5-3, 5-4 summarize the evaluation comments for TSV and TSCA, respectively.

Once the evaluations against individual criteria were completed, it was necessary to develop an overall comparison measure combining the multiple criteria. Several approaches were considered. The initial evaluation approach was to use

TABLE 5-2 Options Remaining Afte [illegible] al Filtering

Test System Verification	Test System Condition Assessment
Figures of Merit (FOM)	Self-Checking Circuits
Failure Modes and Effects Analysis (FMEA)	Fault Insertion
Simulation	Overlapping BIT
Pattern Recognition	

Table 5-3 TEST SYSTEM VERIFICATION EVALUATION

	CRITERIA / OPTIONS	FOM'S	FMEA	SIMULATION	PATTERN RECOGNITION
QUALITATIVE	Benefits/ Advantages	Easy to compute, New FOM'S give better indication of BIT performance.	Easy to implement. Approach can be easily tailored to program requirements. Automated FMEA can standardize process.	Accurate results. Common usage for IC design. Repeatable verification results.	Good design and evaluation technique for analog BIT.
	Limitations/ Disadvantages	Requires predetermined parameters such as fault detection, false alarm rate, etc.	Results depend on individual analyst. Results inaccurate. Problems with failure modes of integrated circuits.	Requires model development and computer resources. Capability may not be available to all manufacturers.	Applies only to analog BIT. Produces a new FOM.
	Type of BIT	All Types	All Types	Digital only for gate level simulation. All types for behavioral level.	Analog only
	State of Development	Well developed	Needs development of automated FMEA and method for evaluating failure modes and effects of integrated circuits	Behavioral level simulation languages available now. Need development of fault simulation capability.	Needs further development for multiple fault mode distributions
	Development risk	None	None as is. High to improve capability	Medium	Low
	Level of applicability	All	All	All	Board level, limited subsystem level.

Table 5-3 TEST SYSTEM VERIFICATION EVALUATION (continued)

CRITERIA	OPTIONS	FOM'S	FMEA	SIMULATION	PATTERN RECOGNITION
QUALITATIVE	Skill level requirements	Low	Low skills required, but quality of results increases with skill.	Moderate skills required. Effective development of behaviorial language simulations will decrease skill requirements.	Moderate mathematical skills required.
	Technology Impact	None	Accuracy to decrease rapidly with technology trends toward larger scale integrated circuits.	Need to simulate at higher levels of abstraction with trends toward larger scale integrated circuits.	Less demand with trend away from analog implementations.
	Facility Impact	None	None in present form. Automated FMEA will require host computer or engineering work station.	Requires host computer or engineering work station.	Requires desk top computer for computation.
QUANTITATIVE	Development and Acquisition Costs	Small, documentation of standardized approach.	Small in present form (documentation). Moderate for automated FMEA, (continuation of recent efforts). Large to improve capability.	Large, development of fault simulation using behavioral language, implementation and documentation of standardized approach.	Small, development of multiple fault distribution mathematics and documentation of strandardized approach.
	Utilization Costs	Low	Low to high; depends on the scope of the analysis.	High for installation. Moderate thereafter.	Low
	Confidence Level	Depends on validity of input parameters.	Moderate	High	Moderate; depends on accuracy of probability distribution.

Table 5-4 TEST SYSTEM CONDITION ASSESSMENT EVALUATION

	OPTIONS CRITERIA	SELF CHECKING CIRCUITS	FAULT INSERTION	OVERLAPPING BIT
QUALITATIVE	Benefits/Advantages	Provide high coverage of BIT faults with minimum additional overhead.	Simple to implement. Moderately high fault coverage with minimum additional overhead.	High fault coverage. Inherent isolation capability for both system faults and BIT faults.
	Limitations/ Disadvantages	Specially designed for specific, limited applications.	Incomplete fault coverage for circuits added for fault insertion. Appiciation dependent techniques.	High overhead. Difficulty in partitioning subsystem and selection of BIT techniques to cover partitions.
	Type of BIT	Limited to BIT for coded circuits.	All	Generally applicable all types of BIT but types may be limited compatible types for specific applications.
	State of development	Mostly academic except for limited applications.	Well developed for numerous applications. Others easily developed.	Needs development of partitioning methodology and BIT selection guidelines.
	Development risk	None for existing techniques but high for new techniqes.	None	Moderate to high
	Level of Applicability	Limited to integrated circuit and board level.	All	All

Table 5-4 TEST SYSTEM CONDITION ASSESSMENT EVALUATION (continued)

CRITERIA	OPTIONS	SELF CHECKING CIRCUITS	FAULT INSERTION	OVERLAPPING BIT
QUALITATIVE	Skill level requirements	High	Low	Moderate to high
	Technology impact	Fewer potential applications at integrated circuit level due to increasing use of cell libraries and fewer potential applications at board level due to increasing use of VLSI devices	None	May be complimented by trend to partitioning VLSI devices.
QUANTITATIVE	Development cost	Moderate for any new applications plus documentation of standard approach	Small; documentation of standardized approaches.	Large, development of partitioning and Bit selection techniques and documentation of standardized procedures.
	Impact on system	0 to 50% more BIT	5 to 20% more BIT	50 to 100% more BIT
	Fault Coverage	95 to 100%	80 to 95%	95 to 100%

a matrix comparison. This would be done by weighting each criterion and scoring each option against each criterion. The total sum of weighted scores would then be compared. Because of the large number of criteria and their interrelationships another technique was investigated. This technique utilized an automated decision making computer program which had been successfully used by TRW to evaluate and select a high speed bus protocol for the Multibus Avionics Architecture Design Study (41). The automated decision making computer program has a valid mathematical basis and is easy to implement. One of its strengths is that it handles the interrelationships of the evaluation criteria in a simplified pair-wise comparison. It also provides a measure of the integrity and consistency of the pair-wise comparisons.

As the evaluation progressed, it became apparent the evaluation methodologies were not working because of the substantial differences between the TSV and TSCA techniques being evaluated and the interrelationships of the evaluation criteria. As a result a simplified approach was taken. This approach was to evaluate the remaining alternatives only for certain key criteria. This reduced the number of evaluation criteria involved, eliminated additional candidates and simplified evaluation of the remaining techniques. The following describes the results of that evaluation.

Test System Verification

The alternatives evaluated for TSV were:

a. Figures of Merit (FOM)

b. Failure Modes and Effects Analysis (FMEA)

c. Simulation

d. Pattern Recognition

The FOM approach was ruled out because new FOM's are undesirable (new standards for specification would need to be implemented), and they still rely on some method for determining the underlying performance parameters (e.g. fault detection rate). The pattern recognition approach was ruled out because of its applicability being limited to analog circuits. This narrowed the evaluation to the FMEA and simulation approaches.

The current FMEA approach requires no substantial development and is easy to use, but is inaccurate. Adequate improvements to the FMEA approach are impractical. They would require a mechanism of determining failure modes of IC's, what the effects are, and how they are detected. This would be more complicated than developing the simulation. Current simulation capability is inadequate for subsystem level TSV. Improvements to simulation capability would provide a practical, accurate TSV tool, but there is some development risk. The real trade is between current FMEA approach and improvements to simulation. Since an accurate method was required, the FMEA approach was unacceptable. It was decided that the best option was to proceed with the development of improvements to the simulation capability to provide for accurate subsystem level TSV.

Test System Condition Assessment

The alternatives evaluated for TSCA were:

a. Self-Checking Circuits
b. Fault Insertion
c. Overlapping BIT

The self-checking circuits approach was ruled out because of the limited applicability of any particular implementation. This narrowed the evaluation to fault insertion and overlapping BIT.

Fault insertion provides a lesser capability than overlapping BIT but does not require additional development other than generating a handbook of techniques and involves minimal additional BIT overhead. Overlapping BIT provides high performance capability but requires more BIT overhead than fault insertion and involves additional development risk. Development of overlapping BIT is recommended, and both approaches should be documented. The approach selected for a particular application will depend on its requirements. An application with frequent maintenance, BIT overhead limitations or reduced fault coverage requirements can use fault insertion. An application with infrequent maintenance and requiring high BIT performance can use overlapping BIT.

Summary

It was recommended that improvements to simulation capability be developed for TSV, and overlapping BIT be developed for TSCA. In addition, the various fault insertion applications should be documented.

6.0 TSV IMPROVEMENT DETAILS AND SPECIFICATIONS

As discussed in chapter 5, the evaluation of techniques for Test System Verification led to the conclusion that simulation techniques, in particular behavioral level simulation, showed the best promise for evolution toward a practical, consistent, usable method for TSV. In the TSV portion of Task 5, these techniques were investigated further to try to validate these conclusions by analysis and to determine what kinds of improvements were necessary. This chapter describes the details of this analysis and includes a specification for use of simulation as an effective TSV tool.

A review of work on behavioral or architectural level simulation at the system level revealed few relevant papers, but the work of two groups seemed particularly appropriate. One was the group at the University of Illinois, led by Jacob Abraham, and the other was a group at AT&T Engineering Research Laboratories. Both groups had published papers on functional fault modeling of microprocessors and had used this technique in conjunction with architectural level microprocessor simulation. The analysis performed in this task extended this work to a simple computer system consisting of a microprocessor, RAM, ROM, and I/O. BIT was added to the system model and a conventional fault simulator on a CAE workstation environment was run to estimate BIT coverage. Section 6.1 dicusses the methodology for using functional fault modeling for TSV. Section 6.2 provides the details of the modeling performed on the contract, covering the example system model (6.2.1), BIT design (6.2.2), fault model (6.2.3), simulation environment (6.2.4), and the results of the simulation run (6.2.5). Section 6.3 discusses key issues surrounding use of these techniques. Finally section 6.4 contains a specification for the development and use of architectural fault simulation for TSV at the system and LRU level.

6.1 Methodology

The objective in architectural level functional modeling is to develop system and BIT models consisting of high level building block elements and their interconnections. BIT effectiveness is measured by inserting functional faults into the system and checking to see if the BIT model can detect them. A fault simulator which can handle functional models is used to trace propagation of the faults through the system. This methodology is not limited to verification at the end of the design cycle; it can be used at any point in the design cycle, with the level of detail in the model limited by how much is known about the design at each phase. A key requirement for this procedure to be a real check on the design is that the development of the models and selection of the list of faults to be simulated be done by analysts working independently of the the design team. The following discussion provides more details on the types of models and faults utilized and on the procedural aspects of this methodology.

To develop the model, the unit under test is functionally decomposed into a complete set of building blocks that represent data processing, storage and communications elements, plus any hardwired logic and interfacing in the system. Each element in the model is characterized in terms of the information it receives, processes and transmits. Included in the description are: (1) locations for data storage, such as registers in microprocessors and memory cells, (2) functions for obtaining or transmitting data, such as read or write instructions, (3) functions for processing data, such as arithmetic or logical instructions, and (4) control and timing functions. The BIT system whose design is to be verified is modeled in the same manner. If a hardware comparator is used, for example, then the model will include a compare function. In addition, an interconnection model is developed to describe data transfers between the various building blocks. The level of detail in these models may vary considerably from case to case, depending on the type of system, portion of the design cycle at which verification is being performed and requirements of the simulator being used.

To perform fault insertion in the modeled system, a list of functional faults for each element of the system must be assembled. The types of faults in this list include (1) incorrect operations, including substitution of one operation for another, no operation performed or two operations combined, (2) fetching data from or transmitting data to the wrong place, such as a read from a different location than requested, and (3) control or timing faults.

Simulation software, capable of (1) representing such a system in the computer and (2) exercising fault models, needs to be provided. This software may already have models of various system primitives or even the building block elements themselves. Alternatively these models may have to be developed by the analysts working on the particular system. Ultimately, having a widely usable library of primitive models available is highly desirable. Commercial simulators are available to provide this capability, but many of them currently run on workstations and do not provide the speed required to do this analysis in a reasonable time. This issue is discussed further in section 6.3.

To evaluate the performance of the BIT, BIT coverage is calculated by exercising the model with each fault in the fault list, determining whether or not BIT detects the fault, and computing the ratio of faults detected to total faults utilized.

Figure 6-1 illustrates how this methodology can be put into practice. The designers provide design details to a simulation analyst and a test analyst. First the test analyst independently reviews the design and develops a list of potential faults in the system. The simulation analyst then (1) decides on the functional decomposition of the system, (2) develops models of the building blocks and interconnections, and (3) develops functional models of the faults in the fault list developed by the test analyst. The test analyst then (1) selects a subset of the fault list to use for fault insertion, (2) runs the simulation, (3) analyses the results, and (4) provides feedback to the designer(s) if the design fails to meet specifications or if unusual behavior is detected.

The key assumption being made here is that fault coverage, when computed using data from a functional model, will accurately represent the physical behavior of the delivered system. The example system discussed in section 6.2 was developed to test this hypothesis in a single case. Future work that needs to be done prior to any widespread utilization of this concept includes validation, such as running a complete model of a real system and comparing with field data, or comparing architectural level and gate level models of a reasonably complex system.

6.2 Architectural Simulation Test Case

To determine the feasibility of using behavioral level simulation for calculating BIT coverage, an example consisting of a simple computer system was analyzed with a conventional fault simulator running in a workstation

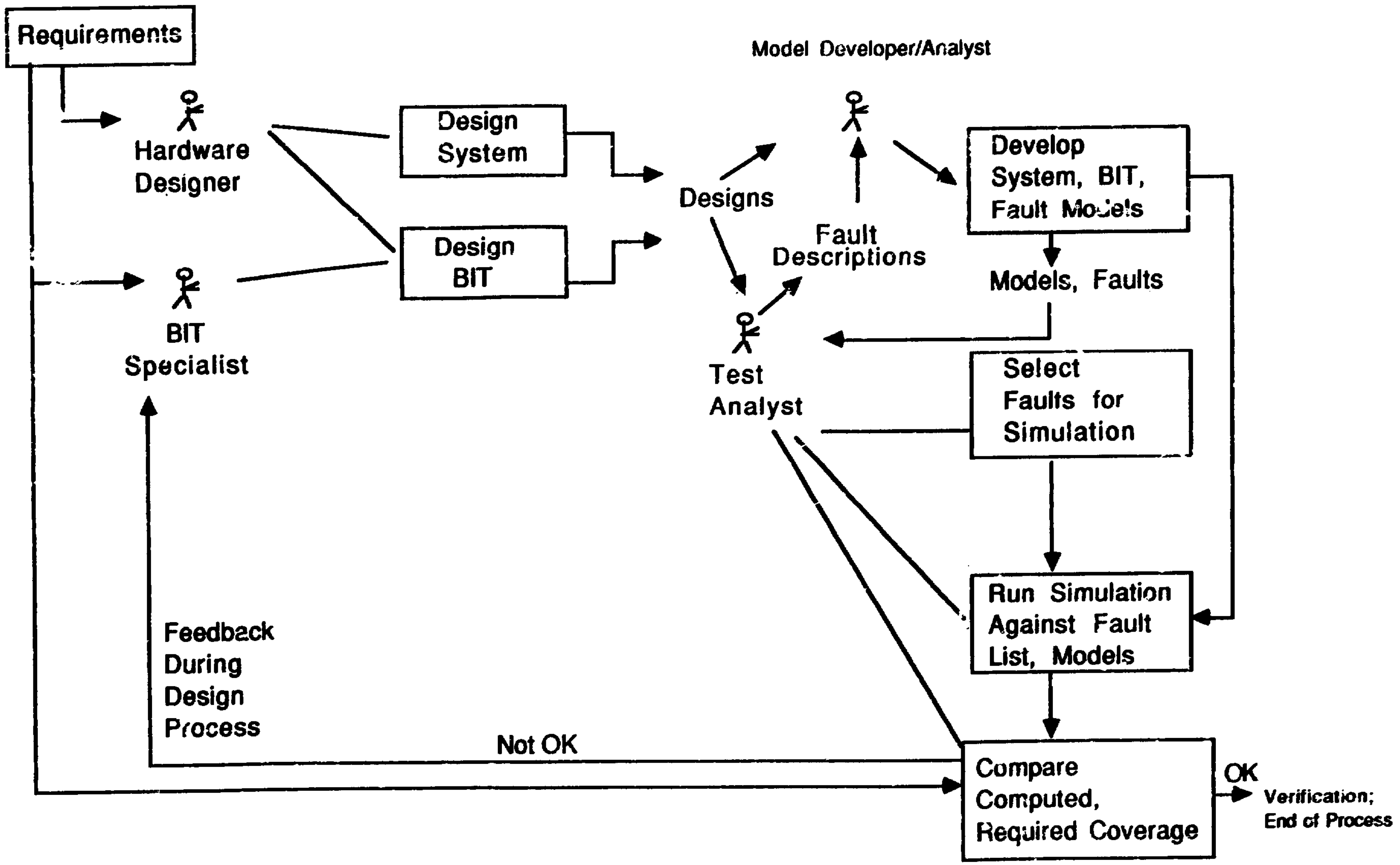

Requirements
Hardware Designer
BIT Specialist
Design System
Design BIT
Designs
Model Developer/Analyst
Fault Descriptions
Develop System, BIT, Fault Models
Models, Faults
Test Analyst
Select Faults for Simulation
Run Simulation Against Fault List, Models
Feedback During Design Process
Not OK
Compare Computed, Required Coverage
OK
Verification; End of Process

Figure 6-1 Design/Analysis Flow for Simulation Use in TSV

environment. The system modeled in the test example included a processor, ROM, RAM and I/O. A 16-bit information bus (address, data) allowed communication between the various modules.

6.2.1 Example System and Model

The following discussion of the example system covers the system elements and how they were modeled, along with some details on how the operation of the system was modeled.

6.2.1.1 Description of System Model Elements

The simulated processor was based on the Fairchild F9450. Its instruction set included a subset of the MIL-STD-1750A instruction set and contained one hundred sixty three opcodes. MIL-STD-1750A floating point operations and some executive control functions were not included. The processor executed NOPs (no operation) if an illegal opcode was received.

ROM contained 61 16-bit words starting at address 0000 hex and contained only the assembled BIT routines that were to be tested by this exercise. Data and address buses had separate inputs to the ROM. Not chip enable input NCE was pulled low during a read cycle.

RAM contained 4096 16-bit words starting at address 4000 hex. Input and output lines were the same as the ROM model with an additional RNW (read and write) line controlling the direction of data flow.

The I/O model contained two 16-bit ports starting at address

C000 hex. One port was an output port and one port was an input port. All control lines were the same as the RAM, with the RNW line selecting the input or output port.

Each module had a fault clock input which could be activated during simulation to insert faults into that module.

Since both addresses and data were carried on the same information bus, it was necessary to add an address latch to the model to retain the address generated during an addressing operation so the peripherals were properly addressed during the ensuing data clock cycle(s).

For data to pass between the processor and its peripherals in a normal manner using the simulation software selected, the operation of address, data, and control lines had to be modeled in more detail than by just specifying a data transfer to or from a certain address. It was necessary to indicate when the information bus carried data and to control the latching of addresses. A processor strobe output (STRBA) was used to perform these functions. One clock cycle was required every time address lines were set, data was written, or data was read. Once all operands necessary for the execution of the pending opcode had been loaded, all internal functions of the processor were executed without any delay.

On power up the processor addresses ROM location 0000 to fetch its first instruction. It then follows the logic determined by the ROM instructions (correct or faulted) until all instructions are executed.

In each module, if no internal register or memory location is selected during a read cycle due to a fault, a word consisting of all ones is used as data. If no internal register or memory location is selected during a write cycle due to a fault, no data is written anywhere. If more than one internal register or memory location is selected during a read cycle due to a fault, the resulting AND of all selected registers or memory locations is returned. If more than one internal register or memory location is selected during a write cycle due to a fault, the same data is written to all registers or memory locations selected.

6.2.2 Built-In Test Description

Built-In Test for the demonstration system consisted of a series of off-line self-test routines. The BIT routines included a checksum test on the ROM area of memory, a checkerboard write/read check of RAM, and a wrap-around check of the I/O ports (output port driving the input port).

A BIT module was added to the simulation to report the result (pass or fail) of the BIT tests. This module simply consists of a latch which is activated when data is sent to address 8000 hex upon detection of an error. This latch is reset at the beginning of each simulation cycle in which a new fault was introduced.

The BIT module also monitors the address bus. A write to address 8000 indicates that a BIT routine internal to the processor has failed. If 8000 is accessed, the BIT module output will latch to a logic one.

6.2.2.1 ROM Checksum Routine

In this routine, all the (unsigned) values of ROM are added together and the result is compared with the known, good result. After resetting the accumulator, the checksum routine starts by setting a pointer to the highest ROM address and adding the ROM data at that location to the accumulator. The pointer is then decremented and the addition is repeated until the pointer reaches address 0000. The data in the accumulator is then compared to a good checksum (stored in the next ROM location beyond the data tested). If they are not equal, accumulator value is written to address 8000. This latches the BIT output and also makes the calculated checksum available for troubleshooting purposes.

6.2.2.2 RAM Checkerboard Routine

The checkerboard routine consists of successive write-then-read and compare operations across all of the RAM. It starts by writing AAAA to the top address in RAM. It then reads back data from the address just written. If there is a difference, address 8000 is written to latch the BIT output. If no error is found, the routine continues by writing and trying to read 5555 at the same address. If there is still no error, an address pointer is decremented and the write/read tests are run on the next lower address. This repeats until the lowest address of RAM is tested or an error has been detected. AAAA and 5555 are used as complementary patterns of alternating zeros and ones.

As can be seen from the test results, the checkerboard routine was not a good BIT technique for RAM. A parity test would be a better BIT candidate.

6.2.2.3 I/O Wrap-around Routine

The idea behind the wrap-around test is to provide switches at the external interfaces that allow inputs to be connected to outputs for testing. In the simulation model, the output port is connected to the input port during the I/O test. A write-then-read test is performed, first with AAAA, then with 5555. In each case the value is written then read. If the value read back is not the same as the original value, address 8000 is written to latch the BIT error output.

6.2.2.4 General Comments

Only a limited number of registers and instructions were to be faulted in the demonstration, so no explicit test of registers or instructions was made. The above routines were used to indirectly test registers and instructions. This resulted in all except four of the registers being used in the above tests.

The fault simulation run on the demonstration system was performed using a logic simulator. The behavioral level models were develcped to include a fault clock input. By forcing N transitions at this input the model would modify its behavior to simulate the Nth fault in a list of faults was activated. The net effect of this procedure was the running of a serial fault simulation. The advantage gained by using this procedure was the ability to describe faults at a functional level and not merely as stuck-at faults that are commonly found in gate level fault simulators. The main disadvantage of this procedure is the fact that serial fault simulation requires a large amount of computation time.

6.2.3 Fault Model Description

The processor was faulted using stuck-ats on output lines, improperly decoding registers, and improperly decoding instructions. Improper decoding of registers included using an incorrect register, no register, or using the AND of a correct register and an incorrect register. When more than one register was used as a source the data from the selected registers was ANDed. When more than one register is a destination, data was sent to both. Improper decoding of instructions was modeled as using an incorrect instruction or using no instruction.

Fault models for the I/O, ROM, and RAM modules were similar to the processor fault model. Incorrect decoding of addresses in the memory modules is analogous to register decoding errors in the processor. Stuck-at faults at the output pins of both the memory and I/O modules were used as well.

Timing estimates made it clear that the simulator used for this demonstration is much too slow to be practical for a large scale simulation. For this reason, only a sampling of the described faults were actually simulated. The actual faults simulated for the processor include:

(1) incorrect decoding of three registers (1, 5, and 12)
(2) incorrect decoding of the add and subtract opcodes
(3) stuck-at faults on the information bus, STRBA, STRBD, and RW lines

Stuck-at faults were applied to the data bus of the I/O, RAM, and ROM modules. In the ROM and RAM models, faulty decoding of 50 addresses consisted of pointing to a neighboring location.

After the subset of the total fault set was selected to be run for the system, a list of the selected faults was compiled as input to the simulation run. The simulator was set up to run through each fault in the list until all of the faults had been simulated. A fault list pointer for each module provided the mechanism to activate the faults one at a time, using the fault clock input pins in the processor, I/O, ROM, and RAM modules.

6.2.4 Simulation Software Model Description

The simulator used for this test is an event driven simulator. When an input to a modeled device changes, the simulator calls a software procedure to determine what effect the change of input should have on the output of the modeled device.

For example, if an OR gate input changed from a one to a zero, the simulator would call a procedure that modeled OR gates. The procedure would determine the other inputs to the OR gate. If they were all zero, it would schedule a zero to appear at the output of the OR gate after the proper propagation delay time. With any other input equal to one, no action would be taken to change the output. Had the input gone from a zero to a one, the procedure would have checked the present output. If the present output was a zero, an output of one would be scheduled after the proper propagation delay. If the present output was one, no action would be taken.

OR gates, inverters, and latches are all examples of primitives for which the simulator has predefined procedures. The term primitive indicates that the device modeled is not made up of other lower level device models but is the lowest level model. In the systems we modeled, the processor, I/O, RAM, ROM, and BIT models are all primitive models. They are not made up of lower level models. However, their models are procedures defined by the ucer of the simulator.

6.2.4.1 Processor Model Procedures

The processor procedure takes action when the CLK or FAULT_CLK input pins go high. All other input pins to the processor are read at appropriate times but do not directly affect processor outputs whenever they change.

When FAULT_CLK goes high, a fault counter is incremented. This fault counter is accessed during normal simulation of the processor to determine if a fault is activated (if the processor's control procedure should be modified). No other action is initiated by the FAULT_CLK input.

When CLK goes high, one of two subprocedures is called. If an instruction is not in the process of being loaded through the information bus, then the decode instruction subprocedure is called. Otherwise the load next instruction subprocedure is called.

Several counters are used to keep track of the loading of an instruction. One counter keeps track of which word of the instruction is being loaded. The other keeps track of which portion of the read cycle the processor is in (addressing of memory or reading of data).

When CLK goes high after an instruction has been completely loaded the decode instruction subprocedure is called. The decoding of instructions is accomplished by a series of case statements that eventually call one of 67 subprocedures (one for each of the 66 instructions implemented, regardless of addressing modes, and one for handling illegal instructions). The instruction is executed after being decoded. If an external write is required, counters are used to keep track of the write cycle in the same manner as in the external read cycle.

Only one output change occurs as a result of each positive transition on the CLK input. In a read cycle the first CLK transition causes address lines to be set and the STRBA line to go high. The second clock allows the data on the information bus to be read by the processor. The counters described in the proceeding paragraphs are critical to controlling the read and write cycles that extend over several CLK transitions. These counters control the flow within the external read and write subprocedures.

Faulting subprocedures are accessed in several places. After an instruction is loaded, a subprocedure is called to fault that instruction if the fault counter indicates that the instruction just read should be faulted. If a fault should occur, the correct instruction is replaced by an incorrect instruction and the model continues operation, simulating the incorrect instruction.

Similarly, when a register is to be accessed for a read or write operation, a register faulting subprocedure is called. If the fault counter indicates that the register should be faulted, the subprocedure will replace the correct register with an incorrect register.

Output lines are faulted by checking the fault counter and masking the output data with stuck-at lows or stuck-at highs as required.

6.2.4.2 Peripheral Model Procedures

The procedures for the ROM, RAM, I/O, and BIT are essentially the same. They are all modeled to operate as memory devices. The I/O model is different in that it reads and writes to output ports instead of memory locations. The BIT model has an output that will latch high if written. The BIT also has a reset input that the other models do not. The ROM model cannot be written to and has its memory locations predefined. In every other sense these models are the same.

The FAULT_CLK input performs the same function in these models as in the processor model. For every positive transition the fault counter is incremented No other action is initiated by the FAULT_CLK input.

A change on any other input line will cause a logging of the time that the input changed. This will be used to verify setup and hold times. If setup and hold times are violated, an error message is printed. A main control subprocedure is then called. The current state of each input is determined. Based on the inputs, the data lines are either tri-stated, read, or written.

Faulting subprocedures are used in the same manner as in the processor model. Before output lines are driven, the fault counter is checked and output lines are masked with stuck-at lows or stuck-at highs if required. When reading or writing a memory location, a faulting subprocedure similar to the processor's register faulting subprocedure is called. If the fault counter indicates that the correct address should be faulted, an incorrect address is returned and used in place of the correct address.

6.2.5 Results

The intent of the analysis was to compare the results of the simulation run with the coverage value predicted for a design of this type by a designer with extensive experience with BIT designs for computer systems. Table 6-1 shows the test performed and the predicted coverage value for each system element, the expected contribution to the overall failure rate of the element, and the total fault detection rate expected for the system. Since there were some discrepancies between the types of BIT tests for which predictions were made and tests run in the simulation, comparison on the

Table 3-1: BIT Coverage Predictions

Hardware Element	Type of BIT Test	Contribution to Failure Rate	Fault Coverage	Adjusted Rate
RAM	Parity	20%	98%	196 %
ROM	Checksum	50%	99%	495 %
CPU	Instruction Test	10%	90%	090 %
I/O	Loop Back & Watchdog Timer	20%	95%	190 %
				971

Predicted Module Fault Detection Rate = 97%

overall system level was not possible. Comparisons were made on an element by element basis, and these are summarized in table 6-2. For the ROM and the CPU, good agreement was obtained. For the RAM and I/O, the tests used in the predictions were not implemented or only partially implemented in the simulation model, leading to large discrepancies in the results.

The good agreement for the ROM and CPU cases leads to optimism regarding the use of this methodology for TSV. It also suggests that exercising a complete fault list may not be necessary in all cases. A deeper understanding of how to select fault subsets for different kinds of elements and BIT techniques is needed, as is additional validation work on this methodology in general.

6.3 Key Issues in using Behavioral Fault Simulation to Verify BIT Coverage

In addition to the need to further validate this methodology, several important issues concerning its use and the potential availability of applicable tools arose during the study. These issues and some improvements in functional modeling which would help resolve them are summarized in Table 6-3 and discussed in more detail below. The development of improved behavioral simulation tools, along with better support environments for model development, will allow us to address many of the issues raised above. Much of this will fall out of progress which will be made in due time by commercial software and hardware developers. Other items will require added incentives to develop, as indicated in the discussion that follows.

Table 6-2: BIT Coverage Comparison:
Simulation Results vs Predicted Values

Hardware	Predicted	Values Obtained	Comments
CPU	90%	86.5%	-
ROM	99%	99.4%	-
RAM	98%	-	Parity test was recommended, but write-then-read test was implemented
I/O	95%	57.8%	Watchdog timer was recommended, but not implemented. Information bus faults caused program redirects which would have been caught by watchdog timer, yielding coverage close to prediction

Table 6-3. Issues and Improvement Areas

Major issues	Potential Improvements
Engineering effort to develop models	Development of generic functional models Decoupling of functional model and fault model
Simulation time to run models	Use of supercomputers to increase simulation speed Use of concurrent fault simulator Extrapolation based on simulation of a subset of the complete fault set
Relationships between physical, gate level and architectural level faults	Additional research on correlating functional and real world faults

In our example, the functional models, written in Pascal, took one man-month to write. The models are not detailed in the functions that they simulate. Testing of models was not done in depth due to time constraints. It is estimated that an actual system containing a MIL-STD 1750A processor, RAM, ROM, I/O, and BIT would take six to twelve man-months to properly model and completely test using current techniques. Realistic models of more complex systems would take much longer. This modeling time must be significantly reduced to consider this a practical verification technique. To improve model development time, generic functional models should be developed. A generic functional model for a processor would allow for simple entry of device specific parameters such as bus widths and instruction decoding. Compilation of the generic models would then be used to eliminate unused portions of model code and to optimize the speed of the simulation of those models. Use of generic forms for common elements such as processors will reduce time to model and test models by at least an order of magnitude. It is expected that the industry will develop this for its own use whether or not it is used in fault simulation.

In addition, study in the area of removing the fault model completely from the behavioral model of the device is needed. In this demonstration there was a tight coupling of the functional model and the fault model. It would be more desirable to define the functional faults separately from the functional model of the device. The faults to be tested should be defined at the time of the simulation. Functional faults could be defined within the generic functional models and be activated randomly by the simulator. Defining faults independently of the device modeled will ease the modeling process substantially.

Speed of simulation is the largest problem faced. In the small system that was defined for this demonstration we were able to define over 150,000 functional faults. Only a small portion (673) of the possible functional faults were simulated in our example. Actual simulation time was approximately 35 hours at an effective rate of twenty simulated processor clock cycles per second. Simulation of all defined faults would have taken approximately 2.9 million hours using a serial fault simulator on a workstation. It will be necessary to realize at least five orders of magnitude improvement in simulation speed to make this practical for more complex systems. Use of mainframe supercomputers would improve this by two orders of magnitude. Other areas where improvements can be made include (1) identifying equivalent faults so only a select subset of possible faults need be simulated and (2) implementing some form of concurrent fault simulation. An architecture linking a workstation environment for engineering model development and analysis post-processing to a supercomputer running the fault simulations may have to be implemented to solve the speed problem. The concern here is that only large industries will be able to afford these machines, limiting use of these techniques.

Concurrent fault simulators only simulate areas of the circuit that contain deviations from a fault-free simulation. Much less time is spent resimulating unaffected circuitry. CHIEFS is a concurrent, hierarchical and extensible fault simulator presented by W. A. Rogers and J. A. Abraham at the 1985 International Test Conference (39). CHIEFS uses a detailed model when simulating faulty sections of a circuit and higher level functional descriptions for unfaulted sections of the circuit. While CHIEFS still requires gate level descriptions of the circuits and is not directly applicable to this project, its hierarchical approach to simulation is an idea that should be incorporated in every simulator. Investigation is needed to determine whether concurrent fault simulation can be performed on behavioral level models. This capability is less likely to be developed in the near future by industry.

Another major area which needs resolution involves the correlation of functional faults to real world faults and the definition of fault coverage when measured using functional faults. If the functional faults simulated completely describe the universe of real-world fault manifestations, and a very high percentage is detected, then a very high percentage of real-world faults will be detected. If 50 percent of the functional faults are detected, it is not yet possible to claim any percentage of detection with certainty. Further research, experimentation and analysis is needed to establish relationships between physical faults, gate level faults and functional faults, in order to come up with a coverage formula at the architectural level that accurately and consistently represents reality.

There is one other major consideration when using functional fault simulation to verify the design of BIT. The BIT cannot be verified when it runs concurrently with the system. Simulation time would be astronomical if this were allowed. This is not to say that the BIT itself cannot be a concurrent BIT. If separate routines can be written and shown to completely exercise the BIT as it would be exercised in normal operation of the system, then using these routines to verify BIT "offline" would be sufficient. More confidence in the test results would be gained if those routines used to test the BIT were actually used as a startup self-test and not relied on as operating the same as the concurrent BIT routines.

6.4 Architectural Level Simulation for TSV: Specification for Further Development

6.4.1 Objective

The objective of this specification is to scope future development to resolve issues regarding the use of architectural level/functional simulation as a practical and economical Test System Verification (TSV) technique, and to develop an architecture for implementing this technique.

6.4.2 Scope

This effort will involve:

(1) Performance of validation tests for simple systems, sufficient to instill confidence regarding the correctness of the methodology and its utility in the TSV process.

(2) Research into the relationships between physical gate level and architectural level faults.

(3) Investigation into improvements to hierarchical, concurrent fault simulators which have potential for application to TSV.

(4) Specification of a computer system or systems for implementing architectural level/functional simulation as a practical TSV tool. This includes identificaticn of both hardware and software that would need to ba acquired, plus specification of any additional software development required. The specification will also

identify training requirements for system users. A timeline for implementing an operational facility utilizing such a system will be incorporated into the document.

6.4.3 Background

The current effort surveyed approaches and techniques for Test System Verification which were in use, under development, or proposed as of 1985. The objective was to identify techniques which were effective, accurate, practical and economical, and to pursue initial improvements to the methods which showed the most promise. As a result of the techniques evaluation, architectural level/functional simulation was selected as the TSV technique meriting further investigation. An initial test case for a simple system was developed and BIT coverage was computed using fault simulation in a workstation environment. The results compared favorably with coverage predictions by experts, indicating potential for further development.

A number of issues were raised by this analysis, including:

(1) the need to validate results

(2) how to select a subset of the total fault set to provide a representative set of faults, based either on

(a) knowledge of the relationships between functional and physical faults, or

(b) statistical evidence on the frequency of occurrence for various fault types

(3) the need to drive simulation time down by 5 orders of magnitude

This specification scopes an R&D effort oriented toward providing answers to these issues.

6.4.4 Tasks/Technical Requirements

The specified effort consists of four technical tasks plus associated documentation and presentations.

1) Perform validation tests.

Select two systems or subsystems as follows:

(a) a system for which gate level models are available.
(b) a system for which field data on failure occurrences and BIT coverage are available

This task involves developing an architectural level/functional model of the system and comparing BIT coverage obtained from architectural level fault simulation with that obtained by gate level fault simulation in case (a) and the field data in case (b). The chosen systems shall be representative of those in new military applications but be simple enough to permit modeling and simulation within 10 man-months for each case.

2) Perform research into the relationships between physical, gate level and architectural level faults. The task consists of two parts:

(a) Identify past and current research efforts in this area and summarize the key conclusions
(b) Determine open areas for research, identify promising solutions, perform initial research leading to a long range research plan

3) Develop a methodology to allow accurate computation of BIT coverage using a subset of the complete fault set (fault subset) as input to a fault simulation. This included determining:

(a) How to select a suitable fault subset (using classes of equivalent faults, or by identifying representative faults for certain classes of architectures, structures or behaviors)

(b) algorithms for extrapolating total coverage from data generated by a simulation based on use of fault subsets

4) Identify hierarchical, concurrent fault simulators with potential application to TSV. Identify improvements which would most likely allow achievement of the TSV goals. Perform initial research on the most promising improvements.

5) Specify a computer system architecture for implementing architectural level/functional simulation as a practical TSV procedure. The specification shall include both hardware and software. For software extending currently available capability a requirements level specification shall be provided. Both functional and performance specifications for the hardware shall be given.

) The results of the tasks described above shall be ocumented in technical reports and presentations.

6.5 Specification for Utilization of Architectural Level Simulation for Test System Verification.

6.5.1 Scope

This specification develops the requirements for utilization of behavioral level simulation for Test System Verification. The objective of this technique is to be able to verify BIT coverage both during the design process and at its conclusion by simulating a high level functional model of the unit under test, its BIT (as designed) as well as models of the fault mechanisms expected in the design. The methodology is covered in section 6.1 and will not be repeated here. References (36) to (39) are key papers on functional modeling and architectural simulation that would be useful background for undertaking a TSV simulation task of this type.

6.5.2 Requirements

The basic requirements for using behavioral level simulation are outlined in the following sections. First, requirements on the unit under test and the BIT to be verified are covered. Then simulation tool requirements are discussed. General and fault modeling requirements and implementation requirements conclude the section.

6.5.2.1 Unit Under Test requirements

The principal requirement on the Unit Under Test is that it is decomposable into building block elements, that functional models of each element can be developed, and that models of the interconnects between building blocks can be developed.

6.5.2.2 BIT requirements

There are no specific requirements on the types of BIT for which behavioral simulation can be used. It must be possible to model the BIT as well as the UUT at the functional level.

6.5.2.3 Simulation Tool Requirements

A simulation tool to be used for behavioral simulation for BIT verification should have the following characteristics.

1. The simulation tool used should be a concurrent fault simulator which

   (a) Operates on behavioral models

   (b) Handles functional faults concurrently

   (c) Runs on at least a 100 MIPS machine

2. The simulator must include the capability to activate the faults necessary to complete each fault simulation pass. Definition of faults to be simulated should be the responsibility of the analyst, but the software must activate faults at the proper time.

3. A modeled processor system with 4K of RAM, 4K of ROM and two I/O ports should be able to execute system code at a minimum effective rate of 200,000 clock cycles per second. Effective rate is defined as (number of clock cycles for a good simulation) times (the number of faults simulated plus one) divided by (total time for complete fault simulation).

4. The modeling tools developed to support the simulator should allow a model of complexity similar to MIL-STD 1750A processor to be modeled and tested in 100 manhours.

6.5.2.4 General modeling requirements

The analyst developing a behavioral model for use in Test System Verification must set up the following items:

1. A decomposition of the Unit Under Test (UUT) into functional blocks.

2. A hierarchical system model structure, including procedures describing the behavior of system components and a model of the interconnects.

3. Behavioral models of system components.

4. Inputs to drive the selected simulation tool.

5. Stimuli to activate modeled faults during simulation.

6. Outputs in the models of individual blocks to indicate how the effects of internal element testability are reflected to the rest of the system.

7. A means of monitoring BIT output to determine if BIT has detected an activated fault.

8. A calculation to compute coverage as a ratio of faults detected to faults simulated.

6.5.2.5 Fault modeling requirements

1. Functional faults need to be defined.

2. Lists must be developed in the model to define which functions will fail due to faults (a list of opcodes that will fail, for instance).

3. Lists must be developed in the model to define which functions will replace the faulted function (for example, this might be a list of different opcodes that should be executed instead of the faulted opcodes).

4. Behavioral models of faults in system components must be developed.

5. Faulting routines need to be included in the lowest level procedures of the hierarchical structure. They interpret the fault lists and change the function as appropriate. For example, the access register and decode opcode procedures are two of the low level procedures that would include the faulting routines for a processor.

6. A fault pointer must be included in the model to indicate the activated fault.

7. A mechanism for activating and simulating each fault must be included in the model.

8. A counter for tracking fault detections must be included in the model. This counter is incremented when the BIT output of the faulted system differs from that of the unfaulted system.

6.5.2.6 Implementation requirements

1. Suitable computer hardware for running the required simulations must be available.

2. Designs for both the unit under test and its BIT must be provided by the system designers.

3. Appropriate engineering staff must be available to develop and analyze the models. The process of developing the system and fault models will require an analyst familiar with the simulators being used. A test analyst must also be available to determine and characterize faults that can occur in the unit under test to select a subset of the total fault set for simulation and to analyze the results of the simulation and their consequences.

7.0 TSCA IMPROVEMENT DETAILS AND SPECIFICATIONS

The evaluation of TSCA techniques (chapter 5) led to the decision to investigate improvements to two techniques. fault insertion and overlapping BIT. This chapter discusses the analyses, designs and specifications resulting from that investigation.

7.1 Overlapping BIT

Overlapping BIT involves using redundant BIT elements or segments to provide self-diagnosability for Test System Condition Assessment. It applies to systems that can be partitioned into disjoint subsets which can be tested individually. To implement overlapping BIT, the Unit Under Test (UUT) and BIT are both partitioned in such a way that each subset of the UUT is tested by at least two BIT segments. With this arrangement, a failure in the unit under test should generate failure indications in two BIT segmants. If only one BIT segment indicated a failure, then it can be concluded that one of the BIT segments has failed and the status of the UUT is unknown. This assumes there is a low probability of multiple failures occurring simultaneously. This method also provides isolation of the failure to a particular UUT or BIT segment.

The investigation included: (1) designing an overlapping BIT control system, including an interpreter circuit to analyse the output from the individual BIT circuits, (2) identifying potential applications, and (3) developing a specification for utilizing overlapping BIT. This section includes the output of these efforts. 7.1.1 defines the overlapping BIT concept and structure, 7.1.2 describes the control system and interpreter circuit, 7.1.3 discusses examples and applications, and 7.1.4 contains the specification for overlapping BIT use.

7.1.1 Overlapping BIT concept and structure

Figure 7-1 illustrates the concept of overlapping BIT. In the figure, sections P1 to P4 partition the UUT into four subsets. Each pair of adjacent partitions is analyzed by a BIT segment, with BIT4 covering the first and last partition. Each BIT segment outputs a single status bit, which is 0 if both covered partitions are fault free and 1 if a fault has been detected. These status bits can be combined into a bit stream which is sufficient to determine when a failure has occurred, and whether the failure is in the BIT or the UUT. Two consecutive 1's in the bit stream indicate a failure in the UUT and isolate that failure to the partition common to the two BIT segments. An isolated 1 indicates BIT failure and its position indicates the faulty BIT segment.

7.1.2 Overlapping BIT Interpreter Circuit

The purpose of this circuit is to monitor a bit stream which is the output of overlapping BIT segments and determine the status of both the system and the individual BIT circuits. This circuit will assume that only single BIT or system failures occur. The design used in this illustration is intended to aid in evaluating the ultimate complexity of the overlapping BIT. It was not intended to generate an optimal circuit at this time. The bit stream is a series of binary digits each of which represents the results of one of the overlapping BIT circuits.

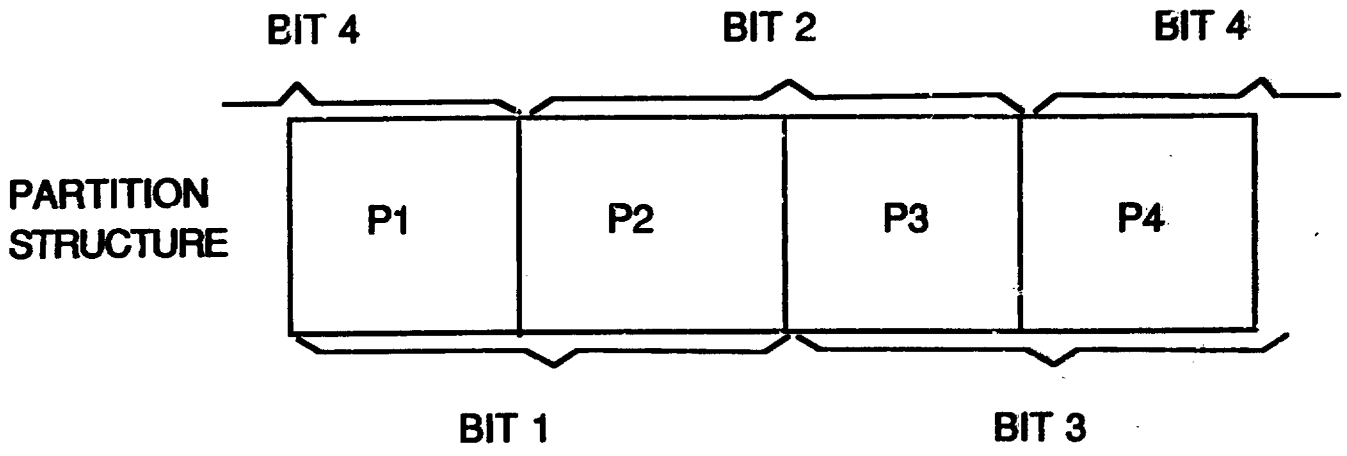


Figure 7-1 Overlapping BIT Structure

The requirements for the overlapping BIT interpreter circuit are as follows:

a. Detect isolated BIT error indications (BIT failures).

b. Detect consecutive pairs of BIT error indications (system failures).

c. Isolate BIT failures.

d. Isolate system failures.

e. Operate on a bit stream of selectable size.

The inputs will be:

a. The overlapping BIT indicator bit stream (serial).

b. The number of bits in the bit stream (preprogrammed).

c. A syncronization clock.

The outputs will be:

a. Indication of a BIT failure.

b. Indication of a system failure

c. Identity of the BIT segment or system partition containing the failure.

The major components of the BIT Interpreter are shown in the block diagram in Figure 7-2. The COUNT SYSTEM block is designed to keep track of which main system block is currently being evaluated by the bit stream. It consists of a simple counter and comparator circuit as shown in Figure 7-3. The BIT EVALUATOR block looks at the bit stream for the occurrence of the failure patterns where adjacent 1's in the bit stream indicate failure in a system partition and an isolated 1 indicates failure in a BIT segment. This circuit is shown in Figure 7-4. The two flip/flops save the initial two bits in the bit stream so that they may be compared to the final two bits since the overlap wraps around the system. The FAILURE DETECTOR of Figure 7-4 is a simple sequential circuit which detects the two patterns. Its state diagram

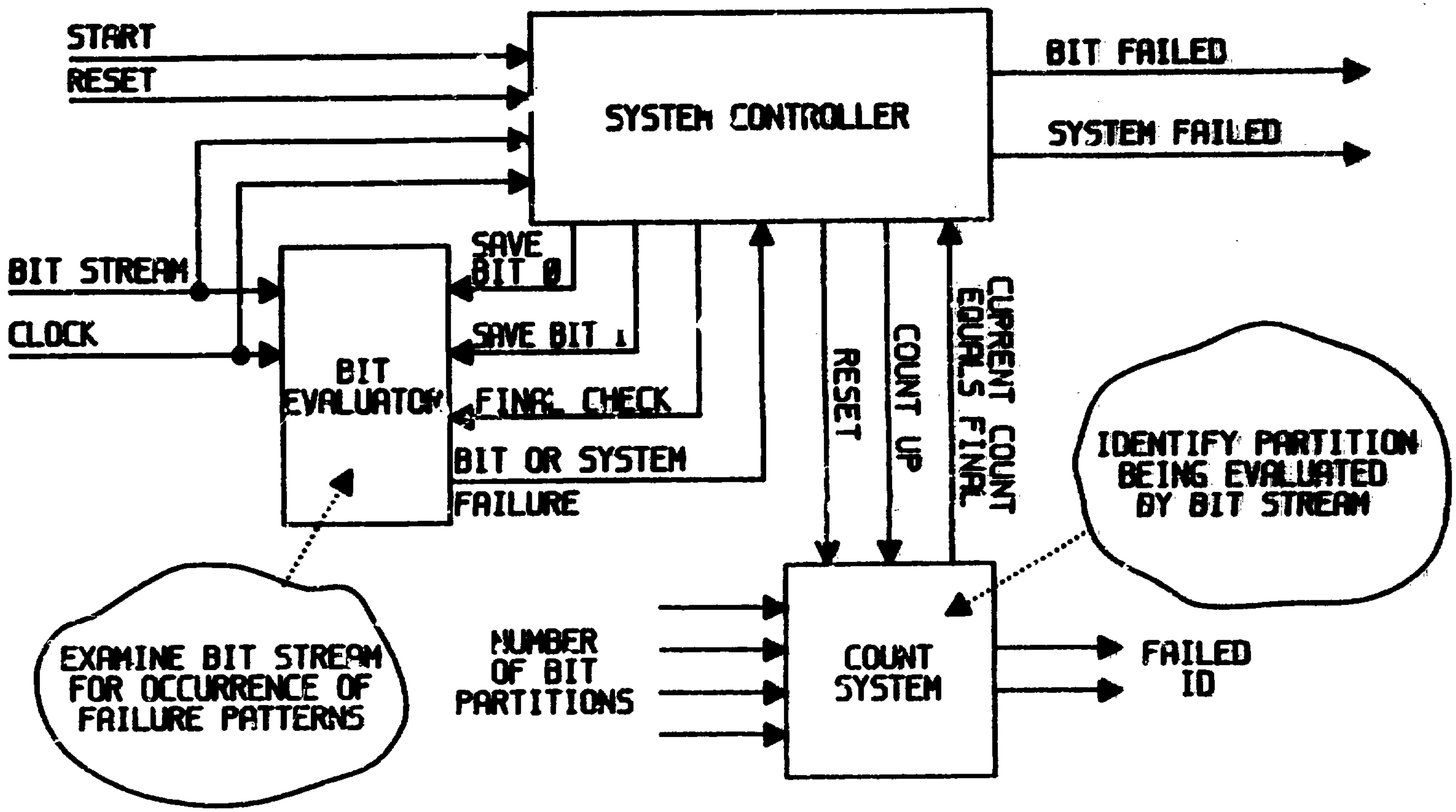


Figure 7-2 Overlapping BIT Interpreter Circuit Block Diagram

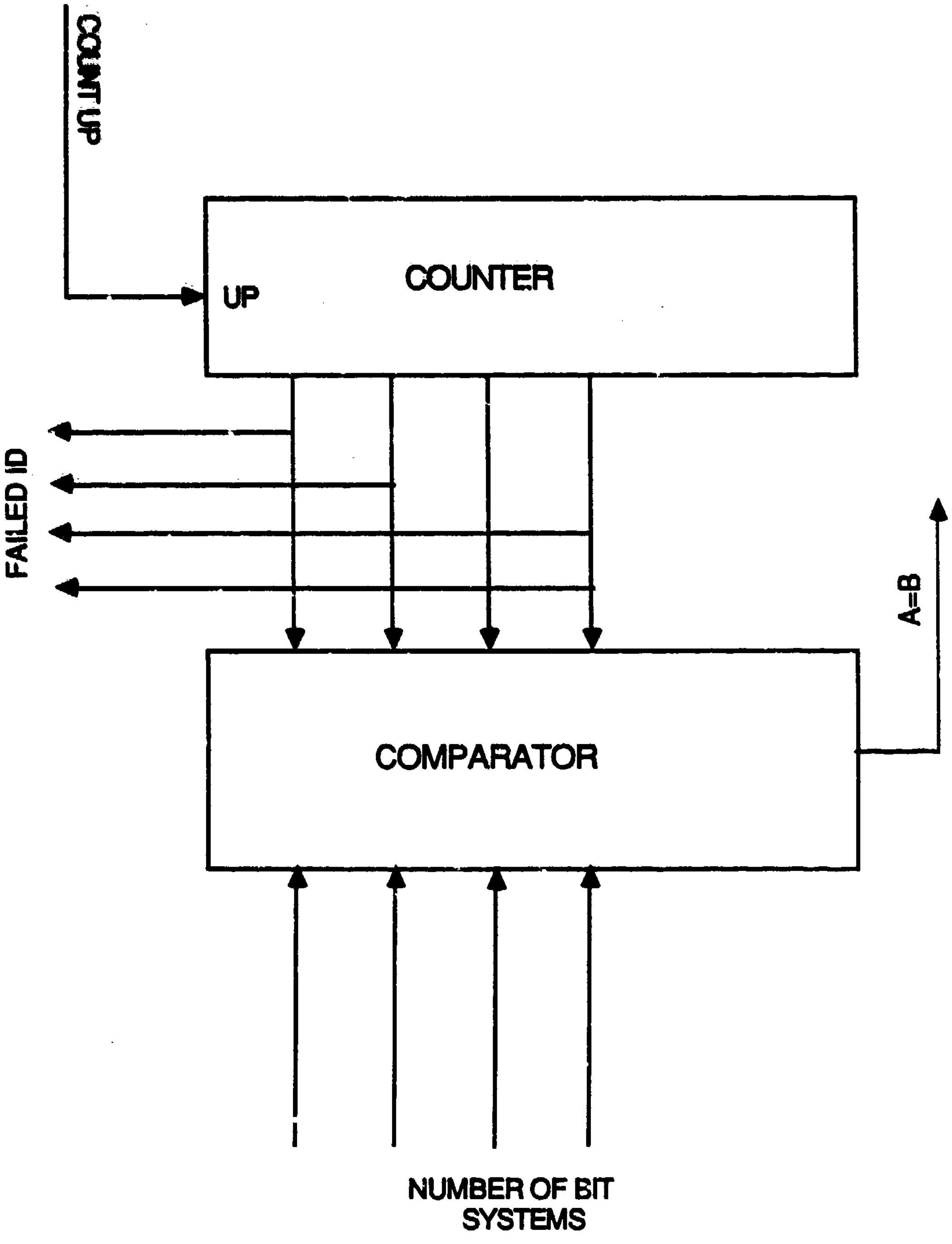


Figure 7-3 Count System

and associated design documentation including the circuit are shown in Figures 7-5, 7-6, 7-7, and 7-8. The FINAL CHECK block is a simple combinational logic circuit which checks for the same patterns in 3 bits as shown in Figure 7-9 .

The control system consists of 3 flip/flops and a 3-to-8 decoder as shown in Figure 7-10. The operation of the control system is described by the state diagram to the state assignment shown in Figure 7-11. The maps which were used to generate the next state decoder system are shown in Figure 7-12.

7.1 . Application of Overlapping BIT

To assess the usefulness and practicality of overlapping BIT, several classes of electronic systems and subsystems were examined to (1) determine the extent of applicability of this concept, and (2) get insight into guidelines for how and when to apply overlapping BIT. Three examples were identified as potential candidates which would not require massive redesign of the system or BIT:

(1) Application of overlapping BIT to a quad-redundant serial data bus, by attaching a comparator to each pair of data lines

(2) Partitioning of a 64 x 8 memory with parity on overlapping segments

(3) Partitioning, in a manner similar to (1), multiple analog lines between a controller and the subsystem it controls.

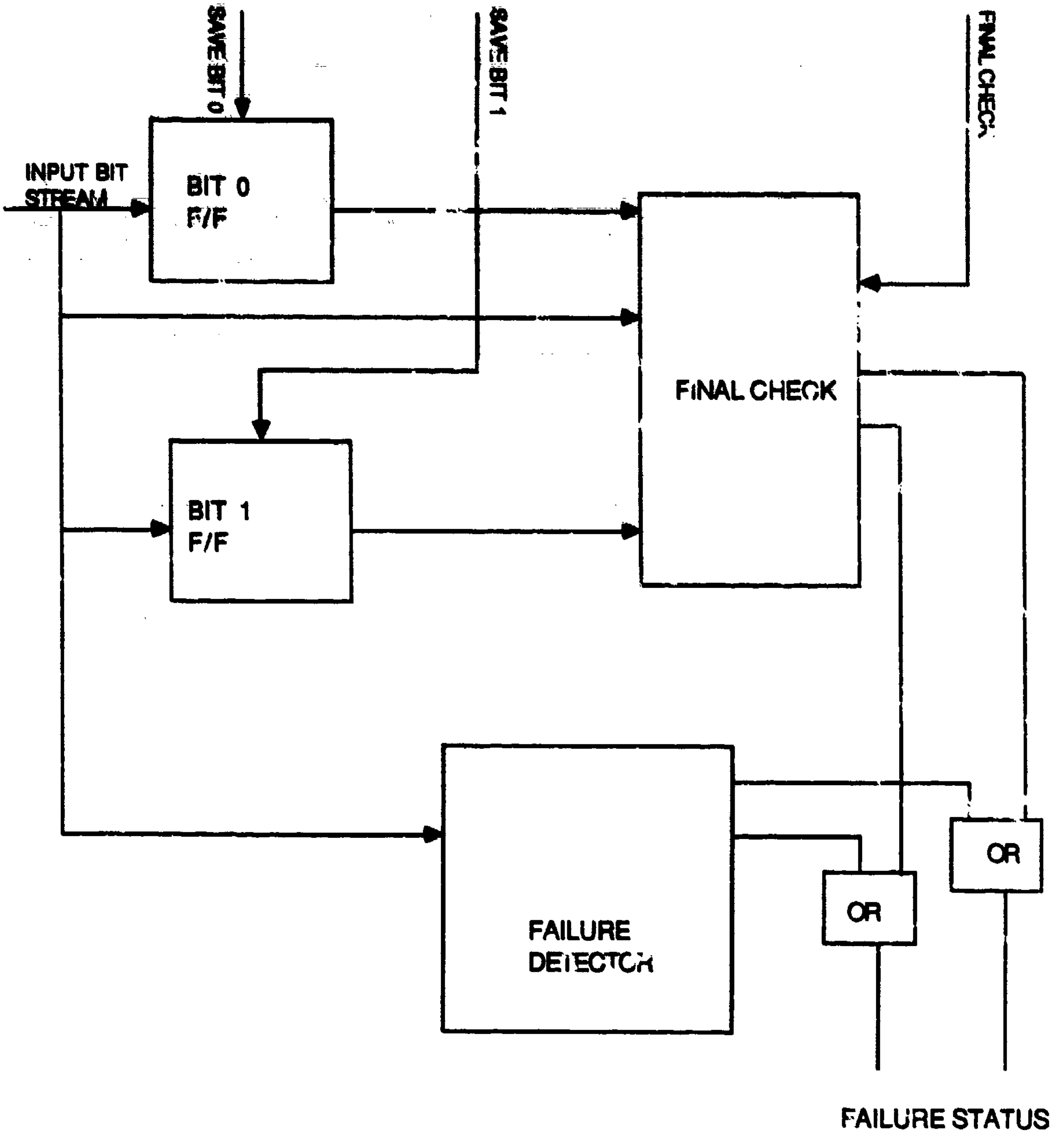
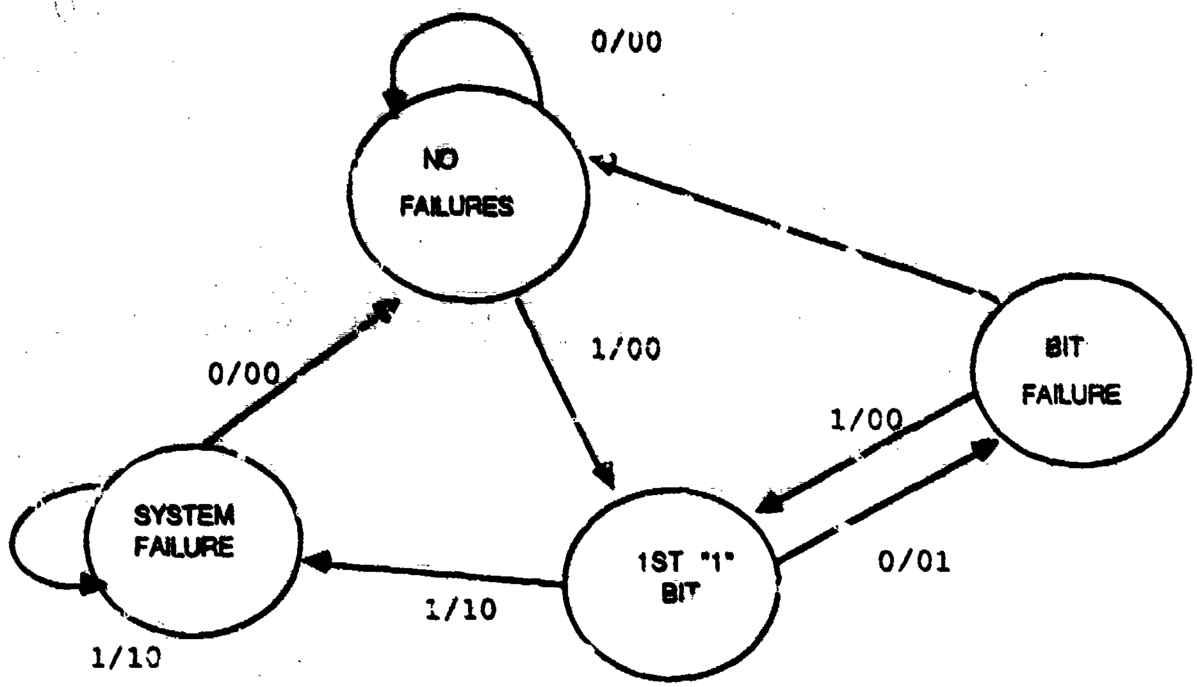


Figure 7-4 BIT Evaluator



Input: Bit Stream

Output: 00 = No Failure
10 = System Failure
01 = BIT Failure

Figure 7-5 State Diagram for the Failure Detector

B \ A	0	1
0	NO FAILURES	SYSTEM FAILURE
1	1ST "1" BIT	BIT FAILURE

State Assignment

OUTPUT:

00 = no failure
10 = system failure
01 = BIT failure

PRESENT STATE			NEXT STATE		OUTPUT	
A	B	IN	A	B	O1	O2
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	1	0	1	0	1

Figure 7-6 State Assignment and State Table

IN \ AB	00	01	11	10
0	0	1	0	0
1	0	1	0	1

A

IN \ AB	00	01	11	10
0	0	1	0	0
1	1	0	1	0

B

Figure 7-7 State Map

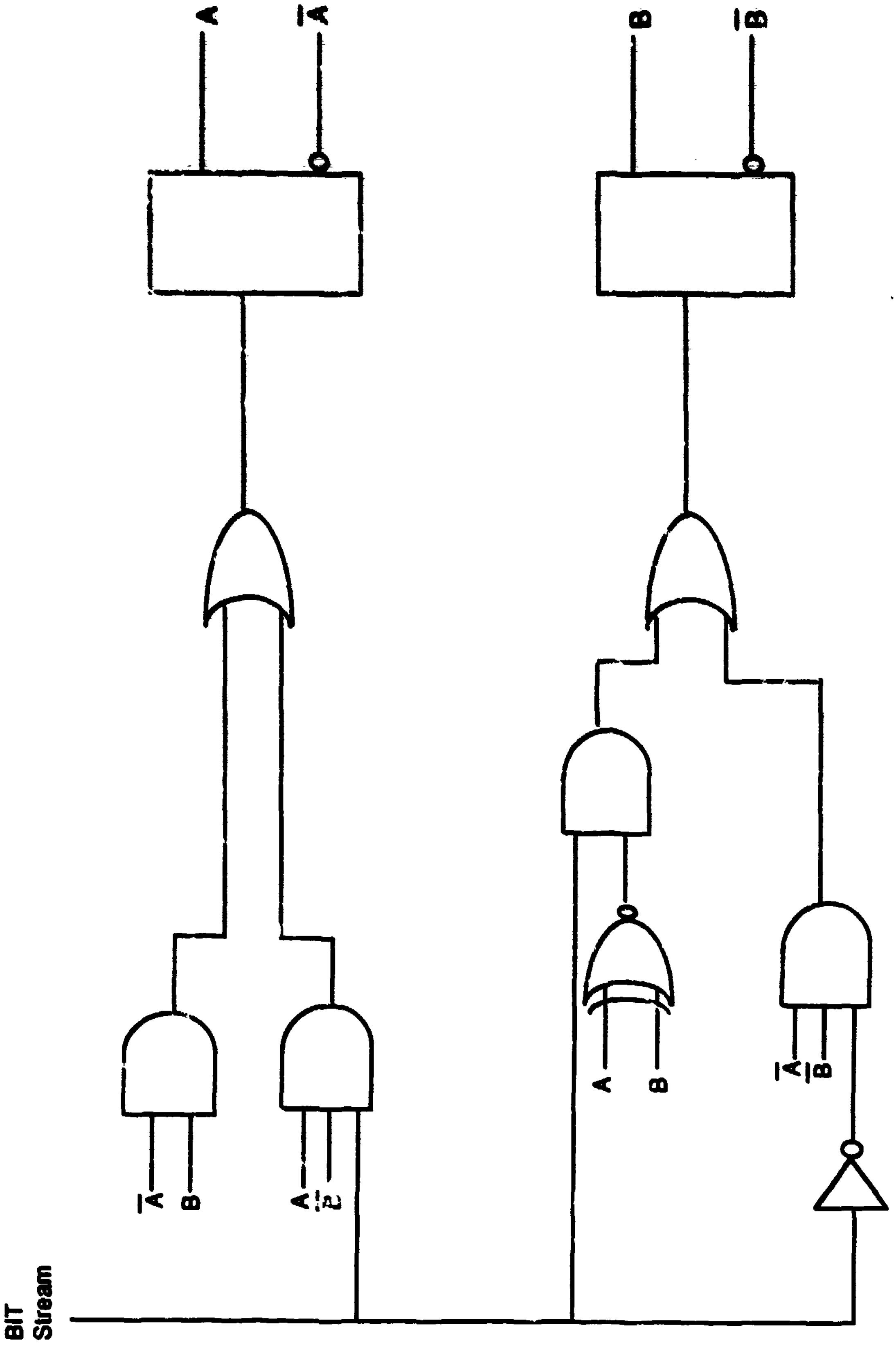
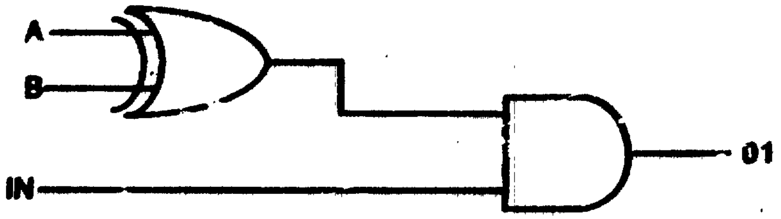


Figure 7-8 Failure Detector State Control Logic Diagram

IN \ AB	00	01	11	10
0	0	0	0	0
1	0	1	0	1

$01 = (A \oplus B)IN$



IN \ AB	00	01	11	10
0	0	1	0	0
1	0	0	1	0

$02 = B(\overline{A \oplus IN})$

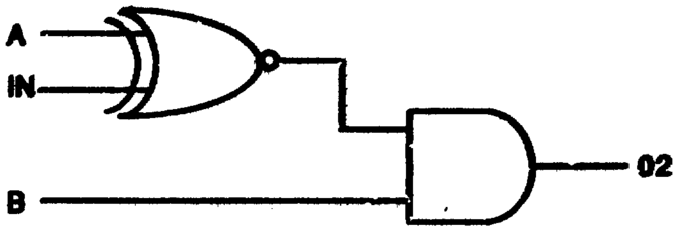


FIGURE 7-9 FINAL CHECK LOGIC

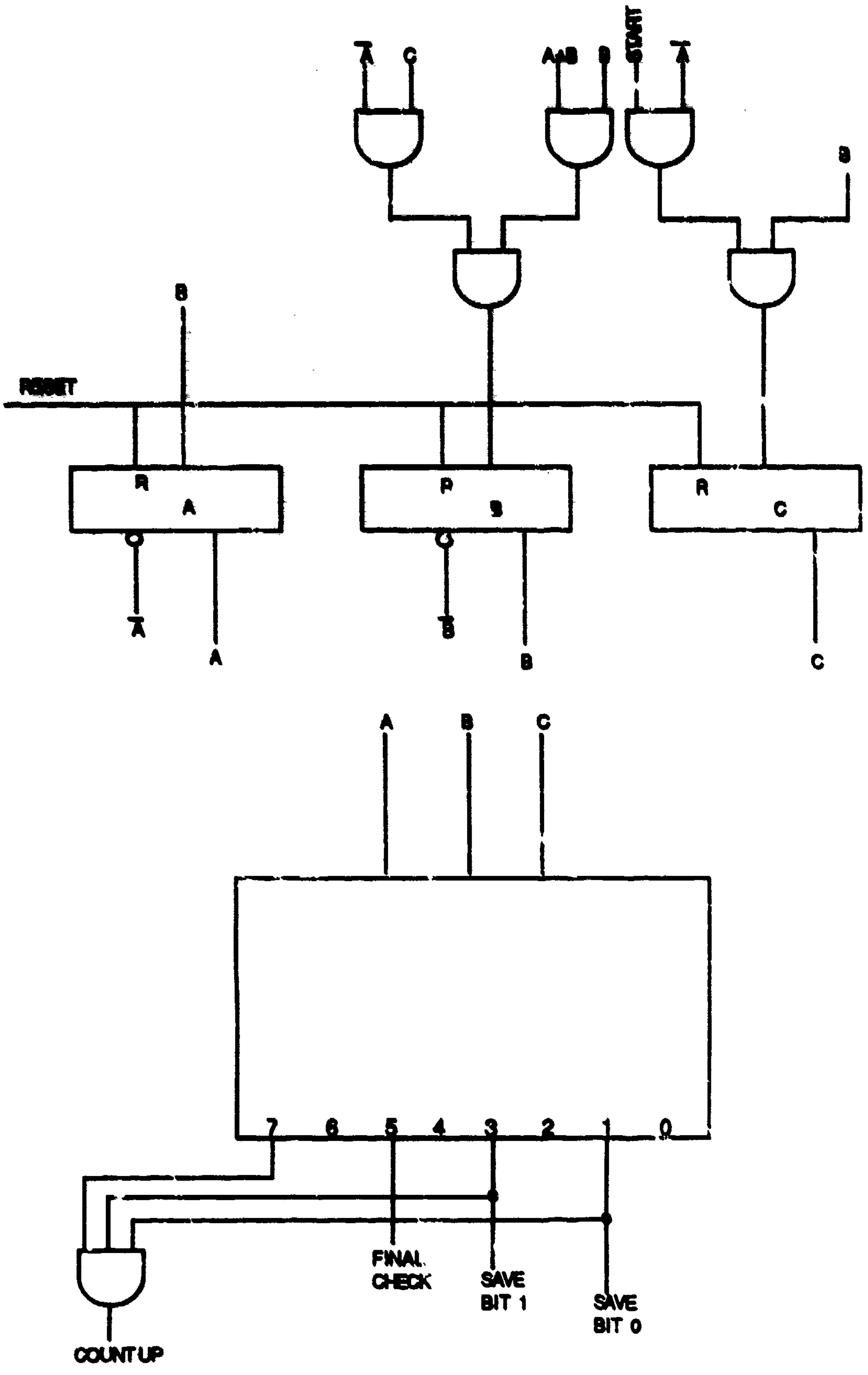
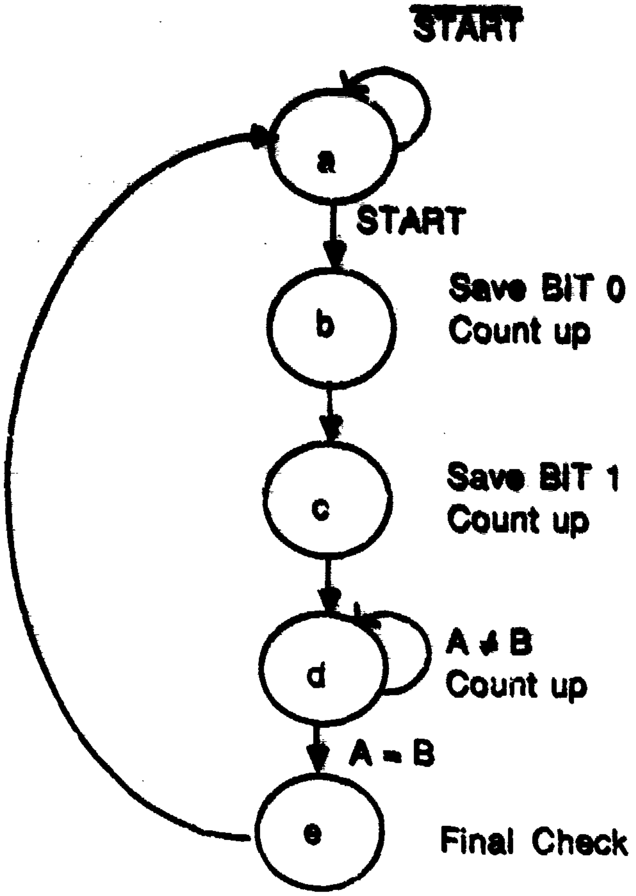


Figure 7-10 Control System Logic Diagram



C \ AB	00	01	11	10
0	a	0	0	0
1	b	c	d	e

State Assignment

Figure 7-11 State Assignment

AB C	00	01	11	10
0	0	0	0	0
1	0	1	1	0

$D_A = B$

AB C	00	01	11	10
0	0	0	0	0
1	1	1	$A \neq B$	0

$D_B = \overline{A}\ C + (A \neq B)\ B$

AB C	00	01	11	10
0	START	0	0	0
1	1	1	1	0

$D_C = (\text{START})\ \overline{A} + B$

Figure 7-12 Maps for Next State Decoder

In addition to these examples, examination of the relationship between Hamming codes and overlapping BIT led to a hybrid concept which provides self-checking for the entire process of creating check bits, storing or transmitting data, and decoding the augmented data. This concept, called overlapping Hamming code, is discussed in section 7.1.3.1.

The guidelines that were sought include:

(1) where to make partitions

(2) how many partitions to use

(3) how to subdivide an entire system into subsystems, each of which would have an independent overlapping BIT structure

(4) types of BIT compatible or incompatible with overlapping BIT

(5) suitable implementation levels for overlapping BIT

Essentially, the only restriction to overlapping BIT application that was found was that the system, or any subsystems to which it is applied, must have a regular structure which is conducive to partitioning, with some form of BIT applicable to the subsets in the partition. Overlapping BIT is applicable to a variety of electronic subsystems but must be implemented separately for each type. For example, in a computer system, BIT must be implemented separately for memory, I/O and within processors, since the BIT types applicable to each subsystem are different. Another conclusion was that unless overlapping BIT is designed as an integral part of BIT, it is suitable only for simple BIT types, such as parity or comparison. Overlapping BIT is generally applicable to digital BIT at both the subsystem and IC levels, but has limited application to analog BIT.

7.1.3.1 Overlapping BIT Applied to a Quad Redundant Serial Data Bus

Figure 7-13 illustrates a redundant four-line serial data bus which uses three extra buses for fault-tolerance. The system is monitored by 4 independent BIT devices, each of which checks a pair of lines. Since the lines, if fault-free, should always carry the same information, the BIT devices could be simple EXOR gates. Note that since the lines monitored by each EXOR gate overlap, a single failure should affect adjacent BIT devices.

The output of the four BIT devices form the test syndrome of the system. For example, the syndrome 0000 indicates a fault-free situation for both the system and the BIT devices. However, if line 2 has failed and the BIT systems are operating correctly, then the syndrome would be 1100 since both BIT 1 and BIT 2 would record the failure on line 2. The four possible system failure conditions then are identified by the four syndromes:

1001	Line 1 failed	(or BIT 1 and BIT 4 failed)
1100	Line 2 failed	(or BIT 1 and BIT 2 failed)
0110	Line 3 failed	(or BIT 2 and BIT 3 failed)
0011	Line 4 failed	(or BIT 3 and BIT 4 failed)

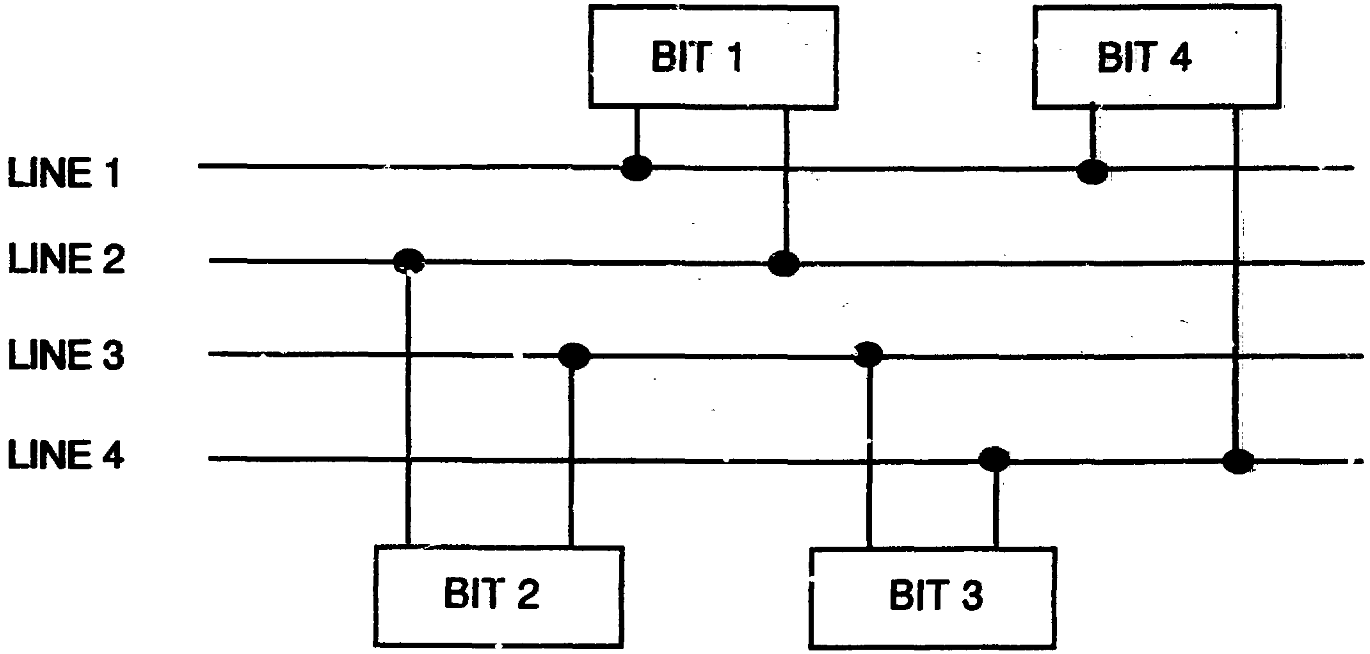


Figure 7-13 Four Line Bus Example

Note that the occurrence of an adjacent pair of "1's" both indicates a system failure and "points" to the failed line. BIT failure conditions are noted by syndromes with isolated "1's". For example, if the system lines are all fault-free but BIT 3 has failed then the syndrome would be 0010. The four possible BIT failure syndromes under the single fault assumption are:

1000	BIT 1 failed
0100	BIT 2 failed
0010	BIT 3 failed
0001	BIT 4 failed

In this case, a single 1 not only flags a BIT failure but also identifies the faulty BIT segment.

7.1.3.2 Overlapping Hamming Code

Hamming codes are commonly used for error detection and correction. They have proven to be a very powerful BIT technique for memory and bus protection. The Hamming code mechanism consists of two main steps:

(1) Extra check bits are added to each word to be stored or transmitted, so that the bits in the augmented word (original word plus check bits) satisfy a set of linear equations called the Hamming equations.

(2) After storage or transmittal, a check is made to see that the augmented word still satisfies the Hamming equations. A discrepancy indicates that an error has occurred, and the data from this check serves to locate the error, be it in the origial word or the check bits.

A discussion of the Hamming equations and the process of creating and decoding them is given in appendix B.

The Hamming code is self-checking in the sense that errors in both the data word and the check bits are detected and can be corrected. However the code itself does not provide a way of checking whether an error has occured during the process of creating the check bits in step 1 or calculating the parity sums for the Hamming equations in step 2. Although the circuitry to perform these operations is simple and not prone to error, its correct operation must be verified in order to assert that the entire Hamming code process is working correctly.

A design which combines the use of overlapping BIT with the application of Hamming codes can provide a completely self-checking mechanism for data storage and transmittal. Instead of applying the Hamming code to the entire word, as is normally done, the word to be stored or transmitted is broken up into subsets in such a way that

(1) a Hamming code can be applied to each subset

(2) the subsets overlap so that each bit belongs to at least two subsets

Figure 7-14 shows a sample decomposition of an 8 bit word into 4 overlapping subsets of 4 bits each. A Hamming code is applied to each subset, not to the entire word. Thus 3 check bits are added to each of the 4 subsets, resulting in a 20 bit augmented word, as illustrated in figure 7-14.

When operating correctly, each individual Hamming checker will identify which bit of the four it checks is bad. Overlapping BIT simply serves to determine whether or not the Hamming checkers are working properly. The detection of a faulty Hamming checker follows the standard procedure for overlapping BIT. For each subset of the original word, the BIT status function outputs a 1 if the Hamming checker has detected an error and a 0 if it has found its subset fault free. A bit stream, consisting of the outputs of these

Original Word: $w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7\ w_8$

		Subword	Checkbits
Subwords and their checkbits:	(1)	$w_1\ w_2\ w_3\ w_4$	$c_{11}\ c_{12}\ c_{13}$
	(2)	$w_3\ w_4\ w_5\ w_6$	$c_{21}\ c_{22}\ c_{23}$
	(3)	$w_5\ w_6\ w_7\ w_8$	$c_{31}\ c_{32}\ c_{33}$
	(4)	$w_7\ w_8\ w_1\ w_2$	$c_{41}\ c_{42}\ c_{43}$

Augmented subwords satisfying Hamming equations:

(1) $c_{11}\ c_{12}\ w_1\ c_{13}\ w_2\ w_3\ w_4$

(2) $c_{21}\ c_{22}\ w_3\ c_{23}\ w_4\ w_5\ w_6$

(3) $c_{31}\ c_{32}\ w_5\ c_{33}\ w_6\ w_7\ w_8$

(4) $c_{41}\ c_{42}\ w_7\ c_{43}\ w_8\ w_1\ w_2$

Final 20-bit augmented word to be stored or transmitted

$w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7\ w_8\ c_{11}\ c_{12}\ c_{13}$

$c_{21}\ c_{22}\ c_{23}\ c_{31}\ c_{32}\ c_{33}\ c_{41}\ c_{42}\ c_{43}$

Figure 7-14: Overlapping Hamming Code applied to an 8-bit word

overlapping BIT segments is examined for the presence of 1's. For example, a BIT output of 0100 would reflect a failure in BIT 2, the Hamming checker for lines 3, 4, 5, and 6. Note that in this case overlapping BIT is not used to (and cannot uniquely) identify the bad line. A bit stream output of 1100 identifies a fault in the overlap of BIT 1 and BIT 2, i.e. in lines 3 or 4. But the Hamming code output must be analyzed to decide which of these lines is bad.

This example illustrates a case in which overlapping BIT can be applied without forcing any major design changes. It also suggests the following questions for future research into overlapping BIT Hamming code hybrids.

1. Is there an optimal partition of the set of lines which minimizes the Hamming code overhead and provides the necessary level of fault isolation?

2. How should the bits be assigned to the individual partitions?

3. Can this approach be generalized to group codes? If it can, will it lead to a relationship between the automorphism group of the overlapping BIT partition and the underlying group associated with the code?

7.1.4 Overlapping BIT Specification

7.1.4.1 Scope

The overlapping BIT concept consists of an organizational structure and analysis method for using multiple BIT systems to provide a self-diagnosability feature to Built-In Test. It allows the BIT to monitor not only the Unit-Under-Test but also its own performance. Overlapping BIT provides a measure of self-detection which increases our confidence in the results of a BIT failure status report. This specification discusses the structure and the analysis of the concept.

7.1.4.2 Overlapping BIT Structure

7.1.4.2.1 Methodology

A standard system with BIT is illustrated in Figure 7-15. It involves a Unit Under Test with a single BIT monitor. The BIT is able to report on the fault status of the system, but there is no way to detect a BIT failure until either the system is pulled and tested or the system affects some other device due to an unreported system failure.

The organization of overlapping BIT requires that the system be partitioned into subsystems and that BIT devices be connected to groups of subsystems. Figure 7-16 gives an example in which the system of Figure 7-15 has been partitioned into 4 subsystems. Four smaller BIT segments, each capable of testing two of the four subsystems, have been used. The key to this organizational structure is that each subsystem is tested by at least 2 different BIT segments.

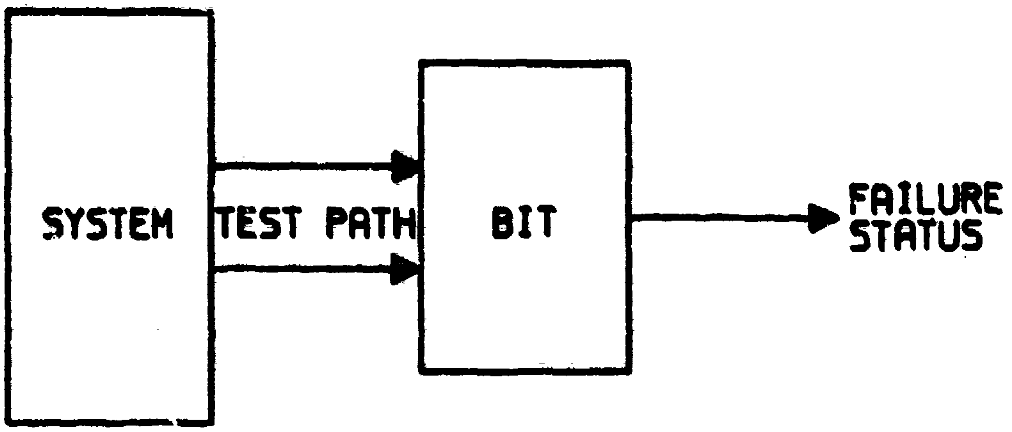


Figure 7-15 Standard System

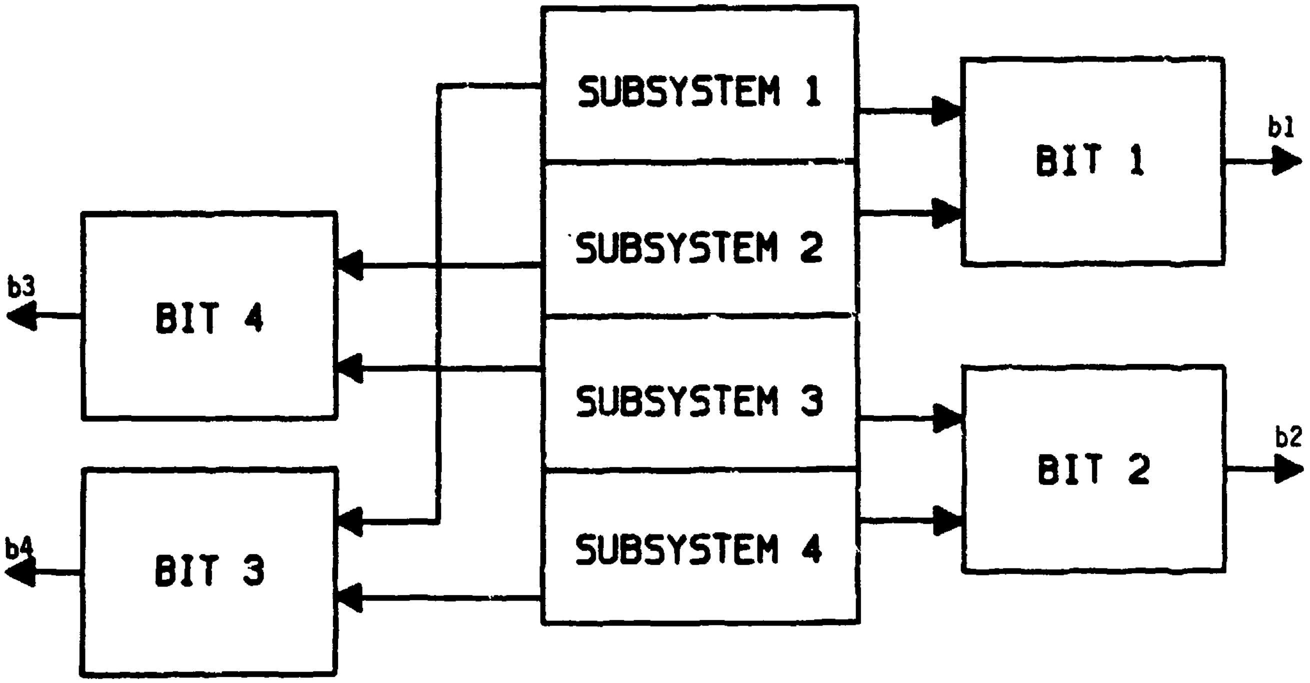


Figure 7-16 Standard Overlapping BIT Structure

7.1.4.2.2 Structural Requirements

The organizational structure of Overlapping BIT when applied to a standard system requires:

1. The original system must be able to be partitioned into disjoint subsystems which can be tested individually.
2. There must be BIT devices capable of testing combinations of the subsystems.
3. It must be possible to arrange the BIT testing pattern so that every subsystem is tested by at least two different BIT devices.

When Overlapping BIT is applied, there can be considerable variation in the BIT processes, algorithms or hardware used to test the system. For example, different types of BIT segments can be used to monitor different groups of subsystems. Also, within a group of subsystems monitored by a single BIT segment, the way in which the BIT segment monitors one subsystem of the group may be different from the way it monitors any other subsystem in the group.

7.1.4.3 Overlapping BIT Analysis

7.1.4.3.1 Methodology

The output of the BIT devices is used to determine the fault status of both the system being monitored and the BIT devices. Each BIT segment must report a binary status signal indicating whether or not it has found a fault in the partitions it is monitoring. It will be assumed here that a BIT output of 1 indicates that BIT has found a failure while a BIT output of 0 indicates no faults have been found. The sequential output of the multiple BIT devices (called the

Overlapping BIT syndrome) is an n-bit binary word where n is the number of BIT devices. The outputs should be arranged in order such that adjacent outputs represent the BIT devices which test the same subsystem. For example, in Figure 7-16, if the output of BIT i is b_i then the outputs should be in the order b_1 b_4 b_2 b_3 since BIT 1 shares subsystem 2 with BIT 4, BIT 4 shares subsystem 3 with BIT 2, BIT 2 shares subsystem 4 with BIT 3, and BIT 3 shares subsystem 1 with BIT 1. The resulting syndrome as reported by the multiple BIT devices can then be easily interpreted to determine the system and BIT status using the following rules:

1. A pair of adjacent 1's indicates a system failure.
2. A single 1 indicates a BIT failure.

The above rules are true only if it is assumed that double faults do not occur or are of low probability when compared to the probability of single faults (see section 7.1.4.4.1).

7.1.4.3.2 Analysis Requirements

In order to analyze the output of a system with Overlapping BIT, the system must have the following attributes:

1. A control subsystem to coordinate the multiple BIT operations must be included in the system.
2. Each BIT segment must be able to issue a signal indicating whether or not it has found a fault in any of the subsystems it is testing.
3. A subsystem to analyze the resulting BIT stream must be included in the system. This subsystem must be capable of (a) accepting as input the overlapping BIT syndrome, (b) determining the presence of BIT indicated failures in the syndrome, (c) identifying whether these represent BIT or subsystem failures and

(d) identifying the failed BIT segment or system partition.

7.1.4.4 Limitations

7.1.4.4.1 Applicability

Use of overlapping BIT to distinguish between BIT and system failures is based on the assumption that there is a low probability of multiple failures occurring simultaneously, either in adjacent BIT segments or in a subsystem and one of the BIT segments testing that subsystem. Using the block diagram in Figure 7-16, Figure 7-17 illustrates how different types of multiple failure conditions can result in generation of bit stream patterns identical to those produced by a subsystem failure occurring with properly operating BIT. As seen in figure 7-18, under some conditions the simultaneous failure of a subsystem and one of the BIT segments which test it could produce the same bit stream as a BIT segment failure.

Therefore, Overlapping BIT should only be used if

(a) the probability that either two adjacent BIT segments or a subsystem and a BIT segment which tests it fail simultaneously is much smaller than the probability of a single subsystem failing

(b) the probability of a subsystem and a BIT segment which tests that subsystem failing simultaneously is much smaller than the probability of a single BIT segment failing

In general, these conditions can be satisfied by choosing the number of partitions sufficiently large.

To translate this criterion into formulas which can be used to verify that it is satisfied, let

Figure 7-17 Cases where multiple failures yield a bit stream identical to single subsystem failure (0110 for P2 failed)

a) Adjacent BIT segments bad (BIT 2 and BIT 3)

BIT Segment/ Subsystem Number	1	2	3	4
BIT Segment Condition	OK	Failed	Failed	OK
Subsystem Condition	OK	OK	OK	OK
Bit Stream	0	1	1	0

b) Subsystem and associated BIT segment bad (BIT 1 and P2)

BIT Segment/ Subsystem Number	1	2	3	4
BIT Segment Condition	Failed	OK	Failed	OK
Subsystem Condition	OK	Failed	OK	OK
Bit Stream	0	1	1	0

Figure 7-18 Case where multiple failures yield a bit stream identical to single BIT segment failure (0100 for BIT 2 failed)

a) Subsystem and associated BIT segment bad (BIT 3 and P3)

BIT Segment/ Subsystem Number	1	2	3	4
BIT Segment Condition	OK	OK	Failed	OK
Subsystem Condition	OK	OK	Failed	OK
Bit Stream	0	1	0	0

n = number of BIT segments = number of subsystems

P_B = probability of having a failure in a single BIT segment

P_s = probability of having a failure in a single subsystem

P_A = probability of having simultaneous failures in 2 adjacent BIT segments

P_{BS} = probability of having simultaneous failures in a subsystem and a BIT segment which tests that subsystem

Then

$$P_A = \left(\frac{\text{\#pairs of adjacent BIT segments}}{\text{\#possible pairs of distinct BIT segments}}\right)\left(\begin{array}{l}\text{probability of 2}\\ \text{BIT segment failures}\end{array}\right)$$

$$= \frac{n}{n(n-1)} (P_B)(P_B) = P_B^2/(n-1)$$

$$P_{BS} = \left(\frac{\text{\#associated BIT segment-subsystem pairs}}{\text{\#possible BIT segment-subsystem pairs}}\right)\left(\begin{array}{l}\text{probability of BIT segment}\\ \text{and subsystem failure}\end{array}\right)$$

$$= \frac{2n}{n^2} P_B P_S = 2 P_B P_S /n$$

IF

P_{B1} = probability of failure in only one BIT segment

P_{S1} = probability of failure in only one subsystem

Then

$$P_{B1} = nP_B (1-P_B)^{n-1}$$

$$P_{S1} = nP_S (1-P_S)^{n-1}$$

The two criteria stated at the beginning of the section are then

(a) $P_A + P_{BS} = k_1 P_{sl}$

(b) $P_{BS} = k_2 P_{Bl}$

where k_1, k_2 are constants much smaller than 1. Substituting the formulas for

P_A, P_{BS}, P_{Bl}, and P_{Sl}, in (a) and (b) yields

$$k_1 = \left(\frac{P_B}{P_S}\right) \left[\frac{nP_B + 2\,(n-1)P_S}{n^2\,(n-1)\,(1-P_S)^{n-1}} \right]$$

$$k_2 = \frac{2P_S}{n^2\,(1-P_B)^{n-1}}$$

Knowing P_B and P_S, these formulas allow the user to determine whether overlapping BIT can be used.

To illustrate, consider a system with an MTBF of 2500 hours (including BIT), in which BIT comprises 25% of total hardware. Suppose the system and BIT are partitioned into 8 subsystems and 8 BIT segments. Table 7-1 gives the failure rates for BIT segment and subsystem hardware, along with the probability of failure in a 1000 hour period, using the formulas $P(t) = 1-e^{-\lambda t}$ with t=1000 and λ = failure rate.

Table 7-1 **Failure Rates And Probabilities For Overlapping Bit Example**

Item	#	Percent of Total Hardware for Each	Failure Rate (Failures per hour)	Probability of failure in 1000 hours
BIT Segment	8	3.125	$12.5\text{x}10^{-6}$	$P_B = .0124$
Subsystem	8	9.375	$37.5\text{x}10^{-6}$	$P_S = .0368$
Total	1	100	$400\text{x}10^{-6}$	

Then

$$k_1 = \left(\frac{.0124}{.0368}\right)\left[\frac{(8)(.0124)+(2)(7)(.0368)}{(64)(7)(1-.0368)^{7}}\right]$$

$$= 6.008 \times 10^{-4}$$

and

$$k_2 = \frac{(2)\ (.0368)}{(64)(1-.0124)^{7}}$$

$$= 1.255 \times 10^{-3}$$

both of which are extremely small numbers. This verifies, for this case, the premise on which the overlapping BIT analysis is based.

7.1.4.4.2 Effectiveness

Even if Overlapping BIT can be applied to a particular system, it will require additional circuitry in the form of extra BIT devices and an Overlapping BIT control and analysis subsystem. The effect of the extra circuits on the overall failure rates and the expected probability of a BIT failure must be examined in order to determine whether Overlapping BIT will indeed improve the performance of the system as a whole.

7.2 Fault Insertion for TSCA

Fault Insertion can be used as a TSCA method to determine the status of a BIT system in the field. Special circuits, called Fault Insertion Devices (FID), are placed at predetermined locations within the operating unit. In its normal operating mode, the FIDs do not affect the circuit or the BIT. However, when placed in a BIT test mode, the FIDs will intercept the normal signal and replace it with an error signal. The BIT runs a test on the circuit with the inserted fault. If the BIT does not detect the presence of the fault, then a BIT failure has been detected. This method can be used in any circuit which meets the following two requirements:

1. There must be space available on the chip or PCB for the FID.

2. The delays introduced by the FID along its signal path must not be critical to system performance. FID's should be on the order of one or two gates to achieve this.

The method does not directly affect the BIT circuit or interact directly with any BIT signals, hence it can work with any BIT device without restrictions. System failures do not affect the performance of a fault insertion device. The fault insertion device disconnects the system input to the BIT when testing the BIT. Therefore a system failure will not influence the results of the TSCA test.

7.2.1 Fault Insertion Implementation

The basic fault insertion mechanism as developed in the literature uses multiplexers (MUXs) to insert incorrect signals for the BIT to catch. The multiplexers in the unit under test normally pass correct signals. However, when the BIT system is under test, an incorrect stuck at value is selected and inserted into the rest of the circuit. If the BIT detects the presence of the incorrect value generated by the multiplexer, then the BIT is assumed to be operating correctly. Otherwise it is clearly in a failed mode. The following is an initial top level description of a system which uses multiplexers. In the next section a new approach to fault insertion is developed which does not depend on the use of a MUX.

Figure 7-19 shows a block diagram of the complete system, including the Unit Under Test, BIT and the control system. The control system will initiate a fault insertion test of the BIT in response to an external signal (START). This signal will instruct a subset of the fault insertion MUXs in the unit under test to pass an incorrect signal value. The control system will also instruct the BIT system to run its test on the operating circuit (RUN). Finally, the control system compares the BIT output to the expected output and reports on the BIT status to the outside world (BIT STATUS). A state diagram for the control system is shown in Figure 7-20. The control system will sit in state "a" until the signal START goes high. At that time it will move into state "b" and initialize itself for the BIT testing sequence by clearing the current BIT STATUS output flip/flop and setting an internal fault counter. The internal fault counter contains the number of faults that will be introduced into the unit under test during the BIT test cycle. After the initilization state the control system goes into state "c" where three events occur:

1. A fault condition signal is sent to the MUX's in the unit under test to produce the required fault conditions;

2. The fault counter is decremented;

3. The BIT system is told to initiate a test operation.

The controller then enters state "d" where it waits for the BIT system to complete its test run and notify the controller with the BIT RDY signal. At that time the controller enters state "e" where it checkn the BIT Output. If the BIT Output indicates that the BIT system detected the inserted fault

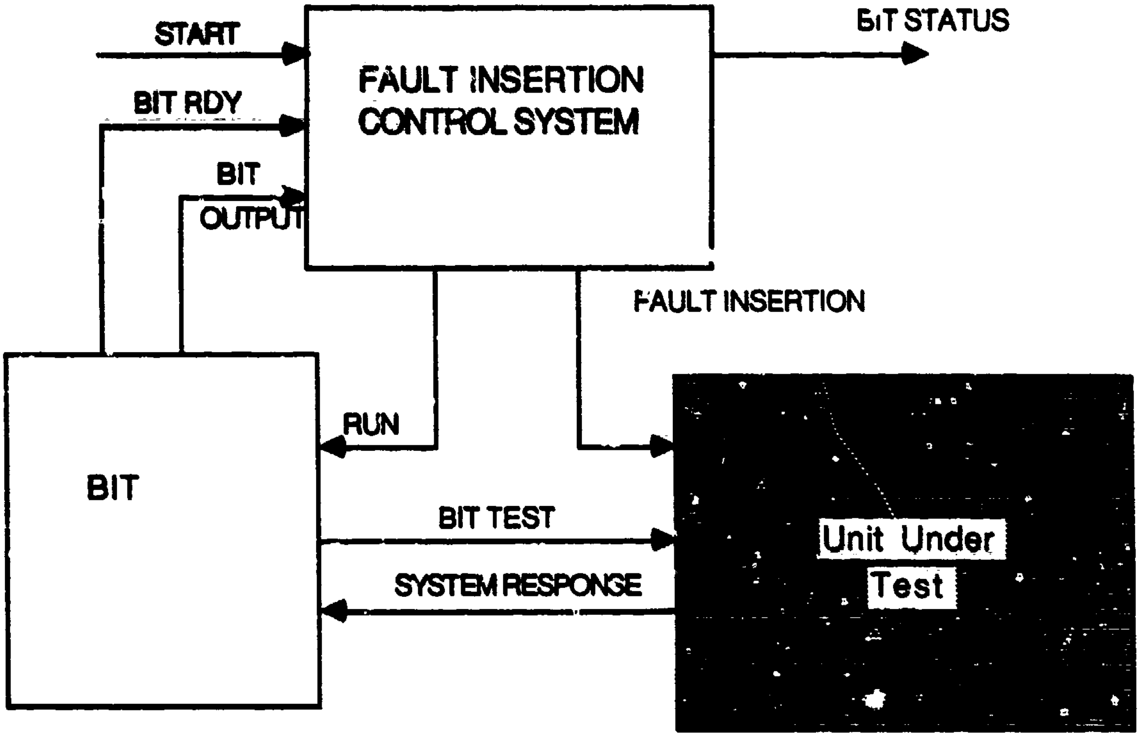


Figure 7-19 Fault Insertion System Block Diagram

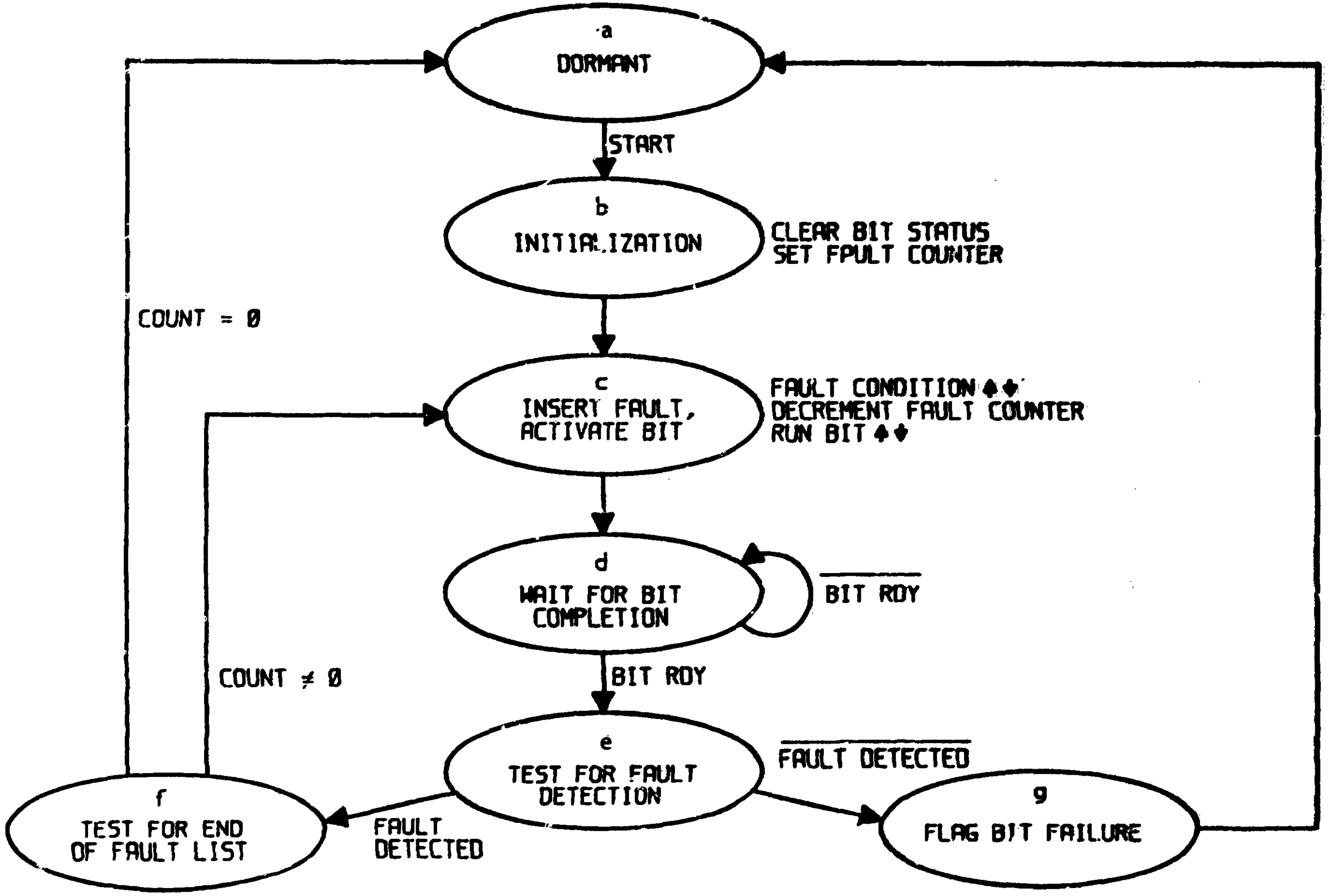


Figure 7-20 Fault Insertion Control System State Diagram

condition, then the system enters state "f" where it checks the fault counter. If the fault counter contains a 0 then all the faults were inserted and the BIT system detected them all so the controller does not change the BIT STATUS flip/flop and returns to state "a" to wait for another START signal. If the fault counter does not contain a 0 then the controller will return to state "c" to insert another fault. The other option for leaving state "e" occurs when the BIT system does not detect the inserted fault. In that case the controller moves to state "g" where it sets the BIT STATUS flip/flop to 1 to signal a BIT failure and returns to state "a" to wait for another START signal.

The control system involves a minimal amount of hardware since it only has 7 states in its state transition diagram. These require only 3 flip/flops to implement. Hence it does not add much to the overhead of an operating system.

7.2.2 Fault Insertion Design Issues

Fault insertion is a simple and effective method for evaluating the status of an operational BIT system. The two major design issues associated with the circuit are:

1. How many faults should be inserted?

2. Where should the faults be inserted?

The answer to the first question requires an analysis of both the BIT approach used and the monitored system in order to determine the necessary degree of fault coverage. The second question also depends on the monitored system. One possible approach to both questions would be a testability-type analysis using the PCB testability measure developed by BAC for RADC (42). This section examines the possibility that such a testability measure may present the solution of the fault insertion problem and suggests potential areas for future research.

A monitored circuit with a high degree of testability may require only a few injected faults in order to verify BIT performance. This is the case because just a few faults could represent a major portion of the possible faults in an easily testable circuit. On the other hand, a monitored circuit with a low degree of testability may require a large number of injected faults in order to cover the wide range of possible system failures. As a result, a simple guideline may be to inject n faults into a monitored system with a first approximation of n given by:

$$n = 10*(T)^{-1}$$

where T is the testability measure for the monitored system. In this case, a monitored system with a testability of 1.0 would require 10 injected faults while one with a testability of .3 would require 33 injected faults.

Once the number of faults that should be injected has been estimated, the next step is to decide where to distribute them in the circuit. This question is a difficult one to answer and will require extensive analysis at some future date. One possible approach would be to extend the testability analysis which was used to determine the number of faults to inject into the circuit. For example, the circuit could be decomposed into 1/T parts where T is the testability of the circuit. Ten faults would be injected into each part to give the required number of injected faults.

7.2.3 Fault Insertion Output Processing

The output signals from a fault insertion TSCA system could be monitored in several ways. First, in an operator controlled system, they could be sent to the system control panel. In this case, the START button could be a panel switch and the BIT STATUS signal could be connected to a light. The user would simply engage the switch and monitor the light. If the light goes on, then the BIT system should be repaired or replaced. This approach would require some space on the control panel for the button and the status light and would involve the human operator directly in the BIT status determination action. Second, the signals could be sent to a separate maintenance panel. This would operate just as in the previous case, except that it would not require control panel space. In this case fault insertion is used only during routine maintenance. This approach requires the system to be off-line or at least not in a duty-cycle before the test condition can be run. Third, the system could operate automatically at set times with the results of each BIT evaluation run saved in a log file. The log file structure would consist of the time of the fault insertion, the type of fault inserted, and the BIT result. The user would examine the log file during a maintenance action to determine the BIT history. This approach allows for an evaluation of BIT performance over a period of time rather then making a TSCA determination based on only one test evaluation. Fourth, the result could trigger an automatic reconfiguration in a system with a redundant BIT architecture. The BIT FAILED signal could be used to switch out the bad BIT and switch in a new BIT system. Fifth, part of the fault insertion control system and output signals could be on a separate device that plugs into an operational system for maintenance action. In this case, only a plug needs to be available on the control or maintenance panel.

These output mechanisms all represent a small percentage of total system cost, on the order of something less than 10%. The cost impact of each alternative is indicated in Table 7-2.

APPROACH	COST	COMMENT
Control Panel Switch/ Signal	Low	Requires control panel space
Maintenance Panel Switch/Signal	Low	Saves control panel space
Plug-in Device	Medium	Requires the design of a separate device. However if used in an environment with a number of systems, it could result in the lowest cost
Log File Structure	High	Requires extra hardware to start the BIT and record BIT results
Automatic Repair	High	Requires extra hardware to start and complete a self-repair operation

TABLE 7-2: RELATIVE COST OF OUTPUT MECHANISMS

These costs represent fixed costs which for most cases is independent of the size of the unit-under-test. As a result, the relative cost of the output mechanism decreases as system size increases.

The choice of output mechanism depends on the nature of the system and its mission. Some of the differences are indicated below for five applications of interest.

1. Ground Control Systems

These systems could use any of the suggested approaches depending on the nature of the system. For such systems, a programmed log file approach may prove to be the most useful. In an unmanned remote radar station, for example, the log file for the entire period between maintenance actions could be easily and quickly examined at the start of a routine maintenance action. The log file can also be used to log intermittent failures, and for fault tolerant systems, to log reconfiguration actions. These can be evaluated during maintenance actions and can be thoroughly analyzed between routine maintenance actions.

2. Aircraft

A pilot will probably not be asked to engage the BIT fault insertion system during flight and control panel space is not likely to be available. However, a control panel signal can be used if space is available and it is desirable to engage the fault insertion mechanism when the unit-under-test is not in a duty cycle. On the ground, either a maintenance panel signal or plug signal mechanism can be used for fault insertion for aircraft avionics. The trade-off between the approaches must be made on a case-by-case basis.

3. Missile Systems

A plug signal mechanism may be the best approach to fault insertion for missile systems since fault insertion will not be engaged during the mission time of these systems. It will be used only on the ground or during active carry. The plug mechanism allows the missile to be plugged into either an external tester or a BIT evaluation system resident on the carrier. In the latter case, the evaluation system could produce a log file or use a panel indicator system which will inform the user on the status of the BIT in all the missile systems on-board.

4. Unmanned Spacecraft

These systems are similar to missile systems. A log file periodically transmitted to the ground may be the best choice.

5. Manned Spacecraft

Depending on the nature of the equipment, it may be possible and even necessary to check out the BIT system during the mission. In this case, a control panel mechanism is preferred, if panel space is available. If not, then a maintenance panel mechanism would be a poorer second choice. Both approaches require less free standing equipment and are always available for check out procedures. However, they both require crew time which may not be available. In such a case, the log file approach which minimizes crew time may be the method of choice. These issues must be traded on a case-by-case basis.

7.2.4 A New Fault Insertion Device

The standard implementation of fault insertion for TSCA uses a multiplexer to introduce faults into an operating circuit as shown in section 7.2.1. This approach allows evaluation of BIT performance under stuck-at and bridging fault conditions. However, it involves the use of a large number of multiplexers. The following approach is suggested as a simpler fault insertion implementation technique. It involves the use of a new fault insertion device (FID) as shown in Figure 7-21.

The device is based on the use of tri-state gates. It has three inputs:

1. Standard Signal: The normal operating circuit signal.

2. Fault Value: The value that is passed into the normal operating circuit in place of the standard signal when the BIT test is being performed.

3. Test Status: Determines if the output results from use of a normal operating circuit signal (test status = 0) or the fault value (test status = 1).

This circuit has two advantages over the multiplexer approach:

1. Fewer gates: The FID requires only 2 inverters and 2 tri-state devices as compared to a standard MUX which contains the equivalent of several NAND gates.

2. Speed: The FID has shorter delays than a multiplexer so it has less impact on the timing of the circuit.

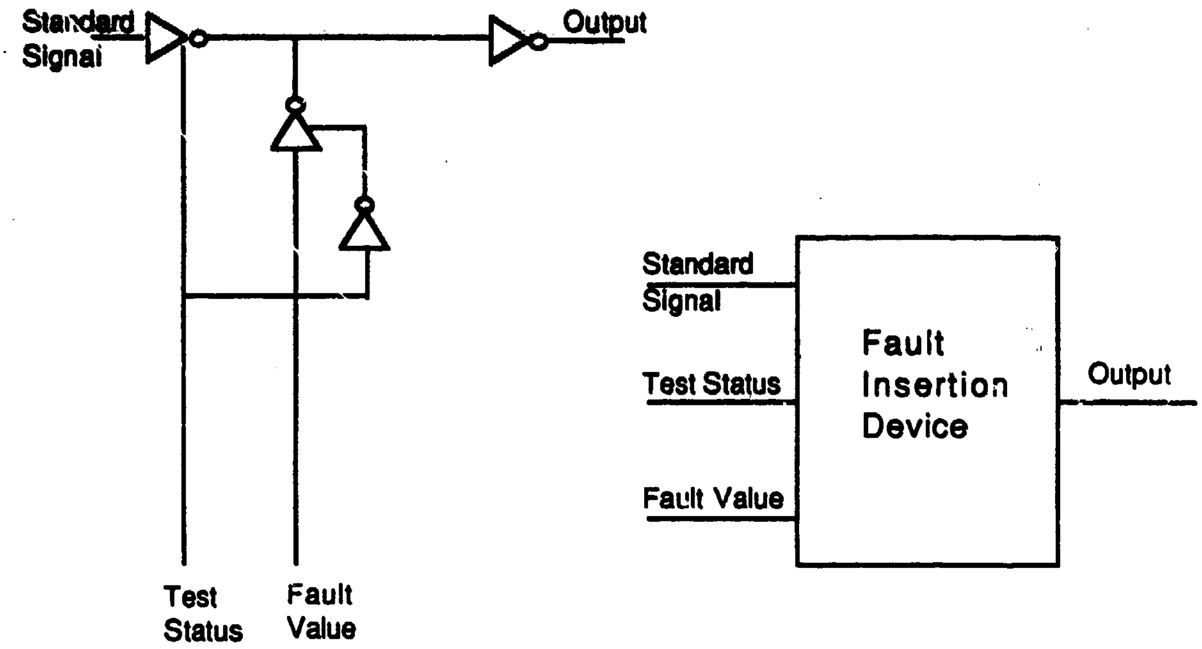


Figure 7-21 Circuit For a Fault Insertion Device

The FID can be used to implement a variety of fault conditions as shown in Figure 7-22.

The intermittent fault control signal (fault control) is 0 normally and is set to 1 when a fault is to be injected. When fault control is at 1 then every clock pulse will cause the FID to go into the fault inject mode. The length and frequency of the intermittent are controlled by the clock pulse, which could be a signal derived from the system clock or could be generated, perhaps randomly, by somethinq other than the system clock.

7.3 TSCA Guidelines

This section looks at TSCA from a system standpoint in order to specify how to approach a large design task. It may be necessary and advisable to use more than one TSCA technique on different segments of the same system. The overall guidelines that could be considered for selecting the best TSCA approach are outlined. Some of the remaining problems in TSCA implementation are also discussed.

There are six considerations which guide the selection of a TSCA technioue for a segment of a given system:

1. Applicability of Overlapping BIT
2. The need for continuous vs periodic monitoring
3. The performance benefits of Overlapping BIT
4. The trade-off between BIT complexity and system complexity
5. The existence of critical timing conditions
6. The ease of overall TSCA system design

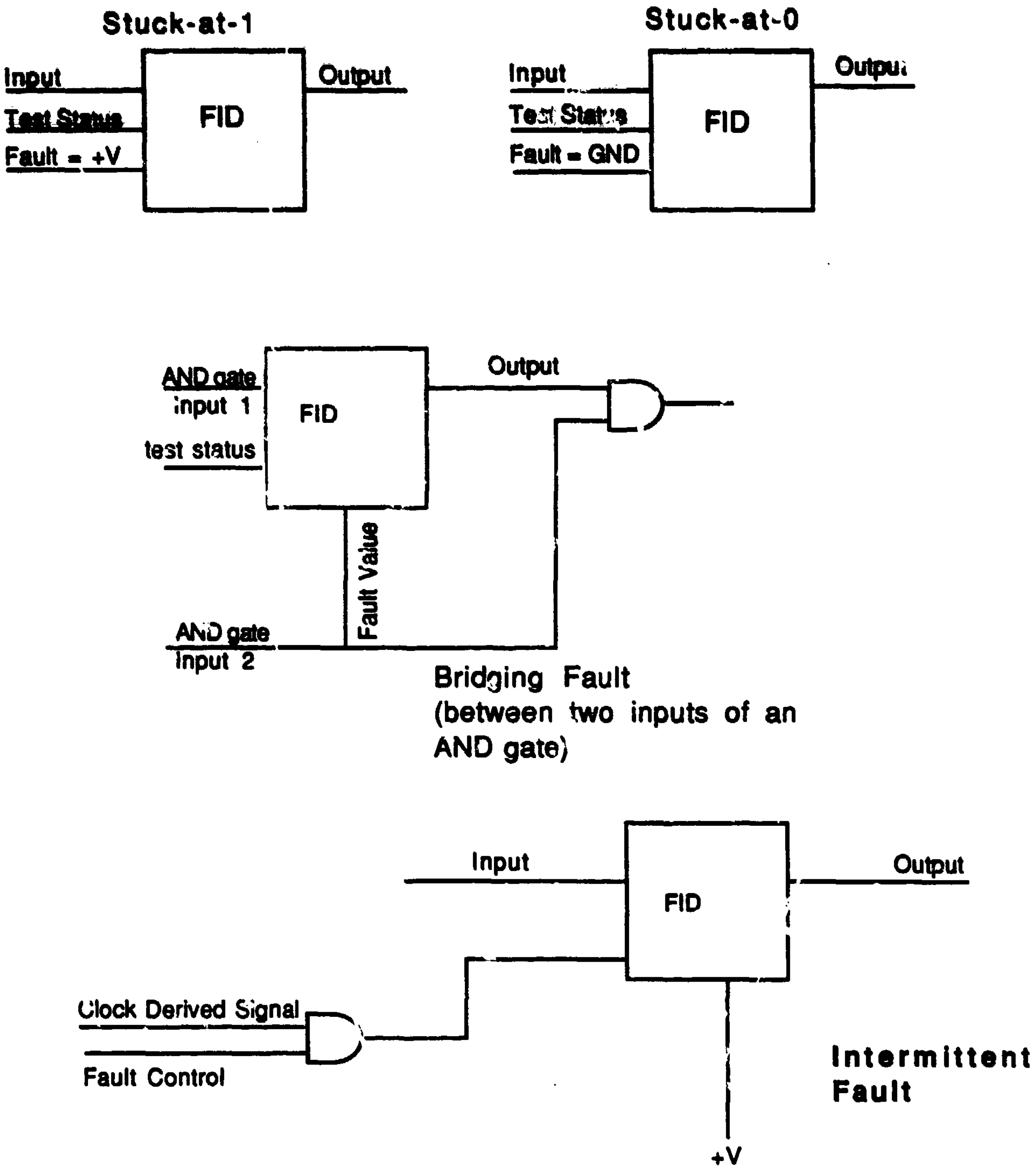


Figure 7-22 Sample Uses of Fault Devices

Each of these will be considered in terms of their impact on the system level design.

1. Applicability of Overlapping BIT

   There are two conditions which must be met before overlapping BIT can be used:

   a) the unit-under-test must be partitioned into non-overlapping subcircuits.

   b) BIT which can monitor more than one partition at a time must be developed.

2. Continuous vs periodic monitoring

   Overlapping BIT provides a continuous on-line check on the performance of the BIT. Whenever a BIT failure occurs, it is immediately detected and the user is notified. Fault insertion requires that actual fault conditions be inserted into the circuit. It can only be used off-line or activated at periodic intervals so as not to interfere with the operational cycle of the system.

3. Performance Benefits of Overlapping BIT

   If the BIT is operating correctly, then a side advantage of Overlapping BIT is that it will isolate faults. The two BIT devices which indicate a system failure will also identify the subsystem which is the source of the failure. Fault Insertion does nothing more than determine the go/no-go status of the BIT device.

4. BIT complexity vs System complexity

Overlapping BIT does not require any substantial changes in the monitored system. It does require a particular BIT architecture. The BIT must be divided into a set of overlapping BIT segments and a control system which evaluates the different BIT outputs must be added to the overall BIT overhead. As a result, BIT complexity can be increased by up to 100% using overlapping BIT while the system is unaffected. The following simple analysis explains the nature of this increase in BIT complexity. If the unit-under-test is partitioned into n sections, then each new overlapping BIT segment must be capable of testing 2 sections. Suppose c is a measure of the complexity of the system, such as the number of logic gates. If the complexity is a linear function, that would imply that the overlapping BIT segments have a complexity of 2c/n. Since there are n overlapping BIT segments required, the complexity would be 2c or twice the original non-overlapping BIT complexity.

For fault insertion, the opposite is true. The BIT design is not changed by the fact that fault insertion is used within the monitored system. The monitored system, however, must have the fault insertion devices added to the design. This could result in as much as a 10% increase in overall complexity. Judged only on this criteria, fault insertion seems to be the best approach. However, all the criteria must be taken into account for any specific application.

5. **Critical Timing**

Since overlapping BIT does not require any monitored system design changes, it will not affect the timing of the system itself. However, the fault insertion approach will add two extra gate delays along any path in which the fault insertion device has been placed. If the added delays cannot be tolerated by the circuit, then overlapping BIT would be the TSCA method of choice.

6. **Ease of Design**

Designing the overall fault insertion control structure is the only substantial design task added by the fault insertion technique. The other design issues associated with fault insertion involve selecting the site for the fault insertion devices, how many fault insertion devices are used, and how to connect the fault insertion devices to the unit-under-test. These are all comparatively minor issues.

For overlapping BIT, the partitioning of the unit-under-test into smaller subcircuits and the development of BIT segments which can overlap in their coverage on the partitions may require several cycles through the BIT design process. The result could be a more complicated design task.

No overall guidelines can be developed regarding ease of design and the issues raised in this discussion must be traded on an individual case-by-case basis.

Issues which must be addressed by the system designer for specific BIT applications include:

1) how a circuit is best partitioned for overlapping BIT implementation;

2) how many faults should be inserted to achieve the desired coverage in the fault insertion approach;

3) where the faults should be inserted.

8.0 Conclusions

The study indicated that more attention is being given to incorporation of testability, and to BIT design in particular. This will continue in the near future. There will be no radical changes in integrated circuit technology during this time, only continued miniaturization and increased performance. Capabilities of VLSI CAE tools will continue to improve. These developments and the increased emphasis on BIT will encourage the evolution of new BIT concepts such as smart BIT and will lead to the emergence of BIT chip sets. The need for more effective and economical BIT verification techniques will continue to increase in the future.

The survey identified quite a few BIT verification techniques, along with numerous variations of each technique. These are techniques that are in use, under development or proposed. The result of the evaluation was that three techniques, behavioral simulation for TSV and overlapping BIT and fault insertion for TSCA, were selected as the most promising candidates for improvement. After further research, these techniques continued to show good promise for developing into effective verification capabilities. The TSCA techniques, overlapping BIT and fault insertion, are applicable in their current form now. For them, details and specifications for their use were developed. Since they apply in different situations, guidelines for their use were included. Investigation into the use of behavioral level simulation for TSV led to the conclusion that the technique is viable, but that improvements in simulators, fault modeling and increased availabilty of powerful computers will be necessary before it can be practical on a fairly universal basis. Most of the necessary improvements will evolve naturally over the next 3-5 years, but some stimulus is necessary in the areas of concurrent fault simulation and the relationship of functional fault models to physical faults.

Agreement on standardization of methodologies was found to be at least as necessary as finding better techniques. Standardization would eliminate the use of the ad hoc methods frequently used now and would make it unnecessary to spend time selecting a method for each new program. The results of the application of BIT verification methods would also be more easily understood and accepted by users. Overcoming the problem of user acceptance would be in itself a significant achievement.

REFERENCES

1. V. D. Agrawal, S. K. Jain and D. M. Singer, A CAD System for Design for Testability, VLSI Design, October 1984.

2. RADC-TR-83-257, Computer-Aided Testability Design Analysis, December 1983.

3. RADC-TR-84-203, Artificial Intelligence Applications to Testability.

*4. RADC-TR-85-148, Smart BIT, Aug 85. (Distribution limited to USGO agencies and their contractors, critical technology).

5. M. Suwa, A. C. Scott and E. H. Shortliffe, An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System, STAN-CS-82-922, Stanford University, June 1982.

6. TR2240, Report on Modular Avionic Packaging, NAC, August 3, 1978.

7. B. A. Zempolich, Technology Needs for Naval Avionics in the '80s, Avionics Systems and Technology - for the '80s, AIAA, February 1981.

8. R. E. Thun, Trends in Military Computer Technology, Defense Electronics, February 1984.

9. Technical Survey: VHSIC Chips Emerge, Aviation Week and Space Technology, July 30, 1984.

*Although this report references the limited document listed above, no limited information has been extracted.

10. D. Bursky, 1984 Technology Forecast - Semicustom IC's, Electronic Design, January 12, 1984.

11. W. J. Kitchen, Jr. and L. D.Sikes, VHSIC/VLSI Capabilities for DoD?, Defense Electronics, February 1984.

12. NASA Space Systems Technology Model, Volume IIB, Space Technology Trends and Forecasts, January 1984.

13. RADC-TR-79-309, BIT/External Test Figures of Merit and Demonstration Techniques, December 1979.

14. V. Tasar and H. Ohlef, A Method for Determining the Statistically Weighted Percent Fault Detection Coverage of a Self Test Program, Reliability and Maintainability Symposium Proceedings, 1979, pp. 39-43,

15. F. C. Kreuze, BIT Analysis and Design Reliability, Reliability and Maintainability Symposium Proceedings, 1983, pp. 328-333.

16. R. E. Collett and P. W. Bachant, Integration of BIT Effectiveness with FMECA, Reliability and Maintainability Symposium Proceedings, 1984. pp. 300-305.

17. N. Benowitz, D. Calhoun, and G. Lee, Fault Detection/Isolation Results from AAFIS Hardware Built-In Test, NAECON Record, 1976, pp. 215-22.

18. J. Bastian, W. Hochwald, and F. Suzuki, Figure of Merit for BIT Testability for VLSI, Autotestcon, 1979, pp. 90-95.

19. J. G. Malcolm, Practical Application of Bayes' Formulas, Proceedings of Reliability and Maintainability Symposium, 1983, pp. 180-186.

20. J.G. Malcolm and G. L. Foreman, The Need: Improved Diagnostics - Rather than Improved R, Proceedings of the Reliability and Maintainability Symposium, 1984, pp. 315-322.

21. R. A. Howard, Dynamic Probabilistic Systems, John Wiley & Sons, 1971.

22. D. Gleason, Analysis of Built-In-Test Accuracy, Proceedings of the Reliability and Maintainability Symposium, 1982, pp. 370-372.

23. M. A. Ramirez, Built-In Test Self-Verification, Proceedings of the Reliability and Maintainability Symposium, 1983, pp. 312-314.

24. D. P. Siewiorek and R. S. Swarz, The Theory and Practice of Reliable System Design, Digital Press, 1982.

25. M. A. Breuer and A. D. Friedman, Diagnosis and Reliable Design of Digital Systems, Computer Science Press, 1976.

26. E. Fujiwara, Self Testing Group-Parity Prediction Checker and its Use for Built-In Testing, Fault Tolerant Computing Symposium, 1983, pp. 146-153.

27. J. Hughes, E. J. McCluskey and D. Lu, Design of Totally Self-Checking Comparators with an Arbitrary Number of Inputs, Fault Tolerant Computing Symposium, 1983, pp. 169-172.

28. M. W. Sievers and A. Avizienis, Analysis of a Class of Totally Self-Checking Functions Implemented in a MOS LSI General Logic Structure, Fault Tolerant Computing Symposium, 1981, pp. 256-261.

29. R. J. Spillman, A Markov Model of Intermittent Faults in Digital Systems, Fault Tolerant Computing Symposium, 1977, pp. 157-161.

30. M.D.Beaudry, Performance Related Reliability Measures for Computing Systems, IEEE Transactions on Computers, 1978, pp. 550-547.

31. Guide to Fault Simulators, VLSI Design, July 1984.

32. J. C. Bezdek, Pattern Recognition with Fuzzy Object Function Algorithms, Plenum Press, New York, 1981.

33. J. Werner and R. Beresford, A System Engineer's Guide to Fault Simulators, VLSI Design, February 1984.

34. D. R. Coelho, Behavioral Simulation of LSI and VLSI Circuits, VLSI Design, February 1984.

35. S. Evanczuk, Getting Out of the Gate: High Level Modeling in Circuit Design, VLSI Design, January 1985.

36. S. M. Thatte and J. A. Abraham, Test Generation for Microprocessors, IEEE Transactions on Computers, C-33, 1980, p. 429 - 441.

37. D. Brahme and J. A. Abraham, Functional Testing of Microprocessors, IEEE Transactions on Computers, C-33, 1984, p.475 - 485.

38. S. Davidson, Fault Simulation at the Architectural Level, 1984 International Test Conference, IEEE, 1984, p. 669 - 679.

39. W. A. Rogers and J. A. Abraham, CHIEFS: A Concurrent, Hierarchical and Extensible Fault Simulator, 1985, International Test Conference, IEEE, 1985, p. 710 - 716.

40. R. J. Spillman, A Continuous Time Model of Multiple Intermittent Faults in Digital Systems, Comput. & Elect. Engng., 8, 1981, pp. 27-40.

41. AFWAL-TR-83-1141, Multibus Avionic Architecture Design Study, October 1983.

42. RADC-TR-83-291, Advanced Application of Printed Circuit Board Testability Design and Rating System, December 1983.

43. R. J. Spillman, An Analysis of the Effects of Intermittent Faults on Digital Systems, Ph.D. Thesis, Utah State University, 1978, p. 36-38.

BIBLIOGRAPHY FOR BIT VERIFICATION TECHNIQUES CONTRACT F30602-84-C-0021 (RADC)

KEY

A	Author's name; each on a separate line
B	Title of book containing item identified by T entry
D	Date
E	Editor of book
I	Issuer (publisher) of book
J	Journal name
K	Keys for searching
N	Issue number
O	Other information -- one line summary
P	Pages of article
R	Report number
T	Title
V	Volume number
X	Summary of item
Y	Comments or suggestions for further work

```
%T Statistical Demonstration of Fault-Isolation Requirements
%A J.E. Angus
%A R.E. Schafer
%B IEEE Transactions on Reliability
%D June 1980
%V R-29
%N 2
%P 116-121
%I IEEE
%X Suggests the use of the multinominal distribution
   for the statistical demonstration of fault isolation requirements.
   Basically it develops the likelihood test statistic and uses it
   to determine the number of trials required to accept or reject
   the hypothesis that the fault isolation requirements have been met.
%Y This could be used for TSV to verify that the testability
   parameters have been met.  It would require a little adaption.


%T A Method for Determining the Statistically Weighted Percent
   Fault Detection Coverage of a Self-Test Program
%A V. Tasar
%A H. Ohlef
%B Reliability and Maintainability Symposium
%D 1979
%P 39-43
%I IEEE
%X Assumes that the probability of faults in the system may vary,
   that is some faults are more likely than other faults, and uses
   this fact to calculate the fault coverage of a self-test (or BIT)
   system.  The major problem of course is how to estimate the
   different probabilities of faults.
%Y Uses FMEA, in fact, it has a good summary of the steps involved
```

in a statistical FMEA. Some interesting mathematics.

%T A Class of Test Generators for Built-in Testing
%A E.M. Aboulhamid
%A E. Cerny
%B IEEE Conference on Circuits & Computers
%D 1982
%P 456-459
%I IEEE
%X Provides a short review of the four types of test
generators for BIT and suggests a new type of BIT test generator.
While the article briefly considers the problem of self-test
in this BIT test generator, it does not have any suggestions
for TSV or TSCA.
%Y No direct application, however, it may be useful to follow up on
the different approaches to BIT since TSV depends on the nature
of the tests used in the BIT.

%T Impact of BIT on Avionics Maintainability
%A C. Locurto
%B Reliability and Maintainability Symposium
%D 1983
%P 333-336
%I IEEE
%X Looks at the Navy experience with BIT false alarms and concludes
that as BIT increases there is a point when maintenance actions
begin to increase because of the false alarms.
%Y Does use Malcolm's work to calculate the U-shaped nature of the
BIT-Maintenance curve.

%T On Built-in Test Techniques in Reliable Computer Systems
%A I.M. Soi
%A K.K. Aggarwal
%B Computers & Electrical Engineering
%D 1981
%V 8
%N 2
%P 109-114
%I Pergamon Press
%X Good summary article on BIT design. Offers a 5 step approach to
BIT design methodology which includes in step 5 the identification
and application of BIT evaluation measures. The authors outline
5 performance parameters:

1. Probability of system fault detection
2. Probability of localizing the faulty element
3. Probability of a false alarm
4. Time to system fault detection
5. Time to localize the faulty element

%Y The article only mentioned the BIT evaluation parameters and did
not suggest how to calculate them. Overall, it is a good summary
article but does not contain any work to expand upon.

%T Meeting the Challenge of BIT/BIST with ATE
%A L.S. Smith
%A R.S. Kjelgaard
%A D.M. Pizinger
%B Autotestcon
%D 1983
%P 96-100
%I IEEE
%X Methods of using BIT/BIST to improve ATE performance rather than allowing BIT/BIST to hamper ATE evaluations are suggested. The article followed several examples to board testing and did not go into any real detail in terms of their suggestions.
%Y Very little application to BIT verification.

%T Practical Application of Bayes' Formulas
%A J.G. Malcolm
%B Reliability and Maintainability Symposium
%D 1983
%P 180-186
%I IEEE
%X Good introduction to the mathematics of Bayesian statistics. Uses Bayes' formula to evaluate test effectiveness. Covers 5 example applications and suggests some good guidelines for using the results.
%Y Very good paper with some potential for an application to task 3.

%T BIT Analysis and Design Reliability
%A F.J. Kreuze
%B Reliability and Maintainability Symposium
%D 1983
%P 328-332
%I IEEE
%X Develops an FMEA-derived BIT analysis technique which is applied to a Digital Flight Control Servo Loop. Also suggests the use of fault insertion procedures. Certainly provides justification for the need for BIT verification.
%Y The example seems to indicate that the FMEA procedure may work well for BIT analysis during the design phase. Should be pursued.

%T Built-in Test Improves Expendable Weapon Readiness
%A R.O. Holbrook
%B Reliability and Maintainability Symposium
%D 1983
%P 339-343
%I IEEE
%X Looks at the impact of BIT on system performance. It assumes a BIT detectability factor, K, but does not suggest how to calculate it. However, given this factor, it offers some mathematical approaches to evaluate the overall effect of BIT on a system.
%Y The procedures are not of much use, but this article should be looked at again if the probablistic approach of Malcolm is

developed since it does use some interesting mathematics which may be of some help.

%T Design Guidlines and Optimization Procedures for Test Subsystem Design
%A D.N. Lord
%A G.A. Walz
%A S. Green
%R RADC-TR-80-111
%D April 1980
%X Provides guidelines and procedures to optimize the design of BIT by properly specifying three key design parameters (test effectiveness, mean corrective maintenance time and test subsystem production costs). These then form the "design to" criteria during development.
%Y There is cursory mention (page 162) of analysis of fault detection using FMEA techniques. Indicates that this is easily done by experienced designers "given only the schematic and general performance parameters."

%T Design and Evaluation Methodology for Built-in Test
%A D.N. Lord
%A D. Gleason
%B IEEE Transactions on Reliability
%D August 1981
%V R-30
%N 3
%P 222-226
%I IEEE
%X Suggests the use of BIT effectiveness as a performance evaluation parameter where effectiveness is defined as the total number of malfunctioning units divided by the total number of maintenance actions. BIT effectiveness then centers on the number of "no defect maintenance actions." The article then examines BIT in terms of its impact on system reliability, maintainability, and availability.
%Y BIT effectiveness may have some role to play in TSV, but at the moment it seems to rely on experimental data which can only be produced after the equipment is in service.

%T Integration of BIT Effectiveness with FMECA
%A R.E. Collett
%A P.W. Bachant
%B Reliability and Maintainability Symposium
%D 1984
%P 300-305
%I IEEE
%X Proposes the use of FMECA as a method for determining BIT effectiveness. The article offers a suggested FMECA worksheet which includes BIT evaluation. The method has been applied at GTE.
%Y While the method has been applied at GTE, from the article it does not seem to be well developed, and it may be one area to

consider for futher work.

%T System Test Visibility -- or Why Can't You Test Your Electronics?
%A F. Danner
%B IEEE Test Conference
%D 1983
%P 635-639
%I IEEE
%Y Not of much use to BIT verification. Mentions the need for improvements in testing and BIT systems, but does not offer many suggestions with the exception of the possibility of the use of AI.

%T Built-in Verification Test
%A E.J. McCluskey
%B IEEE Test Conference
%D 1982
%P 183-190
%I IEEE
%X Presents a method called verification test which leads to the development of a minimal test set for testing combinational circuits.
%Y While it can be used in a BIT system it is of limited value to BIT verification.

%T Analysis of Built-in Test Accuracy
%A D. Gleason
%B Reliability and Maintainability Symposium
%D 1982
%P 370-372
%I IEEE
%X Uses a simple Markov model to determine the probability of the correct operation of the BIT system (called BIT accuracy). The basic differential equations are set up and solved. The result is a simple expression for BIT accuracy in terms of the failure rates.
%Y Only the first step in the model development had been completed. It should be possible to develop the model further and look at such variables as the first passage time, stable state probabilities, etc.

%T A Method of Estimating the Effect of Design for Testability of PC Board Test Costs
%A J.B. Waltrich
%B IEEE Test Conference
%D 1980
%P 176-184
%I IEEE
%X Develops a measure of ATE efficiency based on the number of diagnostic probes required.
%Y Little application to BIT verification.

%T Figure of Merit for BIT Testability for VLSI
%A J.D. Bastian
%A W. Hochwald
%A F.T. Suzuki
%B Autotestcon
%D 1979
%P 90-95
%I IEEE
%X Considers the problem of evaluation of BIT in VLSI systems. Proposes the use of digital simulation with fault insertion as a possible evaluation tool. Runs an example of simulation of a BIT system for a microprocessor.
%Y The simulation tool is very complex and requires a high skill level to use. It would need to be modified significantly to be of practical use.

%T The Need: Improved Diagnostics - Rather than Improved R
%A J.G. Malcolm
%A R.L. Forman
%B Reliability and Maintainability Symposium
%D 1984
%P 315-322
%I IEEE
%X Develops a Bayesian classifier for filtering BIT false alarms. Shows that false alarms become more significant as the reliability of devices increases. Proposes using the Bayesian classifier in conjunction with other techniques such as artificial intelligence to develop a "smart BIT" concept.
%Y Standard Bayesian analysis could be adapted for a TSCA classifier.

%T Built-In-Test Self-Verification
%A M.A. Ramirez
%B Reliability and Maintainability Symposium
%D 1984
%P 312-314
%I IEEE
%X Suggests a "forced" failure mode to test BIT operation. Suggests use of highly reldable BIT when "forced failure" is impractical.
%Y TSCA candidate

%T BIT/SIT Improvement Project (Phase I): Evaluation of Selected USAF Aircraft BIT/SIT Systems/Subsystems
%R ASD-TR-79-5013
%D July 1979
%X Reviews eleven selected Air Force systems and summarizes the BIT or system integrated test (SIT) approach for each. Also compares required performance to field experience. Refers to TSV of three subsystems being performed "by paper analysis only". Recommends using BIT/SIT during developmental flight testing for engineering evaluation of its performance and demonstrating BIT/SIT duringthe reliability and environmental testing.

%Y Does not provide details of TSV techniques used, but may be used
as a source for personal contacts to obtain further information.

%T An Evaluation Tool of Fault Detection Mechanisms Efficiency
%A B. Decouty
%A G. Michel
%A C. Wagner
%B Fault-Tolerant Computing Symposium
%D 1980
%P 225-227
%I IEEE
%X Describes a tool for simulating faults in a test item and monitoring its fault detection mechanism to determine if the faults are detected. Components of the test article are replaced by a probe and faults are simulated by a fault generator through the probe.
%Y Explains basic operation of the system but does not expand on how the results are to be evaluated.

%T Design Specifications of a Self-Checking Detection Processor
%A Y.Crouzet
%A C.Landrault
%B Fault-Tolerant Computing Symposium
%D 1980
%P 275-277
%I IEEE
%X Provides specifications for a self-checking detection processor and illustrates how it would be used in various fault-tolerant computing and communication architectures. The detection processor would be implemented as a VLSI device and used to monitor for and detect failures.
%Y Good background for a BIT processor, but does not elaborate on the self-checking features.

%T A Measure of BIT/ATE Effectiveness
%A D.Gleason
%B ATE Seminar/Exhibit and Test Instruments Conference
%D January 1980
%P 90-101
%I Benwill Publishing Corporation
%X Proposes the use of the expected number of removals (ENR) as a measure of BIT effectiveness. ENR is calculated as a function of:

P(FD) - probability of fault detection
X - average ambiguity level
P(MA) - missalignment factor
FAR - false alarm factor
RRi - maintenance policy removal rate

%Y Utilizes the specifications for the above factors to compute ENR; does not address the evaluation of the design implementation.

%T Radar BIT Design
%A D.L. Emerson
%A R. Meyer
%B Autotestcon
%D 1977
%P 24-33
%I IEEE
%X Describes the BIT design process using radar BIT as an example. Describes the test thoroughness computation at the component level as a TSV technique. Makes a case that verification should be made at the functional performance level since BIT is designed to test at the level. Proposes that preproduction developmental tests be used to "tune" BIT tolerances and for verification of BIT performance.
%Y Good guide for radar BIT design. Not much guidance for BIT verification.

%T An Advanced Fault Isolation System for Digital Logic
%A N. Benowitz
%A D.F. Calhoun
%A G.E. Alderson
%A J.E. Bauer
%A C.T. Joeckel
%B IEEE Transactions on Computers
%D May 1975
%V C-24
%N 5
%P 489-497
%I IEEE
%X Describes a concept for implementing and using BIT in digital equipment, developed as a result of the Advanced Avionics Fault Isolation System (AAFIS) program. BIT effectiveness was demonstrated by simulation using Hughes' SATGEN (Simulation and Test Generation) computer program.
%Y Does not describe SATGEN.

%T Fault Detection/Isolation Results from AAFIS Hardware Built-in Test
%A N. Benowitz
%A D.F. Calhoun
%A G.W.K. Lee
%B NAECON
%D 1976
%P 215-222
%I IEEE
%X Describes implementation and testing of AAFIS BIT concepts. Testing was accomplished by insertion of 275 faults and recording fault detection and isolation results. Faults were stuck-at-one or stuck-at-zero faults at integrated circuit pins or adjacent pins shorted.
%Y There is no discussion of adequacy or validity of the test method or the results. There is a statement that detection and isolation of faults inserted in BIT are to higher levels using AAFIS than for non-BIT circuits. There may be some aspect of

AAFIS design applicable to TSCA.

%T The Theory and Practice of Reliable System Design
%A D.P. Siewiorek
%A R.S. Swarz
%D 1982
%I Digital Press
%X Provides a comprehensive reference for design of fault tolerant computing systems. The first half describes the concept of fault tolerance, how faults manifest themselves, fault detection and redundancy techniques, evaluation criteria and cost considerations. The second half describes specific applications of fault tolerant design.
%Y In the second half there are specific references to BIT verification techniques used in fault tolerant computing systems.

1. Test system condition assessment in the VAX-11/780 by incorporation of logic to force error conditions in various CPU functions.
2. Test system condition assessment in the Sperry Univac 1100/60 by fault injection using hardware and software. Also notes that this capability was used for test system verification.
3. Test system verification for several Electronic Switching System processors using both physical simulation by inserting faults in hardware and digital simulation using computer models.
4. Test System verification for the Voyager spacecraft computer by inserting simulated failures via support equipment or loading into memory data corresponding to software-sensed failures.

%T Early Implementation/Measurement of Testability
%A L.J. Rhodes
%B Reliability and Maintainability Symposium
%D 1981
%P 55-58
%I IEEE
%X Addresses BIT verification by both analysis and test. The analysis given is based on failure rate of detected (or isolated) failures and the total failure rate but does not indicate how to determine the failure rates. Testing in conjunction with the maintainability demonstration is described. Discussion of required sample size concludes that large samples (45/SRU) are needed.
%Y Valid concern for test sample size, but little other detail provided.

%T Testability Analysis
%A F.G. Kovijanic
%B IEEE Test Conference
%D 1979
%P 310-316

%I IEEE
%X Presents a method to quantify and measure testability of a network based on its controlability and observability aspects.
%Y Useful for BIT design but not BIT verification. Refers to coverage results of automatic test generation programs.

%T MICROBIT, A Method of Built-in Test for Microcomputers
%A P.P. Fasang
%J Siemens Forschungs und Entwicklungsberichte
%D 1983
%V 12
%N 1
%P 47-54
%I Springer-Verlag
%X Describes the hardware and software design of BIT for a general purpose microcomputer system. The BIT requires no external test equipment or human intervention. Also describes tests of BIT by physically inserting stuck-at-one and stuck-at-zero faults on the pins of the devices.
%Y Does not provide an assessment of the quality of the tests. States that faults were inserted at the pins because access to the inside of the devices is not possible.

%T US Army Test and Evaluation Command, Commodity Service Test Procedure - Built-in Test
%R MTP-7-3-058
%D November 1971
%X Provides guidelines for developing test methods to determine the degree to which aircraft BIT and associated test equipment meet the requirements.
%Y The procedure uses actual faulty aircraft and failures of opportunity. It gives no guidance on the analysis of the data. The procedure would be of little use to affect design.

%T Fault Detection/Isolation Verification
%A W.T. Etter
%A R.A. Meyer
%A D.E. Krzysiak
%R RADC-TR-82-232
%D August 1982
%X Documents a program to test and evaluate the fault detection and isolation algorithm developed by GTE Sylvania under F30602-76-C-0433. The system was emulated at a functional level then the fault detection/isolation was tested for various scenarios including multiple faults, false alarms and intermittents.
%Y There were only a limited number of fault scenarios and the results were devoted more to test time and fault FD/FI algorithm memory requirements than to algorithm accuracy.

%T Onboard Test System Design Guide
%A K. Derbyshire

%A G. Bramhall
%A F. Hait
%R ASD-TR-81-5023
%D August 1981
%X Defines and describes an onboard test system based on past experience with BIT, BITE, A/E and the Central Integrated Test System (CITS) for the B-1 aircraft. Guidelines and ground rules are presented for the development of such an onboard test system from the conceptual stage through to delivery and acceptance of hardware and software.
%Y Addresses test system verification during design as a part of FMEA but offers no substantive guidance. The sample specification requires verification by test using simulation of all safety-of-flight and mission critical faults and a portion of all other faults. No details of methods are provided. The onboard test system described is a separate subsystem, and a self test capability is specified for it. This essentially provides a test system condition assessment capability.

%T Design Guide, Built-in Test (BIT) and Built-in Test Equipment (BITE), Army Missile Systems
%R DRSMI/RL-CR-81-4
%D April 1981
%X Provides guidelines for specifying, implementing and evaluating BIT/BITE. Provides discussion of options available rather than a "how to design" handbook.
%Y Discusses use of FMEA to evaluate BIT but does not address methodologies.

%T Measurement of Fault Latency in a Digital Avionic Processor
%A J.G. McGough
%A F.L. Swern
%R NASA Contractor Report 3462
%D October 1981
%X Describes a gate level emulation of the Bendix BDX-930 digital computer and its use to measure BIT coverage and fault latency. Faults were simulated as stuck-at-1 and stuck-at-0 faults. A comparison was made between results with faults inserted at the gate level and results with faults inserted at the component pin level.
%Y May be able to estimate the cost of developing and using similar emulations but results may not be applicable since the BDX-930 consists of SSI and MSI type integrated circuits which are more easily modeled than LSI and VLSI type devices.

%T BIT False Alarms: An Important Factor In Operational Readiness
%A J.G. Malcolm
%B Reliability and Maintainability Symposium
%D 1982
%P 206-212
%I IEEE
%X Describes the effects of false alarms on operational

readiness. Develops analysis techniques for analyzing BIT systems to determine causes of false alarms. Finally, presents concepts for designing BIT to reduce the effects of false alarms.
%Y No mention of BIT verification, but good description of potentially new BIT techniques.

%T Diagnosis & Reliable Design of Digital Systems
%A M.A. Breuer
%A A.D. Friedman
%D 1976
%I Computer Science Press
%X Textbook on test concepts and test design for digital systems. Includes sections on testing combinational circuits, testing sequential circuits, logic level simulation, and fault tolerant design.
%Y Good overview of simulation systems and good section on self-checking circuits.

%T A Self-Testing Group-Parity Prediction Checker and Its Use for Built-in Testing
%A E. Fujiwara
%B Fault Tolerant Computing Symposium
%D 1983
%P 146-153
%I IEEE
%X Describes and evaluates an error checking scheme for multiple output combinational circuits.
%Y An example of self-checking checkers.

%T Design of Totally Self-Checking Comparators with an Arbitrary Number of Inputs
%A J.L.A. Hughes
%A E.J. McCluskey
%A D.J. Lu
%B Fault Tolerant Computing Symposium
%D 1983
%P 169-172
%I IEEE
%X Describes a design for a self-checking comparator and proposes applications.
%Y An example of self-checking checkers.

%T Analysis of a Class of Totally Self-Checking Functions Implemented in a MOS LSI General Logic Structure
%A M.W. Sievers
%A A. Avizienis
%B Fault Tolerant Computing Symposium
%D 1981
%P 256-261
%I IEEE
%X Presents a general logic structure for implementation of

the functional core of an integrated circuit, and fault models for NMOS and CMOS versions of the structure. These models are used to analyze a class of totally self-checking functions.

8) An example of self-checking checkers.

APPENDIX A

ASSESSMENT OF VHSIC BUILT-IN TEST

A.1 INTRODUCTION

This appendix provides a general survey,completed in June 1985 as part of this contract, of the current state of the VHSIC phase I contractors and their primary built-in test (BIT) techniques. The information was derived from three sources: 1) The specification handbooks of the various contractors, issued by the VHSIC program office in January, 1984; 2) The most recent technical review reports issued by the contractors; 3) The article "New Circuits Expected to Exceed Projection" which appeared in the July 30, 1984 issue of Aviation Week & Space Technology.

VHSIC stands for Very High Speed Integrated Circuits. As the program name implies, the emphasis is on circuit speed. It was initiated to overcome the technology availability lag from commercial to military products. Sponsorship by the Department of Defense (DoD) is oriented towards development of state-of-the-art military grade parts whose initial applications will be in DOD programs. Speed increases are achieved by using existing technology and design techniques while reducing feature sizes on the integrated circuits.

Another primary focus in the VHSIC program is testability. The contractors are required to have some form of built-in test on the chips. Because of the ambitious nature of the chipsets it was felt that the parts would be difficult to test and use without this feature.

There are six phase I contractors. The sponsors of the contractors are the Army, Navy and Air Force. Each is the sponsor of two of the contractors. Every six months the contractors have a technical review which is attended by all of the sponsors and other interested parties. At that time they report on their progress. All sponsors supply inputs in the review process.

A.2 CHIPSET SUMMARY

A.2.1 Honeywell

A.2.1.1 The Chipset

Chip	Complexity (No. Transistors)	Technology	Status
Parallel Pipeline Processor	142,000	Bipolar	Under test
Arithmetic Unit	121,000	Bipolar	Functional
Sequencer	136,000	Bipolar	Functional

A.2.1.2. General

This chipset is designed for use in electro-optical sensor signal processing. All three chips are microcoded, simplifying modifications. Built-in test is handled at the chip-set level rather than at the chip level. This was possible because all three chips are needed in any system and it permitted Honeywell to have the chips share some of the test circuitry used, thus reducing the impact upon chip size. An additional controller chip set for on-line testing and fault isolation is under development.

A.2.1.3 Technology Insertion

The systems which have been targeted for the Honeywell chipset are an avionics suite for the LHX helicopter and a distributed processor for ballistic missile defense.

A.2.1.4 The Chips.

A.2.1.4.1 Parallel Pipeline Processor (PPP)

The PPP chip forms an electro-optical signal processor when used in conjunction with the other two Honeywell chips. It features a 16-bit processing element, a 512 by 8-bit processing element memory and double buffered I/O memory. The PPP operates at a 25 MHz rate.

A.2.1.4.1.1 Built-In Test for the PPP

Signature analysis is used for the built-in test. Signature analysis requires both a test vector (or pattern) generator and a signature generator (which compresses the results into a smaller code word). The PPP chip contains only the signature generator. The test patterns must be supplied by a control chip or other source.

A.2.1.4.2 Sequencer Chip

The Sequencer chip supplies high speed control signals for the PPP and arithmetic chip. Up to 16 PPP chips may be supported by a single sequencer chip. The chip operates at a 25 MHz rate.

A.2.1.4.2.1 Built-In Test for the Sequencer Chip

Signature analysis is used for the Sequencer chip's built-in test. Both a test vector generator and a signature generator are on-board the chip.

A.2.1.4.3 Arithmetic Chip

The Arithmetic chip generates addresses at a 25 MHz rate for the PPP chip. It contains special hardware for rapid 2-dimension vector generation. Two Arithmetic Logic Units (ALUs) are used for each dimension. Hardware is also used to compute absolute, relative and circular addresses.

A.2.1.4.3.1 Built-In Test for the Arithmetic Chip

Signature analysis is used for the Arithmetic chip's built-in test. The signature analyzer in the Sequencer chip is used to generate test results for this chip.

A.2.1.5 System Level Test Procedure

None of the documents indicate how to test the chipset while it is in a system. Whether the test data is read out over a dedicated line or over a bus was not indicated.

A.2.2 Hughes Aircraft Company

A.2.2.1 The Chipset

Chip	Complexity (No. Transistors)	Technology	Status
Digital Correlator	72,000	CMOS/SOS	Functional
Algebraic Encoder/ Decoder	79,000	CMOS/SOS	In Fabrication
Signal Tracking Subsystem	~60,000	CMOS/SOS	Design complete 10/84
Electro-Optical Signal Proc.		CMOS/SOS	Design complete 8/84
Configurable Gate Array	~32,000	CMOS/SOS	In Fabrication

The Hughes chipset is designed for a wide range of communications systems that operate in a high noise or highly jammed environment. The Signal Processor and Gate Array are recent additions to the chip set and data is not yet available for them.

A.2.2.3 Technology Insertion

The systems which have been targeted for the Hughes chipset are a hybrid of the Position Location and Reporting System (PLRS) and the Joint Tactical Information Distribution Location and Reporting System (JTIDS) called the PLRS/JTIDS hybrid and a signal processor for the F/A-18.

The PLRS/JTIDS hybrid system will be able to operate in both the PLRS and JTIDS environments though it may not be able to operate in all modes in both. The F/A-18 will use Hughes designed parts fabricated by Fairchild.

A.2.2.4 The Chips

A.2.2.4.1 Digital Correlator

The digital correlator is a 129-stage by 4-bit correlator configured as four 32-stage by 4-bit sections. Each stage correlates two bits of in-phase data and two bits of quadrature-phase data with a single reference bit. By proper scaling of the sections, the chip may be reconfigured as a 64-stage by 8-bit correlator with each stage correlating with four bits of in-phase and four bits of quadrature-phase data. The length of each section may be selected to be from 2 stages to 32 stages in steps of quadrature-phase data. The length of each section may be selected to be from 2 stages to 32 stages in steps of 2. A magnitude circuit is available at each section for noncoherent correlation. A threshold circuit is provided on each chip. The chips are configured by passing data to internal control registers over the system command data-bus. Correlator chips may be cascaded to form correlators with lengths of up to 10,912 stages.

A.2.2.4.1.1 Built-In Test for the Correlator Chip

The correlator uses both set/scan and signature analysis for its built-in test. Each correlator section uses signature analysis. The signal register is configured as a 32 stage pattern generator and the reference register as a 32 bit signature analyzer. Initial patterns may be loaded into the pattern generator and signature analyzer over the command databus. The number of clocks in the test are controlled externally by a TEST pin input. Communications between the sections on the chip and additional circuitry (such as the threshold detection circuit) are tested by set/scan. The input and output registers are linked together as a long setscan chain. Data may be input to this chain over the system command databus. The data is then processed (controlled by the TEST pin). Results are read back through the command databus to the control processor. In both tests, the control processor checks the results against known good results. Fault simulation indicates a coverage of better than 95%.

A.2.2.4.2 The Algebraic Encoder/Decoder Chip

The Encoder/Decoder chip possesses processing elements capable of encoding and decoding all primitive binary

Bose-Chaudhuri-Hocquenghem (BCH) codes, including Reed Solomon codes, up to length 64, with 32 data bits. Data may be passed through the chip either by dedicated data pins or the command databus. The eight-bit command databus connects the chip to the control processor and is used for chip configuration and test commands.

A.2.2.4.2.1 Built-In Test for the Algebraic Encoder/ Decoder Chip

All of the registers on the chip, except for some of those in the test circuitry, are included in one of 5 set/scan chains. The set/scan chains are loaded over the command databus by the controlling processor. The test is then initiated by a command sent over the bus. On-board circuits time the test. Status bits report results of the test. The set/scan registers may also be read to help isolate faults. Fault coverage information was not available.

A.2.2.4.3 Signal Tracking Subsystem Chip (STS)

This chip contains one tracking loop which may be configured for phase frequency or code tracking. Also, one pseudo-random noise (PN) sequence generator and a code/sample clock are on-board. Two or more chips are required for most applications. The chips are programmable to handle a wide range of signal tracking requirements.

A.2.2.4.3.1 Built-In Test for the Signal Tracking Subsystem Chip

Set/scan and signature analysis are both used for the STS chip. Tests are initiated by a control processor with commands over the control databus. Results are read by the controlling processor and compared to known good results. Fault coverage information was not available.

A.2.2.4.4 Electro-Optical Signal Processor

This part was originally an in-house project. It was recently picked up for support by the VHSIC program. As a result, no information was available.

A.2.2.4.5 Configurable Gate Array

This part was originally an in-house project. It was recently picked up for support by the VHSIC program. As a result, no information was available in the documents examined.

A.2.2.5 System Test Procedure

These chips are designed to operate in a system which is controlled by a microprocessor or other smart controller. The same databus used to program the chips 'for a particular configuration (which configuration may be changed on-line) is used to control the testing of the parts. The actual testing must be done off-line (that is, during time periods when data is not being transmitted or received). The testing is initiated by transmitting a command to put tha parts in the self test mode. At this time their set/scan registers may be initialized for the test. An externally supplied timing signal is used to begin the test process. At the end of the test, results are read by the controlling processor and compared to stored values of the correct results.

A.2.3 IBM

A.2.3.1 The Chipset

Chip	Complexity (No. Transistors)	Technology	Status
Complex Multiply & Accumulate	100,000	NMOS	Many delivered to DoD

A.2.3.2 General

The IBM chipset consists of just one part, the Complex Multiply & Accumulate (CMA) chip. The CMA chip can perform 100 million multiplications per second. Four 16 by 16 multipliers with accumulators, each with a 24-bit output, are available on chip for real or complex multiplication and addition, 100 million real or 25 million complex operations per second may be performed. Each subsection (containing multiplication and addition circuitry) operates at 12.5 MHz real or 6.25 MHz complex rates.

A.2.3.3 Technology Insertion

The systems which have been targeted for the IBM chip are the AN/UYS-1 sonobuoy signal processor used on the Lockheed P-3C and S-3A aircraft and the IBM/Sikorsky Lamps Mk.3.

A.2.3.4 Built-In Test for the Complex Multiply and Accumulate Chip

Signature analysis using a level sensitive scan design and on-chip monitoring are used for built-in test. Parity bits are available on all chip interfaces. Adder overflow and comparators for adjacent multipliers are also available. The comparator may be used when spare sections are available in the application. Errors are recorded in a 10-bit register and one of three interrupt requests is generated: one for parity errors (INT1), one for compare errors (INT2) and one for overflow errors (INT3). The chip is stopped on the detection of an error. All of the latches on the chip are connected to form 16 scan strings. A 20-bit pattern generator feeds all of the scan strings. Results of the test are formed in a 16-bit signature analyzer. This test has a fault coverage of over 90%.

A.2.3.5 System Test Procedures

Two types of testing are available on the chip, on-line and off-line. The off-line test requires that the system not be using the CMA for calculations during the test procedure. This is when the signature test is performed. It is typically performad during power-up procedures, but may also be performed periodically if the system schedules periods during which health status is determined. The on-line tests are performed during normal operation. The inputs and outputs to the chip include parity bits. Most I/O failures eventually lead to parity failure. The on-line test may be used when the system design results in spare multiply and accumulate sections being available. The results of adjacent sections are compared. If they do not match, the test has failed. Failure of any of the above tests results in an interrupt request being generated. At that point it is up to the control processor to take appropriate action.

A.2.4 Texas Instruments

A.2.4.1 The Chipset

Chip	Complexity (No. Transistors)	Technology	Status
Static RAM	460,000	NMOS	Some delivered to DoD
Multipath Switch	30,000	Bipolar	Some delivered to DoD
Array Controller/ Sequencer	120,000	Bipolar	Under test
Vector Arithmetic Logic Unit	85,000	Bipolar	Under test
Vector Address Generator	130,000	Bipolar	In layout
Data Processor Unit	190,000	Bipolar	In final design
General Buffer Unit	130,000	Bipolar	In final design
Device Interface	190,000	Bipolar	In final design

A.2.4.2 General

Four of the seven chips are designed for use in an array processor for high speed signal processing. The other three chips may be used in a standard 1750A type processor.

A.2.4.3 Technology Insertion

The systems which have been targeted for the Texas Instruments chip set are a MIL-STD-1750A data processor, the launch and leave guided bomb, the Hellfire air to surface missile guidance system, TOW-2 antitank missile guidance subsystem improvement, the Army LHX helicopter central processor, the Army M-1 tank fire control system processor and the Integrated Communication Navigation Identification Avionics (ICNIA) program.

A.2.4.4 The Chips

A.2.4.4.1 Static RAM

The Static RAM is a 72 kilobit Random Access Memory configured as an 8K by 9 bit memory. It has a parity generator and checker built in. The ninth bit is the parity bit. It features write protection circuitry for blocks of 1K words and pipelined operations to improve throughput.

A.2.4.4.1.1 Built-In Test for the Static RAM

The only built-in test feature on the RAM chip is the parity generator/checker circuitry.

A.2.4.4.2 Multipath Switch

The Multipath Switch contains a programmable crossbar multiplexer capable of providing up to six one-way paths between six 4-bit memory ports and six 4-bit processor ports. Up to six independent bus-to-bus connections per clock cycle are possible.

A.2.4.4.2.1 Built-In Test for the Multipath Switch

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.4.3 Array Controller/Sequencer (AC/S)

The AC/S chip provides microprogram addresses for 64K words of external memory or 2K words of internal mask-programmable ROM. It operates at a 25 MHz rate. Capability for sequential addressing, conditional or unconditional branching and subroutine call linking and return through a 16-deep stack are provided.

A.2.4.4.3.1 Built-In Test for the Array Controller/ Sequencer Chip

On-Chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. A microcode test checks all instructions and all of the AC/S functions. This test is also compressed using signature analysis. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.4.4 The Vector Arithmetic Logic Unit (VALU)

The VALU is a high performance computation unit optimized for signal processing applications. It forms part of a configurable arithmetic pipeline which includes a 16 by 16 multiplier, a 44-bit adder/subtracter and a 16-bit full capability arithmetic logic unit. It uses an external ROM microcode, making modifications to the code structure straight forward. While it does not perform floating point operations, it supports them with 9 opcodes.

A.2.4.4.4.1 Built-In Test for the Vector Arithmetic Logic Unit

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.4.5 Vector Address Generator (VAG) Chip

The VAG chip is a high speed, 16-bit addrass generator which is optimized for use with highly regular data structures such as arrays. A full 16-bit ALU (Arithmetic Logic Unit) is provided fcr calculations including addresses and other operands. It can handle sequential addressing, bit reversed addressing (for Fast Fourier Transforms (FFT)), indexed addressing and data directed addressing.

A.2.4.4.5.1 Built-In Test for the Vector Address Generator

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent. This test requires 6250 clocks or one fourth of a millisecond to run.

A.2.4.4.6 Data Processor Unit (DPU)

The Data Processor Unit is a general p' rpose microprocessor for the 1750A instruction set. It operates at speeds to 25 MHz with a maximum throughput at 25 MHz predicted to be two to four MIPS with the DIAS mix. It can perform a register to register add in 40 nanoseconds and a 16-bit multiply in 440 nanoseconds. It has three programmable timers.

A.2.4.4.6.1 Built-In Test for the Data Processor Unit

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.4.7 General Buffer Unit (GBU)

The GBU is a bus coupler used between the memory bus (MBUS) and the system bus (SBUS).

A.2.4.4.7.1 Built-In Test for the General Buffer Unit

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.4.8 Device Interface Unit (DIU)

The DIU is a general purpose microprocessor for use as a coprocessor with the DPU. It implements I/O, DMA and remote multiprocessing operations. It operates at speeds to 25 MHz and has 16 interrupt levels. It is the same circuit as the DPU but uses a different microcode.

A.2.4.4.8.1 Built-In Test for the Device Interface Unit

On-chip self testing is provided using signature analysis. If a result disagrees with the predetermined fault-free signature, an error bit is set. Fault coverage (excluding I/O circuitry) is expected to exceed 99 percent.

A.2.4.5 System Test Procedures

Tests for these chips are controlled by sending commands over the system maintenance bus. The signature analysis tests must be performed off-line. The test results may be read over the system maintenance bus. If the chipset is configured as the system main processor, testing is controlled by the Data Processor Unit. Periodic testing requires the availability of time segments during which the processor may be freed from normal duties. All chips report via the system maintenance bus to the DPU which stores the system health information in a status register. It is the responsibility of the DPU to act upon that information.

Testing may also be initiated by a separate processor over the system maintenance bus.

Fault simulations have not yet been performed to verify the high fault coverage predictions.

A.2.5 TRW/Motorola

A.2.5.1 The Chipset

Chip	Complexity (No. Transistors)	Technology	Status
Window Addressable Memory	58,000	Bipolar	Functional
Content Addressable Memory	66,000	Bipolar	Functional

Matrix Switch	13,500	Bipolar	Functional
Four Port Memory	60,000	CMOS	Functional
Multiplier/ Accumulator	42,000	Bipolar	Under Test
Microcontroller	25,000	Bipolar	In Fabrication
Register Arithmetic Logic Unit	36,000	Bipolar	In Fabrication
Address Generator	34,000	Bipolar	In Fabrication
Convolutional Decoder	73,000	Bipolar	In Design
Convolver	74,000	CMOS	In Fabrication
Fast Fourier Transform Arithmetic Unit	50,000	CMOS	Layout Complete
FFT Control Unit	60,000	CMOS	In Design
Configurable Gate Array	26,000	CMOS	In Design

A.2.5.2 General

The TRW/Motorola chip set is designed for use in electronic warfare and extremely high frequency satellite communications.

A.2.5.3 Technology Insertion

The systems which have been targeted for the TRW/Motorola chipset are a programmable signal processor for ICNIA, a general purpose programmable signal processor, the AN/ALG-131 airborne jammer pod improvement program, INEWS (Integrated Electronic Warfare System) and a next generation military satellite.

A.2.5.4 The Chips

A.2.5.4.1 Window Addressable Memory (WAM)

The WAM chip allows incoming data to be examined through eight different adjustable "windows" simultaneously to determine if certain variables fall in the range of values viewed through the windows. It provides an eight-way sorting action.

A.2.5.4.1.1 Built-In Test for the WAM

Information on the WAM was not included in the chip's specification document.

A.2.5.4.2 Content Addressable Memory (CAM)

CAM allows a string of input data bits, called a vector, to be compared with previously stored data. A vector may be divided into fields, permitting searches for matches in a number of ways.

A.2.5.4.2.1 Built-In Test for the Content Addressable Memory

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.3 Matrix Switch

The Matrix Switch is an eight-input eight-output 4-deep crossbar switch capable of operating at 25 MHz. It switches resources and data between memories, arithmetic units, the address generator, etc.

A.2.5.4.3.1 Built-In Test for the Matrix Switch

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.4 Four Port Memory

The Four Port Memory is designed to read two independent addresses and write two independent addresses each clock cycle. It is configured as 1024 4-bit words with pipelining and both synchronous and asynchronous modes of operation.

A.2.5.4.4.1 Built-In Test for the Four Port Memory

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.5 Multiplier/Accumulator (MAC)

The MAC chip computes sums of products and accumulates intermediate data.

A.2.5.4.5.1 Built-In Test for the Multiplier Accumulator

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.6 Microcontroller

The microcontroler provides microinstruction sequencing control. It generates a 16-bit address.

A.2.5.4.6.1 Built-In Test for the Microcontroller

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5

A.2.5.4.7 Register Arithmetic Logic Unit (RALU)

The RALU implements 16-bit arithmetic and boolean functions. Single chip double precision is supported. Multichip multiple precision is supported (with loss of speed from the 25 MHz single chip operation). The chip contains features which facilitate floating point operations.

A.2.5.4.7.1 Built-In Test for the Register Arithmetic Logic Unit

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.8 Address Generator (AG)

The AG chip is a slave processor which generates sequences of 16-bit addresses (one address in each clock cycle) for quick access to data structures.

A.2.5.4.8.1 Built-In Test for the Address Generator

Set/scan is provided through a maintenance network node. Data is loaded into the chip, which may then be single stepped. The resulting data is then read out to see if it was correctly processed. Information on the maintenance network node is included in section A.2.5.5.

A.2.5.4.9 Convolutional Decoder

Information on the Convolutional Decoder was not included in the chips specification.

A.2.5.4.10 Convolver

Information on the Convolver was not included in the chips specification.

A.2.5.4.11 Fast Fourier Transform Arithmetic Unit

Information on the Fast Fourier Transform Arithmetic Unit was not included in the chips specification.

A.2.5.4.12 FFT Control Unit

Information on the FFT Control Unit was not included in the chips specification.

A.2.5.4.13 Configurable Gate Array

Information on the Configurable Gate Array was not included in the chips specification.

A.2.5.5 System Test Procedure

Each chip has a maintenance network node (MNN). The purpose of the node is to extract command information and data from a serial input stream and initiate a response to that command. All of the registers and flip-flops on each chip, except those in the MNN itself, are configurable for inclusion in a set/scan loop which may be loaded from the MMN. The scan mode provides readout without altering the contents of the chip. System tests must control the MMN nodes of the chip with the proper command stream and be able to interpret the results obtained from any test.

A.2.6 Westinghouse/National

A.2.6.1 The Chipset

Chip	Complexity (No. Transistor)	Technology	Status
Static RAM	400,000	CMOS	4Q84
Pipeline Arithmetic Unit	133,000	CMOS	1Q85
Extended Arithmetic Unit Multiplier	92,000	CMOS	1Q85
General Purpose Controller	79,000	CMOS	1Q85
10,000 gate Gate Array	40,000	CMOS	1Q85
Enhanced Extended Arithmetic Unit	Not Avail.	CMOS	Deferred

A.2.6.2 General

The chipset performs 5 functions: 1) Complex number arithmetic vector processing, 2) Vector-scalar processing, 3) Floating point data processing, 4) Hierarchical multiprocessor system control and 5) Provides bulk memory.

A.2.6.3 Technology Insertion

The systems which have been targeted for the Westinghouse/National chipset are an airborne RADAR signal processor, an electro-optical signal processor for the M-1 tank, air defense RADAR fire control system, the avionics suite for the LHX helicopter, the USAF/Boeing E-3A RADAR, an advanced power management system for electronic warfare, a MIL-STD-1750A general purpose computer, a tactical air control center and an advanced programmable signal processor for retrofit into APG-68 RADAR used in the General Dynamics F-16.

A.2.6.4 The Chips

A.2.6.4.1 The Static RAM

The 64K-bit static RAM is configured as 8K words of 8 bits. It features a 25 nanosecond access/cycle time.

A.2.6.4.1.1 Built-In Test for the 8K by 8-bit Static RAM

No built-in test provisions are provided.

A.2.6.4.2 Pipeline Arithmetic Unit (PAU)

The PAU performs high speed vector-efficient operations. Some of the possible functions performed are FFT, recursive filtering, transversal filtering, matched filtering, convolutional filtering, spectrum shifting, weighting and band limited interpolation.

A.2.6.4.2.1 Built-In Test for the Pipelined Arithmetic Unit

The chip is divided into test cells of up to 5000 equivalent gates. Each test cell has test registers at all of its inputs and outputs. A special test bus (TBUS) is used to control testing. Simulation information was not available.

A.2.6.4.3 The Extended Arithmetic Unit Multiplier

The Extended Arithmetic Unit Multiplier provides high speed fixed and floating point multiply and divide operations. It uses MIL-STD-1750A formats and can also handle the Westinghouse 64-bit floating point format. Pipelining provides the capability of providing a result every 40 nanoseconds.

A.2.6.4.3.1 Built-In Test for the Pipelined Arithmetic Unit

The chip is divided into test cells of up to 5000 equivalent gates. Each test cell has test registers at all of its inputs and outputs. A special test bus (TBUS) is used to control testing. Simulation information was not available.

A.2.6.4.4 The General Purpose Controller

The General Purpose Controller is intended to be used as the microprogrammed control element within a signal processor, embedded processor or general purpose processor. It addresses up to 262K words (16 bits per word) of memory. It features a 40 nanosecond cycle (25 MHz) and has a design goal of 3 million instructions per second (MIPS), Digital Avionics Instruction Set (DAIS) mix, for a MIL-STD-1750A computer configuration.

A.2.6.4.4.1 Built-In Test for the General Purpose Controller

The chip is divided into test cells of up to 5000 equivalent gates. Each test cell has test registers at all of its inputs and outputs. A special test bus (TBUS) is used to control testing. Simulation information was not available.

A.2.6.4.5 The Gate Array

The gate array contains the functional equivalent of 7,904 2-input NAND gates. It has 160 bonding pads and 46 input, 38 output and 76 bidirectional buffers.

A.2.6.4.5.1 Built-In Test for the Gate Array

The entire gate array is treated as a single test cell. All of the inputs and outputs pass through test registers which may be controlled by the test circuitry. A special test bus (TBUS) is used to control testing.

A.2.6.4.6 Tha Enhanced Extended Arithmetic Unit

The Enhanced Extended Arithmetic Unit provides a 16-bit multiply with a 32-bit product, 32-bit floating point addition and subtraction in 3 clock cycles (including denormalization and normalization), and provides support for arithmetic operations on 32-bit integers and 48-bit floating point values. A 16-bit add/subtract is performed in 1 clock cycle. A 32 deep 16-bit stack is available. Features are available to facilitate double precision (64-bit) floating point, single and double precision complex operations and floating complex operations.

A.2.6.4.6.1 Built-In Test for the Enhanced Extended Arithmetic Unit

The chip is divided into test cells of up to 5000 equivalent gates. Each test cell has test registers at all of its inputs and outputs. A special test bus (TBUS) is used to control testing. Simulation information was not available.

A.2.6.5 System Test Procedures

Each chip in the chipset (except for the RAM) has an on-board BIT controller microcell which controls all testing within the chip. It operates on commands and data it receives over the test bus (TBUS). BIT controllers on separate chips within a system will test the connections and I/O buffers in paths between the chips.

The processor controlling the test bus depends upon the system design. It would appear that this system would have difficulty fully testing itself. A separate test processor may therefore be necessary.

A.3 CONCLUSIONS

The schedules for the VHSIC Phase I contractors indicate a deadline for the operational brassboards at the end of 1985. Delivery of 100 parts of each chip in the chipset are also due at that time. However, there are concerns that the yields will be so low that some parts may cost as much as $5,000. To combat this, the program office recently funded yield enhancement programs to each contractor. These programs cost a total of $15 million for a 32 month effort. This would seem to indicate that parts in reasonable quantities will not be available until mid 1987. However, all of the chip specifications should be stable enough for design purposes by mid 1985, and parts may be available in small quantities by the end of 1985. What does the VHSIC program buy us? The Hughes VHSIC digital correlator replaces a 32-stage, 5 MHz part which uses analog summing that has limited cascadability. This correlator also performs the magnitude calculations and threshold detection that previously were done off-chip. It offers five times the processing power and four times the speed of the part it replaces. It also may be configured for a wider variety of applications, including very long correlations. Four times faster at one fifth of the area means the part has 20 times the capability of current parts.

The VHSIC program has stressed making the parts general purpose. Various applications will not use all of the features on a chip. The end result of this is that some of the possible processing gain is lost. However, another goal of the program was to make general purpose VHSIC macrocells available which may be used in new chip designs. These chips may be made more special purpose and use only those cells which are needed. Chip design tools developed on the VHSIC program will simplify the design of those semicustom parts.

APPENDIX B

HAMMING CODES DESCRIPTION

Hamming codes are a class of generalized parity check codes usually used for single error correction. They can also provide a greater level of error detection than a standard parity check code and additional levels of error correction, at added hardware cost. Assume that there are q information bits which we want to protect with a Hamming code. In this case, c check bits must be added where c is determined by the condition that:

$$2^c > q + c + 1$$

The result is a [q + c] - bit word. The check bits are placed in the word so that error correction is easy to do. That is, if the [q + c] - bit word is given by:

$$b_{q+c} b_{q+c-1} \ldots b_3 \; b_2 \; b_1$$

then the c check bits are placed in the positions B_i, where $i = 2^j$ for j=0, 1, 2, . . ., c-1. The values of the bits are determined such that they insure that the parity sum equals 0 for error free words. There are two steps to defining the parity equations.

Step One: Integer Sets

Define P_j to be the collection of integers whose binary representation has:

a. c or fewer bits
b. The j-th bit equal to 1

Construct these sets for j=1 to c.

Step Two: Parity Equations

The parity sums are then defined by:

$$r_j = \Sigma b_k \quad \text{for } k \in P_j, \ k < q + c$$

where the c equations are given for j=1, . . ., c. The value of the check bit is selected so that the parity equation $r_j = 0$ is satisfied for j=1, ..., c. This gives a unique result since each of the parity sums contains exactly one of the check bits and no check bit appears in more than one parity sum.

EXAMPLE

If we are given 4 information bits, then q = 4 and c must be 3 so the code word contains 7 bits with the check bits at locations b_1, b_2, and b_4. The three integer sets are given by:

$$P_1 = \{1, 3, 5, 7\}$$

$$P_2 = \{2, 3, 6, 7\}$$

$$P_3 = \{4, 5, 6, 7\}$$

The three parity equations become:

1. $b_1 + b_3 + b_5 + b_7 = 0$
2. $b_2 + b_3 + b_6 + b_7 = 0$
3. $b_4 + b_5 + b_6 + b_7 = 0$

where + is mod 2 addition.

Now, to send the information bits 1 0 0 1, the three parity bits are selected so that equations 1-3 are true. Thus $b_1 = 0$ $b_2 = 0$ and $b_4 = 1$. The transmitted word is 1 0 0 1 1 0 0. Suppose an error occurs and the word becomes 1 0 0 1 1 0 1 (that is, b_1 becomes a 1). Evaluating the three parity sums in reverse order on the faulty word gives the result:

$r_3 = 1+0+0+1=0$

$r_2 = 0+1+0+1=0$

$r_1 = 1+1+0+1=1$

The binary output of the parity sum is 001 or decimal 1 which correctly identifies b_1 as the faulty bit.