

AD-A056 147

RESEARCH TRIANGLE INST. RESEARCH TRIANGLE PARK NC SYST--ETC F/G 14/2
A STUDY OF A STANDARD BIT CIRCUIT. (U)

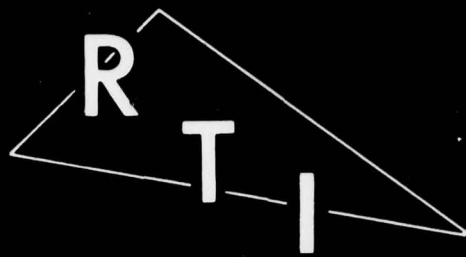
FEB 77 J B CLARY, J W GAULT, S J WEIKEL
RTI-43U-1269

N00163-76-C-0231
NL

UNCLASSIFIED

1 OF 3
AD
A056147





RESEARCH TRIANGLE INSTITUTE

FEBRUARY 1977

101

LEVEL

A STUDY OF A STANDARD BIT CIRCUIT

Final Report

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

Prepared for

Naval Avionics Facility
21st Street and Arlington Avenue
Indianapolis, Indiana 46218
Contract No. N00163-76-C-0231
RTI Project No. 43U1269

RESEARCH TRIANGLE PARK, NORTH CAROLINA 27709

AD A056147

AD No. _____
JDC FILE COPY.



78 07 07 049

LEVEL II

(10)

(6) A STUDY OF A STANDARD BIT CIRCUIT.

(9) Final Technical Report.

Contract NO 0163-76-C-0231

(15)

for

NAVAL AVIONICS FACILITY
Indianapolis, Indiana 46218

by

(10)
J. B. /Clary
J. W. /Gault
S. J. /Weikel
R. A. /Whisnant
R. D. /Alberts

DDC
RECEIVED
JUL 11 1978
F

(12) 239 P.

(14) RTI-43U-1269

(3) SYSTEMS INSTRUMENTATION DEPARTMENT
SYSTEMS AND MEASUREMENTS DIVISION

(1) RESEARCH TRIANGLE INSTITUTE

(2) RESEARCH TRIANGLE PARK, N. C. 27709
RTI Project Number-43U-1269

(11) February 1977

Reprinted June 1978

This document has been approved
for public release and sale; its
distribution is unlimited.

new
410 746

78 07 07 049

JOB

TABLE OF CONTENTS

	<u>Page</u>
LIST OF TABLES.....	vi
LIST OF FIGURES.....	vii
ACKNOWLEDGMENTS.....	x
1.0 INTRODUCTION.....	1
1.1 Scope.....	2
1.2 Technical Assumptions.....	3
1.3 Organization of the Investigation.....	4
1.4 Organization of this Report.....	5
2.0 RELEVANT BIT CONCEPTS.....	6
2.1 Taxonomy of Approaches.....	6
2.2 Passive BIT.....	7
2.3 Active BIT.....	9
2.4 Monitors.....	11
2.4.1 Replication.....	11
2.4.2 BIT Coding.....	11
2.4.3 Known Result.....	14
2.4.4 Emulation.....	14
2.4.5 Vital Sign Monitor.....	16
2.5 Meaning of Standardization.....	16
2.5.1 Standardization of BIT for QED Modules.....	19
2.5.2 Continuous On-Line Standard BIT Circuits.....	19
3.0 ANALYTIC MEASURES FOR COMPARISON AND EVALUATION OF BIT APPROACHES.....	21
3.1 Basic Assumptions.....	21
3.2 Goals.....	21
3.3 Cost Criteria.....	22
3.3.1 Time.....	22
3.3.2 Number of Packages.....	23
3.3.3 Power Consumption.....	23
3.3.4 Failure Rate.....	24
3.4 Performance Criteria.....	25
3.4.1 Percent of Packages Monitored.....	26
3.4.2 Percent of Gates Monitored.....	26

TABLE OF CONTENTS (Continued)

	<u>Page</u>
3.4.3 Other Possible Performance Measures.....	27
3.4.3.1 Function Coverage.....	28
3.4.3.2 Fault Coverage.....	28
3.4.3.3 Cycle Coverage.....	31
3.4.3.4 Composite Measure.....	33
4.0 RECOMMENDED BIT APPROACHES FOR QED MODULES AND APPLICATION OF THE ANALYTIC MEASURES.....	36
4.1 Built-In-Test for Memory Class Modules.....	37
4.1.1 Random Access Memories.....	37
4.1.1.1 Duplication.....	38
4.1.1.2 Parity.....	38
4.1.1.3 Single Bit Error Correction.....	42
4.1.1.4 Off-Line.....	42
4.1.1.5 Recommendations.....	44
4.1.2 Read Only Memories.....	48
4.1.3 First-In-First-Out Memory.....	53
4.1.4 Further Discussion of Built-In Error Correction Techniques for Memory Class Modules.....	55
4.1.5 Standard Interconnection and Interface BIT.....	60
4.1.5.1 Functional Description.....	62
4.1.5.2 Logic Diagram.....	65
4.1.5.3 Truth Tables.....	67
4.1.5.4 Circuit Implementation.....	68
4.1.5.5 Critical Parameters.....	68
4.1.5.6 QED Module Test Equipment Requirements....	68
4.1.6 Application of Analytic Measures to the Recommended BIT Approaches.....	70
4.2 Built-In-Test for Process Class Modules.....	73
4.2.1 Definition of Process Class.....	73
4.2.2 Approaches to BIT for Process Class.....	74
4.2.2.1 Arithmetic Coding Theory.....	74
4.2.2.2 Arithmetic Coding Techniques for Concurrent Error Detection.....	76
4.2.2.3 Non-Separate Codes.....	76
4.2.2.4 Separate Codes.....	79
4.2.3 Standardization of Process Class Module BIT.....	85
4.2.3.1 Functional Description.....	88
4.2.3.2 Logic Diagrams.....	92
4.2.3.3 Truth Tables.....	96
4.2.3.4 Circuit Implementation.....	96
4.2.3.5 Critical Parameters.....	103
4.2.3.6 QED Module Test Equipment Requirements....	103

TABLE OF CONTENTS (Continued)

	<u>Page</u>
4.2.4 Standard Timer.....	104
4.2.4.1 Functional Description.....	104
4.2.4.2 Logic Diagrams.....	107
4.2.4.3 Truth Tables.....	107
4.2.4.4 Circuit Implementation.....	112
4.2.4.5 Critical Parameters.....	112
4.2.4.6 QED Module Test Equipment Requirements....	114
4.2.5 Parallel Multiplier.....	114
4.2.6 Arithmetic Logic Unit.....	114
4.2.7 Eight-Bit Index Counter.....	116
4.2.8 Microprocessors.....	119
4.2.8.1 Additional Microprocessor BIT Techniques..	121
4.2.9 Application of Analytic Measures to the Recommended BIT Approaches.....	125
4.2.10 Process Class BIT by Partial Duplication.....	127
4.3 Built-In-Test for Control Class Modules.....	133
4.3.1 Programmable Timing Generator.....	133
4.3.2 Priority Encoder.....	134
4.3.3 Application of the Analytic Measures to the Recommended BIT Approaches.....	136
4.4 Built-In-Test for Interface Class Modules.....	139
4.4.1 Dual 8-Bit Switch.....	139
4.4.2 Dual Parallel 8-Bit Interface.....	140
4.4.3 Asynchronous Serial Interface.....	140
4.4.4 NTDS Input Buffer.....	144
4.4.5 NTDS Output Buffer.....	147
4.4.6 Application of Analytic Measures to the Recommended BIT Approaches.....	149
5.0 BIT NET GAIN EXAMPLE.....	153
5.1 Cascade Fast Fourier Transform (FFT) Processor Design.....	154
5.2 The Use of QED Modules In Cascade FFT Design.....	155
5.3 Subsystem Level Built-In-Test Design.....	162
5.4 FFT Processor Built-In-Test Evaluation.....	164
6.0 SUMMARY AND RECOMMENDATIONS.....	172
6.1 Summary of BIT Concepts and Approaches.....	172
6.2 Recommended Further Work.....	175
REFERENCES.....	178

ACCESSION for	White Section	☒	☐	☐
	Buff Section	☐	☐	☐
NTIS				
DDC				
UNANNOUNCED				
JUSTIFICATION				
BY				
DISTRIBUTION/AVAILABILITY CODES				
SP. CIAL				

TABLE OF CONTENTS (Continued)

Page

Appendix A

Reliability Data for Integrated Circuit Packages Used on QED Modules	A-1
---	-----

Appendix B

Performance Measure Tables for Memory Class and Interface Class Modules	B-1
--	-----

Appendix C

Discussion of a Standard Sample Monitor	C-1
---	-----

Appendix D

Bibliography	D-1
------------------------	-----

LIST OF TABLES

<u>Table Number</u>		<u>Page</u>
3.1	Example Cycle Coverage.....	34
4.1	Truth Table for SIIB.....	67
4.2	SIIB Implementation Alternatives.....	69
4.3	Analytic Measures Tabulation for Memory Class.....	71
4.4	Percent of Words Checked as a Function of the Check Modulus.....	80
4.5	Modulo-3 Results for Each Bit of an Eight Bit Byte Number.....	83
4.6	Truth Table for Two's Complement Residue Generation.....	99
4.7	Arithmetic Residue Computation Truth Table.....	100
4.8	Residue Checker Pass/Fail Truth Table.....	101
4.9	Residue Coding Hardware Implementation Alternatives.....	102
4.10	Truth Table for Standard Timer.....	110
4.11	Standard Timer Implementation Alternatives.....	113
4.12	Self Diagnosis Parameter.....	124
4.13	Process Class BIT Evaluation.....	128
4.14	Comparison of Residue Arithmetic and Partial Duplication	132
4.15	Analytic Measures Tabulation for Control Class.....	138
4.16	Failure Rate Data on NTDS Input Buffer BIT.....	146
4.17	Failure Rate Data on NTDS Output Buffer BIT.....	148
4.18	Analytic Measures Tabulation for Interface Class.....	152
5.1	FFT Arithmetic Unit Modules.....	159
5.2	FFT Memory/Address Modules.....	159
5.3	FFT Control Modules.....	162
5.4	Example FFT Processor Subsystem.....	167
5.5	System Availability With and Without BIT.....	169

LIST OF FIGURES

<u>Figure Number</u>		<u>Page</u>
2.1	Characterization of Module Level BIT.....	8
2.2	Decentralized Testing Isolation Requirements.....	10
2.3	Typical BIT Approaches Using Replication.....	12
2.4	Verification by Coding.....	13
2.5	Known Result Monitor With Partial Coverage.....	15
2.6	Sampled Monitoring.....	15
2.7	Levels of Standardization.....	17
3.1	Monitor With 100 Percent Input Coverage.....	29
3.2	Monitor With Partial Input Coverage.....	30
3.3	BIT Monitor With Partial Time Coverage.....	32
4.1	RAM BIT by Duplication.....	39
4.2	RAM BIT by Parity.....	40
4.3	RAM BIT With Error Correction.....	43
4.4	RAM Off-line BIT.....	45
4.5	RAM Module With Recommended BIT.....	46
4.6	Parity Implementation in RAM.....	49
4.7	Parity Implementation in ROM.....	49
4.8	ROM BIT by Longitudinal Parity.....	51
4.9	ROM Module With Recommended BIT.....	52
4.10	FIFO Module With Recommended BIT.....	54
4.11	Reliability of RAM Modules.....	58
4.12	Reliability Gain Using Error Correction.....	61
4.13	SIIB Functional Block Diagram.....	63
4.14	Input Checking.....	64
4.15	Output Checking and Parity Generation.....	64
4.16	SIIB Logic Diagram.....	66
4.17	Non-separate Arithmetic Code Implementation.....	78
4.18	Separate Arithmetic Code Implementation.....	81

LIST OF FIGURES (Continued)

<u>Figure Number</u>		<u>Page</u>
4.19	Modulo-3 Code Generation Algorithm.....	86
4.20	Modulo-3 Residue Code Generation Hardware.....	87
4.21	Residue Generator Functional Block Diagram.....	90
4.22	Residue Checker Functional Block Diagram.....	91
4.23	Even Power Bit Residue Generator.....	93
4.24	Odd Power Bit Residue Generator.....	94
4.25	Module-3 Adder Logic Diagram.....	95
4.26	Arithmetic Residue Computation Logic.....	97
4.27	Residue Checker Comparison and Timing Logic.....	98
4.28	Functional Block Diagram of Standard Timer.....	106
4.29	Standard Timer Control Logic.....	108
4.30	Programmable Counter Logic.....	109
4.31	Input/Output Timing of Standard Timer.....	111
4.32	Built-In-Test for Parallel Multiplier.....	115
4.33	ALU With BIT.....	117
4.34	Index Counter With Recommended BIT.....	118
4.35	Microprocessor Built-In-Test.....	120
4.36	8080A CPU Card Without BIT.....	123
4.37	Microprocessor With "Watchdog" Timer BIT.....	126
4.38	Arithmetic Logic Unit Partial Duplication BIT.....	129
4.39	Index Counter With Partial Duplication.....	130
4.40	Programmable Timing Generator With BIT.....	135
4.41	Priority Encoder With Recommended BIT.....	137
4.42	Dual Eight-Bit Switch With Recommended BIT.....	141
4.43	Dual Parallel Eight-Bit Interface With BIT.....	142
4.44	Asynchronous Serial Interface With BIT.....	143
4.45	NTDS Input Buffer With BIT.....	145
4.46	NTDS Output Buffer With Recommended BIT.....	150
5.1	FFT Algorithm. Eight Point Transform Pre-scrambled, Radix 2, Decimation in Time.....	156

LIST OF FIGURES (Continued)

<u>Figure Number</u>		<u>Page</u>
5.2	FFT Elemental Computation Unit (Butterfly).....	157
5.3	Cascade FFT (Radix-2, Decimation in Time).....	158
5.4	Time-shared Cascade FFT Processor.....	160
5.5	Basic FFT Computational Unit Organization.....	161
5.6	FFT Built-In-Test Monitor Interface.....	163
5.7	System Availability Versus System Complexity With and Without BIT.....	170

ACKNOWLEDGMENT

This report was prepared by the Research Triangle Institute, Research Triangle Park, North Carolina, for the Naval Avionics Facility, Indianapolis (NAFI) under Contract N00163 76-C-0231. The work was administered by Mr. Gene Bradley of NAFI with technical coordination provided by Mr. Bill Dejka of the Navy Electronics Laboratory Center (NELC), San Diego, California.

This work has been accomplished under Subtasks 1 through 5 of the statement of work. Dr. R. A. Whisnant served as RTI Project Leader with programmatic guidance provided by Mr. R. D. Alberts. Dr. J. W. Gault of the Electrical Engineering Department, North Carolina State University served as consultant on this project.

1.0 INTRODUCTION

The desirability of reducing life cycle costs of complex digital systems is obvious. The mechanisms for accomplishing this reduction and the approach(es) to seeking these mechanisms are not so obvious. One approach which has been successfully taken by the Navy is the utilization of standard modular hardware to minimize equipment sparing costs by reducing the number and types of modules spared. In addition, the use of modular digital hardware on mobile weapon platforms having long duration missions precludes access to large spare equipment inventories (e.g., ballistic missile submarines). Utilization of modular digital hardware in such cases allows equipment repair to be promptly effected and thereby increases system availability. Thus the Navy standard modular hardware program such as the Standard Electronic Module (SEM) program lead directly to life cycle cost reduction and increased system availability.

The overall objective of this work is to explore ways to further reduce digital system life cycle cost and increase system availability through improved fault detection and isolation techniques. The particular approach taken in this study involves the use of built-in-test (BIT) circuits at the replaceable unit level to facilitate fault detection and isolation. Special emphasis is given to on-line, continuous BIT approaches which are particularly appropriate for non-redundant (i.e., single string) systems with high availability requirements such as communications systems, conventional weapon fire control, and surveillance radar signal processing. In these instances off-line fault diagnostic approaches which are possible with redundant systems cannot be used because of the requirement for continuous system availability.

A further objective of this study is to provide modular system designers with circuit modules that have integral built-in-tests which result in maximum error detectability with minimum impact on the system design itself. The increased power, thermal conditioning, space and interconnections required by the BIT circuits should not limit the system

designer's capabilities and options in realizing the functions necessitated by system performance requirements.

Finally, it is an objective of this study to consider module built-in-test alternatives which are, in some sense, "standard". That is, it is a goal to design and specify as few unique BIT circuits as possible to be used in multiple ways to provide on-line, continuous, non-interfacing fault monitoring at the replaceable module level. A discussion of the issues relevant to achieving non-interfering continuous, fault monitoring with standard built-in-test techniques is included in this report.

1.1 Scope

The scope of this work, within the broader objectives given above, can be indicated by a series of questions. Considering a functional module family

- (1) What approaches can be taken to provide module level BIT,
- (2) What is the cost involved, and
- (3) How effective is the resultant fault detection?

These questions form an initial sense of direction and work statement. Later in this report, by looking a little deeper at the complexities, additional objectives and more explicit concerns will be defined. The strongest immediate concern in view of the contract requirements, is the idea of standardization.

- (4) Can the BIT circuit approaches be both standard and universally applicable to all functional digital hardware modules?

The following section outlines the assumptions, approach and progress of work relating to the pursuit of the answers to these four questions.

1.2 Technical Assumptions

It is essential to establish the explicit ground rules which define the testing environment and beginning point for this report. This work will focus on those issues and concepts which are relevant to all functional digital circuit modules. However, the answers to the four questions posed earlier are initially sought specifically for the Quick and Easy Design (QED) modules [1], [2].

The environment in which BIT circuits of interest will be examined consists of systems configured from QED modules. These systems will be assumed to have passed a suitable acceptance test. A suitable acceptance test is defined as a test which removes

- (1) Design errors,
- (2) Manufacturing errors,
- (3) Software errors, and
- (4) Which verifies all performance specifications.

Understanding the status of a system prior to the maintenance task helps to establish the type of faults which are to be considered. The fault model describing the faults which the BIT will be expected to detect is the solid fault(s) which produces results (data or status) differing from the desired results. Intermittent faults, if they are detected, will be indicated to the system as hard failures. The fact that the fault model does not include intermittents implies that no attempt will be made to detect intermittents specifically.

For the test environment and fault model as defined, the test objective is to

- (1) Detect failures by monitoring performance with BIT circuits,
- (2) Diagnose the failure to a QED module, and
- (3) Remove the fault manually with the aid of a visual cue.

The BIT circuit defined to meet the above criteria might be conceived from either a short-term, mid-term, or long-term point of view with respect to packaging constraints. The near-term demands that the proposed circuit fit within the package and pin availability of the existing QED hardware implementations. A mid-term view would allow the use of currently available devices and repackaging of the present implementation to allow for any necessary changes due to the inclusion of the BIT hardware. The long-range view carries with it the additional possibility of utilizing LSI technology. The long-range view allows at least two further options. First, the BIT, if it can be expected to be widely utilized, can itself be defined as an LSI package(s). This would provide for some projected gain in both lower cost and high reliability of the BIT circuits. The second option would be to consider the BIT as part of the same LSI package which might contain the QED module. This later option may yield the greatest gain in both lower added cost and reliability.

The primary objective of the work reported here is to investigate and define feasible BIT approaches in the context of the mid-term or long-term views given above. For that reason package technology and pin availability are not taken as binding constraints. There is, however, every effort made to meet the guidelines defined by the QED maintenance plan presented in [3].

1.3 Organization of the Investigation

This study is organized to accomplish the contract objectives through a simultaneous, two-part approach to the functional module built-in-test problem. The first part considers module BIT from an overview or "top-down" standpoint. This effort emphasizes problem definition in conjunction with BIT cost and performance measurement. In this portion of the study special attention is given to defining areas of life-cycle cost which can be reduced through improved fault detection techniques. Closely akin to this effort is the definition and quantification of appropriate measures of BIT effectiveness, cost and the corresponding reliability impact.

The second part of the investigation considers specific approaches to BIT for the QED modules. This "bottom-up" analysis of functional module built-in-test alternatives effectively illuminates the real-world problems

of fault detection and localization at the module level. An important aspect of this part of the study is an enhanced understanding of the BIT circuit standardization problem. The cost and effectiveness of the BIT circuits recommended for the QED modules are evaluated through the use of an example subsystem.

1.4 Organization of this Report

The intent of Section 1.0 is to set the stage for all that follows. This is done by establishing the scope and assumptions underlying the study in Section 1.1 and Section 1.2 and by identifying the goals, objectives, and constraints.

Section 2.0 presents some material which is important in a general and conceptual sense. The taxonomy of approaches to BIT given in Section 2.1 provides a framework for the specific BIT approaches to the QED modules presented in Section 4.0. The analytic measures discussed in Section 3.0 are applicable across the range of BIT approaches and modules. In Section 2.5 specific approaches to standardization of BIT are discussed as general concepts. The presentation of these concepts helps to clarify some of the rationale applied in deriving the detailed circuits described later.

Section 4.0 focuses upon the results of investigating BIT approaches for the specific QED modules. Each QED module is discussed in this section and one appropriate BIT approach is recommended and evaluated using the criteria presented in Section 3.0.

Section 5.0 describes the application of the recommended module BIT approaches to an example system. In particular, a non-recursive digital filter in the form of a fast Fourier transform (FFT) processor is used to illustrate the reduction in Mean Time to Repair (MTTR) possible with module level BIT. Included in this discussion is a comparison of system mean time to repair with and without the recommended BIT.

Section 6.0 summarizes the results of the study including the recommended BIT approaches and measures of effectiveness. The report concludes with recommendations for further work in built-in-test.

2.0 RELEVANT BIT CONCEPTS

Prior to reporting specific results there are some relevant issues which are general in nature and germane to the built-in-test considerations at hand. It is important to characterize and categorize the BIT approaches in breadth prior to becoming too involved in the details of specific approaches. This is done in the following subsections.

One of the most difficult aspects of specifying one BIT approach over another is the quantification of its effectiveness. The original task description for this study defines a set of measures which may be used to quantify BIT effectiveness. Section 3.4.3 outlines some other suggested measures which may be considered in evaluating BIT. Finally, since one of the four primary questions to which an answer is sought has to do with standardization, Section 2.5 deals with some of the potential pitfalls and contradictions particular to standardization which should be considered.

2.1 Taxonomy of Approaches

There are a number of BIT approaches which adhere to the assumptions of this work as outlined in Section 1.0. The purpose of classifying and characterizing BIT approaches in a general way is to give visibility to those approaches not considered in detail. In pursuing alternate BIT approaches, there are a number of properties and attributes to which the classification might be tied.

In the past, two primary testing objectives have been identified. They are detection and diagnosis. Since, for the purposes of this work the circuit is on a module and diagnosis to the module level is desired, there is almost a direct correspondence between the level of detection and the desired level of diagnosis. Therefore, very little distinction is drawn between detection and diagnosis in this study.

Another general characteristic of testing circuits is the ability to control the unit under test including test value insertion, monitoring of outputs, and verification of the actual results against the correct results. As a matter of convenience, monitoring and verification are usually considered as part of the same capability. These attributes are often referred to as controllability and observability. In this context, controllability implies an active capability, while observability (monitoring) is passive. These attributes provide the first dichotomy in categorizing

BIT approaches (Figure 2.1). Notice that the definition of active versus passive BIT is related to other definitions frequently found in the literature of on-line versus off-line BIT.

An on-line test is any test of normal operational circuits which is concurrent with, and which at no time preempts or degrades, operation of the tested circuit. Continuous on-line monitoring is referred to throughout this report as concurrent fault monitoring. An off-line test is any test of operational circuit which preempts or interferes with normal circuit operation.

The following section presents detailed active versus passive BIT considerations.

2.2 Passive BIT

The passive BIT circuit is strictly a monitor. For purposes of this discussion, the level of monitoring is taken to be at or within the QED module. There are three important characteristics demonstrated by a monitor. The first of these is the ability to validate output results from input patterns. This property or characteristic will be referred to as input coverage. Input coverage is defined as the ratio of the number of input patterns whose outputs can be validated divided by the total number of input patterns possible. For a more detailed description and utilization of the idea of input coverage, see Section 3.4.3.

A second important characteristic of a passive BIT is its function coverage. The function coverage characteristic of passive BIT provides a measure of the number of module functions which are monitored. Function coverage is defined as the ratio of the number of functions monitored by the BIT divided by the total number of functions in the module. This idea is expanded and used to define parameters in Section 3.4.3.

The third passive BIT characteristic, which is considered less often in the literature, is the idea of cycle coverage. Cycle coverage is defined as the number of module cycles which are monitored divided by the total number of module cycles possible. Cycle coverage is also discussed in more detail in Section 3.4.3.

One can see then, that from these three characteristics, the basic idea of a monitoring capability can be defined. With respect to the

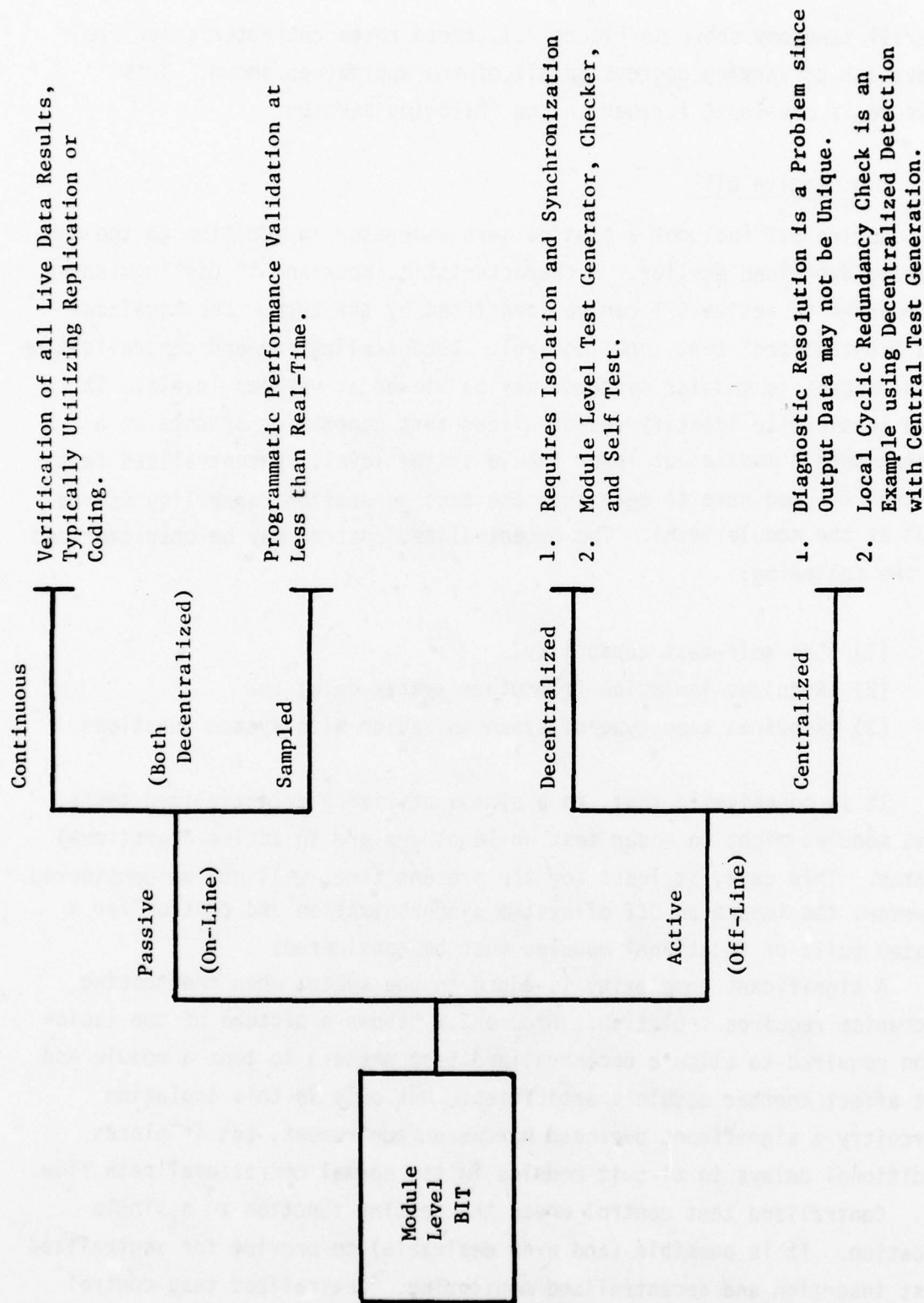


Figure 2.1 Characterization of Module Level BIT.

overall taxonomy shown in Figure 2.1, these three characteristics are prevalent to varying degrees in all of the approaches shown. This measure is developed further in the following section.

2.3 Active BIT

Active BIT includes a test pattern generator in addition to the monitoring described earlier. A characteristic important in distinguishing approaches to active BIT can be identified by the terms "decentralized" and "centralized" test input control. Decentralization and centralization, with respect to modular systems, may be viewed at various levels. It is also possible to identify decentralized test generation schemes at a higher than a module but lower than a system level. Decentralized test control is used here to mean that the test generation capability is present at the module level. The decentralized control may be characterized by the following:

- (1) Has self-test capability;
- (2) Requires isolation from other system data; and
- (3) Requires some type of synchronization with system functions.

It is conceivable that, in a system utilizing decentralized tests, some modules might be under test while others are in active operational status. This case, at least for the present time, will not be considered. However, the impact on BIT of system synchronization and control for a system built of functional modules must be considered.

A significant complexity is added to the system when the testing mechanism requires isolation. Figure 2.2 shows a picture of the isolation required to allow a decentralized test pattern to test a module and not affect another module's activities. Not only is this isolation circuitry a significant overhead hardware requirement, but it places additional delays in circuit modules in the normal operational path flow.

Centralized test control moves the testing function to a single location. It is possible (and even desirable) to provide for centralized test insertion and decentralized monitoring. Centralized test control may be typified by the fact that the pass/fail signals from the module monitors are examined in a central facility.

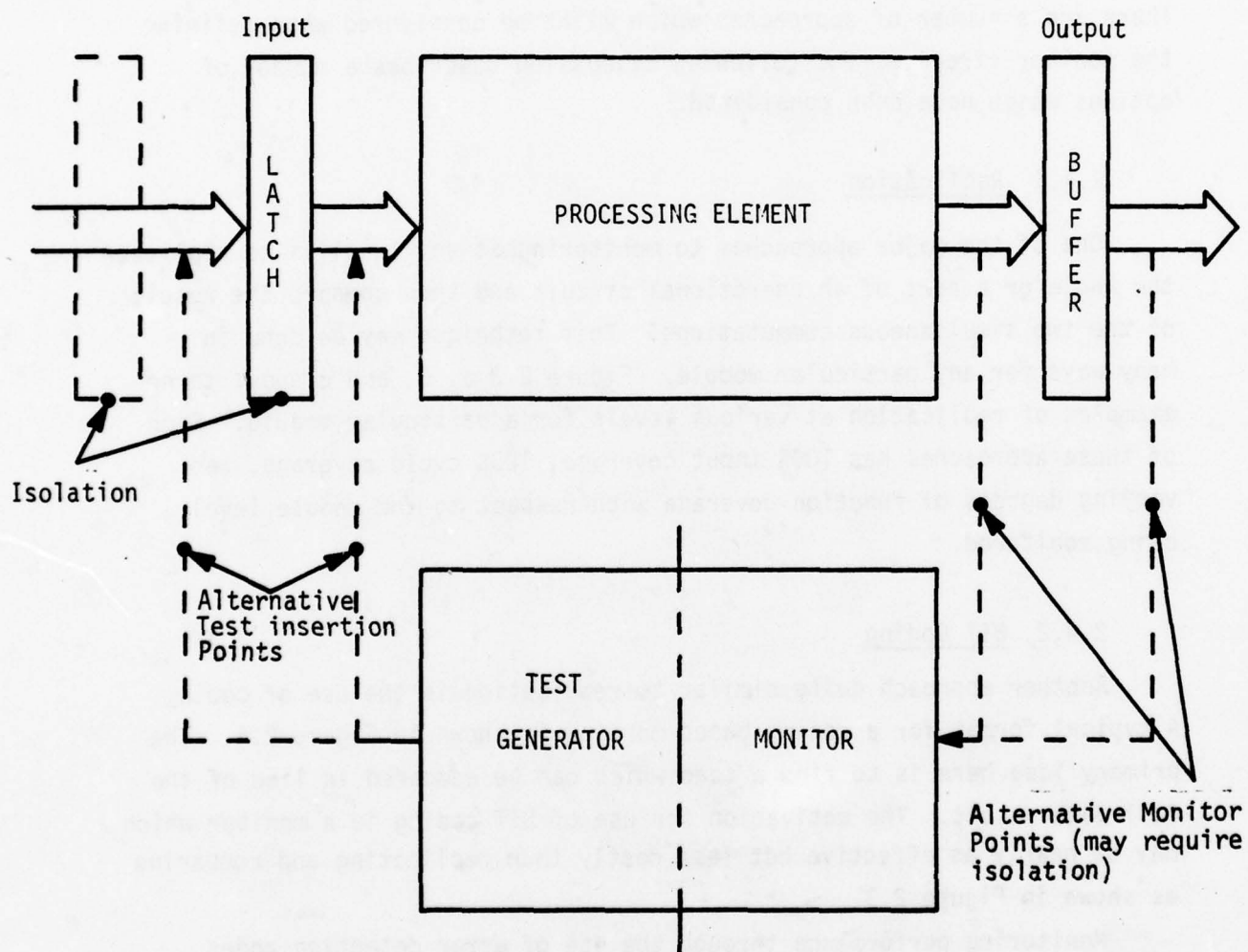


Figure 2.2 Decentralized Testing Isolation Requirement

The attribute that distinguishes active BIT from passive BIT is, in fact, simply a test generator. It is a non-trivial task to determine a meaningful test set (i.e., a set of input test vectors) for each module in a system. However, once that test set has been determined, it is generally not particularly difficult to develop hardware to present the test set as input conditions.

2.4 Monitors

The common aspect of passive and active BIT is the monitor circuit. There are a number of approaches which might be considered when defining the monitor circuit. The following discussion describes a number of options which have been considered.

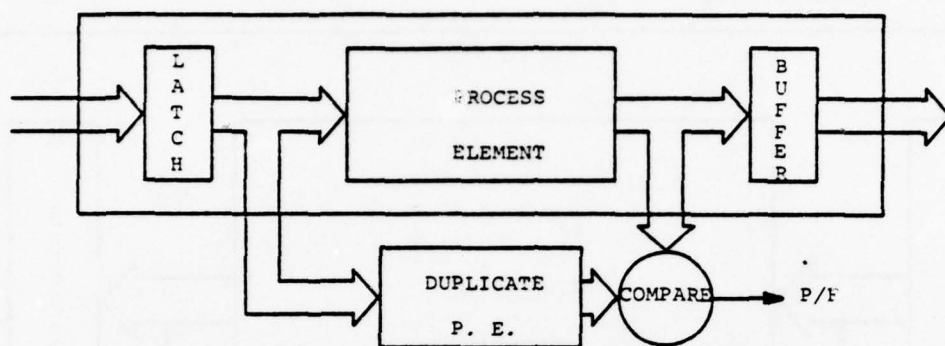
2.4.1 Replication

One of the major approaches to monitoring at any level is to replicate the whole or a part of an operational circuit and then compare the result of the two simultaneous computations. This technique may be done in many ways for any particular module. Figure 2.3 a, b, and c shows three examples of replication at various levels for a particular module. Each of these approaches has 100% input coverage, 100% cycle coverage, and varying degrees of function coverage with respect to the module level being monitored.

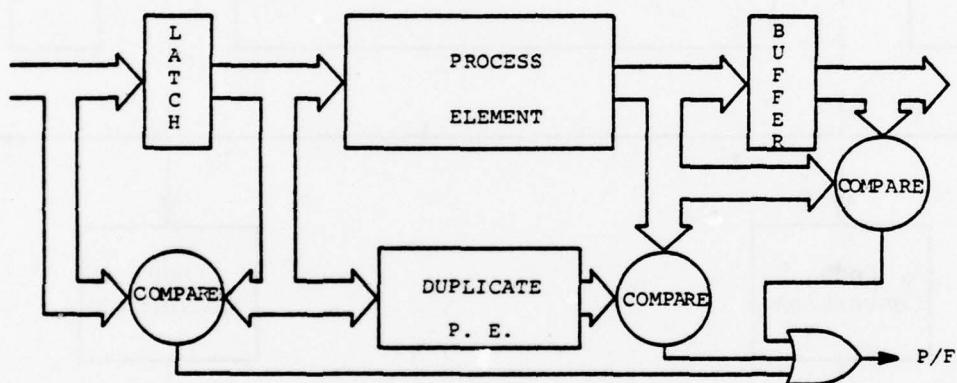
2.4.2 BIT Coding

Another approach quite similar to replication is the use of coding. A typical format for a coding based monitor is shown in Figure 2.4. The primary idea here is to find a code which can be compared in lieu of the full data result. The motivation for use of BIT coding is a monitor which may be nearly as effective but less costly than replicating and comparing as shown in Figure 2.3.

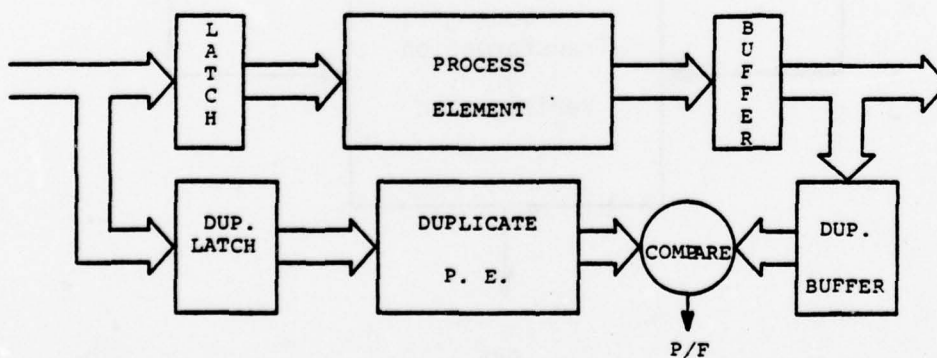
Monitoring performance through the use of error detecting codes introduces a subtle distinction not adequately characterized by the three ideas of input, function and cycle coverage. This can be understood by thinking of parity as a code checking device. While 100% of the input patterns may be monitored for 100% of the time, it is clear in the case



a. Partial Replication



b. Complete Distributed Monitor



c. Duplicate Module and Compare

Figure 2.3 Typical BIT Approaches Using Replication

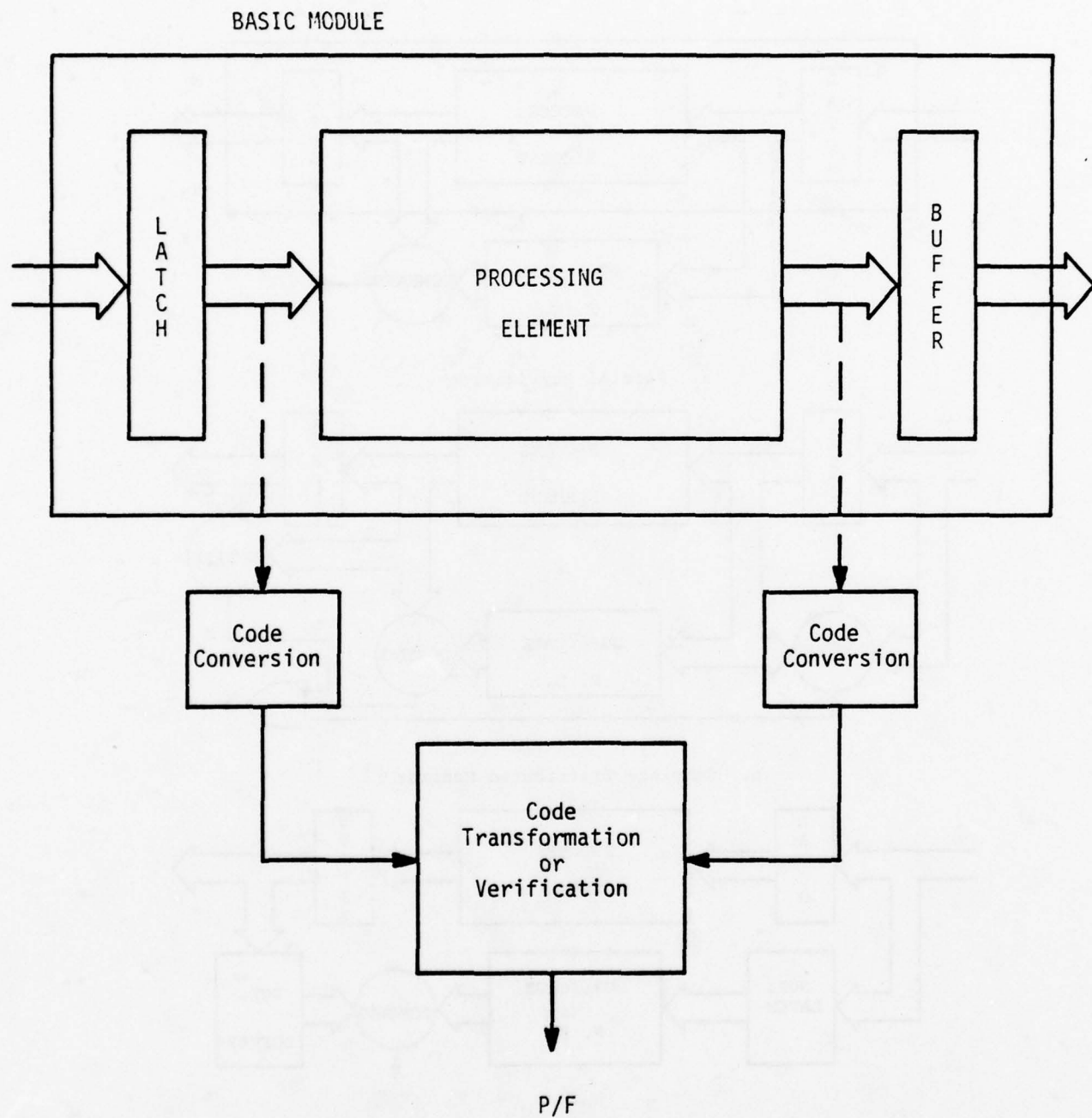


Figure 2.4. Verification by Coding

of parity that one would be reluctant to say that there was 100% monitoring of the function fault behavior. The idea is that there are certain errors which will in fact disrupt normal results but will not appear in the coded form.

2.4.3 Known Result

The preceding two monitoring approaches are characterized by the fact that the desired answers for particular inputs are not known a priori. This places a computational demand upon the monitor to determine a correct answer to be used as a basis of comparison. It is possible to do monitoring of a simpler form when the desired answer can be established ahead of time. Since this case is more restrictive, one might suspect the effectiveness is somewhat reduced. The motivation for pursuing this idea is based on the expectation that the cost of such a monitor would be significantly lower and its applicability more universal than the replication or coding approaches discussed previously.

An example of a known result approach is shown in Figure 2.5. Since the total range of possible computations is large, there is motivation to provide fewer stored results than possible computed results. In this example the monitor checks for special cases of input occurrence, and, rather than compute a result to use as a comparison, a stored result is used. With a careful selection of input patterns, significant checking of the circuit performance can be accomplished.

As an aside, it should be noted that the effectiveness of this approach depends in large part upon the arrival of those special input cases at the module. In the case of a passive monitor, no control over the input set may be exerted. Therefore, the overall effectiveness of the known result monitor approach is a function of the data presented to the module under named system operation.

2.4.4 Emulation

To extend the idea of partial coverage, consider Figure 2.6. In this particular approach the cycle coverage is reduced from 100%. The basic operation of the monitor is to sample with particular attention given to the timing relationships between the input and the output. The sample

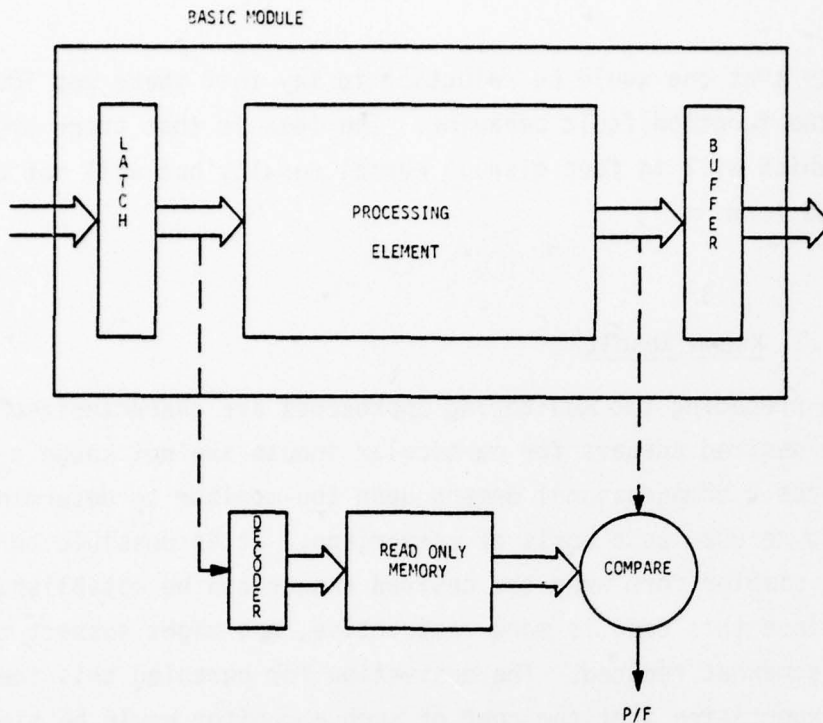


Figure 2.5 Known Result Monitor with Partial Coverage

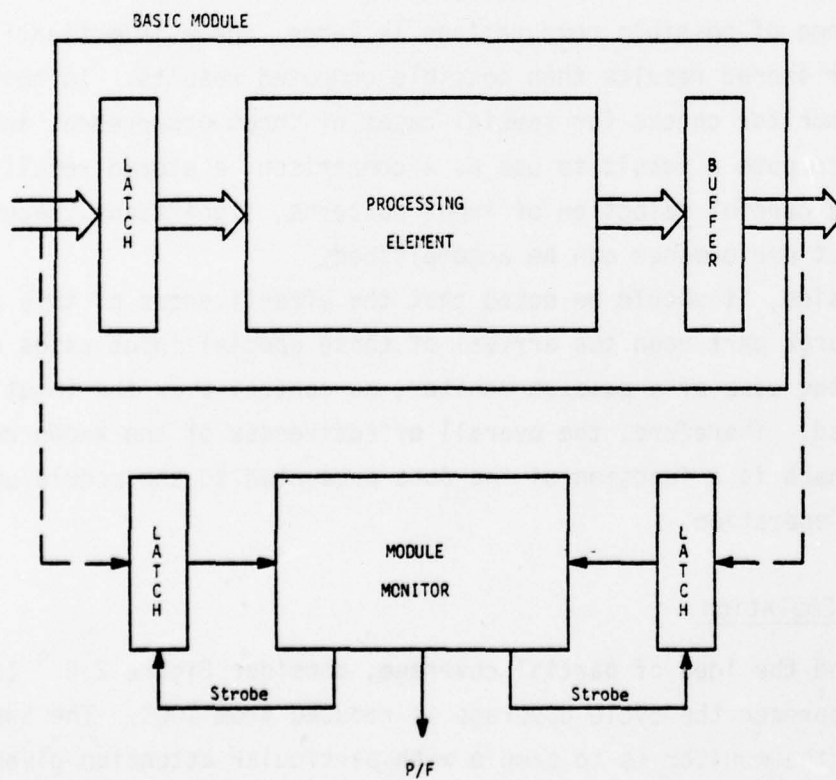


Figure 2.6 Sampled Monitoring

monitor then computes, based on the input, its version of the output, and compares that to the sample output to determine the status of the network.

The primary motivation for considering this approach to passive BIT is that this particular task (that is, sampling and computation), is well suited to a programmatic passive monitor. A programmatic passive monitor could be utilized in a standard configuration to monitor a wide variety of processing elements or module functions.

Although an increase in flexibility of application is gained from use of programmatic devices, there will often be an attendant reduction in speed. Therefore, the concept of sampling becomes imperative since the programmatic monitor cannot, in general, perform computations in the same time frame as the operational modules in question.

The difficulty encountered in both the known result and the sampled monitoring approaches is that it becomes somewhat more difficult to define the effectiveness of this type of monitoring. This problem is addressed in the extensive discussion of BIT measures given in Section 3.0.

2.4.5 Vital Sign Monitor

One last possibility which is mentioned here is the idea of monitoring of vital signs. A vital sign monitor observes a fairly restricted portion of a circuit for presence or absence of a desired behavior. Two examples are, power level monitoring and basic oscillator activity. It is conceivable that this approach might be formulated and perceived as falling into the above categories. However, it is mentioned here to give added visibility to the range of possibilities when considering BIT approaches.

2.5 Meanings of Standardization

The objective of this section is to examine the interpretations which may be applied to the meaning of standardization. The intent of standardizing BIT circuits is to establish test hardware which can be used routinely to provide a certain level of testing confidence, that is, to establish BIT capability as part of the normal design procedure.

The idea of applying this definition at various levels is shown in Figure 2.7. Standard BIT hardware at various levels of complexity is

analogous to the problem of standardizing electronic modules. The QED modules represent one choice as to the level of complexity which is desirable for standard modules.

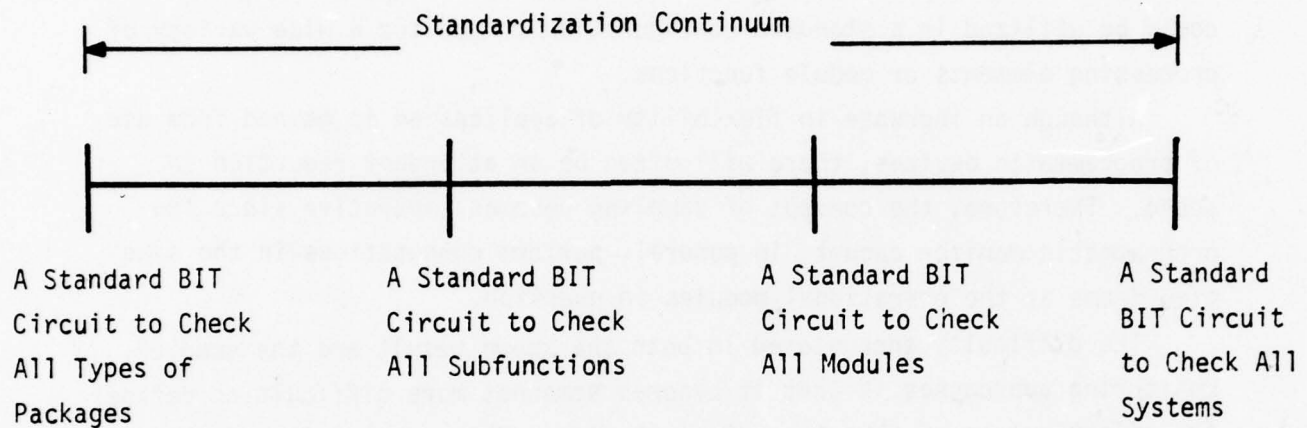


Figure 2.7 Levels of Standardization

Now consider what standardizing BIT means with respect to QED modules or other functional modules with similar attributes. The far right represents the Utopian approach valid for all systems. Just right of center of the spectrum portrayed in Figure 2.7 resides the idea of a single BIT circuit (the standard) which is suitable for use with all modules. Farther to the left is the identification of a lower level BIT circuit designed to be used in conjunction with the occurrence of functions which are sub-elements of a module family and finally BIT standards associated with packages. The idea of a single BIT circuit which can be personalized to a specific task is sufficient but not

necessary to the intent of standardization. The existence of a general purpose BIT circuit at the module level is far more likely than at the package level because of the inherent overhead required by generality. The point can be made then, that the level and number (hopefully small) of BIT circuits do not of themselves violate the precept of standardization; what is fundamental is that the rules for the utilization of these BIT circuits in a design be procedural. A corollary to this is that if the BIT procedures are followed a certain level of testing and fault detection will be achieved.

Let us examine for a moment the projected utilization of the BIT independent of the level of solution. In the mid-term (as defined in Section 1.2) the selected BIT circuits can be implemented with conventional packages and the QED cards reconfigured to include the added hardware. In the long term there are two possibilities which may be appropriate. One possibility is that the BIT circuit(s) could be defined as a MSI or LSI package(s). These special packages would then be incorporated into the QED module implementation. This approach would require fewer chips and would have better failure rate characteristics than an implementation using conventional packages.

A second long term view would include the BIT circuits as part of the functional module definitions and would realize the entire unit as an LSI package. The range of interpretations of standard taken earlier can now be examined in light of the projected mid or long-term utilization envisioned.

The use of standard to mean one circuit as opposed to several circuits for an entire module family is of primary importance only if the intent is to define MSI or LSI packages explicitly for BIT functions. Then perhaps there is an advantage to defining fewer new packages. Even then if the anticipated demand for BIT packs is large enough, there is little distinction between one and several distinct types.

When the BIT level is at the module or lower as it is here, then if there is an on-going effort to specify and design modules, it probably is desirable to have fewer BIT circuits with which a designer must deal. This is analogous to saying that designing with TTL would be easier if there were fewer types of packages. From this analogy, it can be seen that there is some balance between the simplicity of fewer types and

the capability and richness of wide selection. If few will do the job, then few are desirable but if and only if those few do the job.

The net outcome of this argument is to conclude that the identification of a standard BIT circuit does not offer much reward over providing a fairly small number of BIT circuits for a range of functions, with perhaps a handbook to define their preferred utilization.

The specification of BIT approaches for the QED family presumably is a more or less one-time task. The identification of BIT approaches for other sets of modules can then be logically extended from these specific examples.

2.5.1 Standardization of BIT for QED Modules

The possible interpretations of the meaning of standard were described in the preceding section. The purpose of this section is to summarize and make as explicit as possible the meaning of standardized BIT as it applies to the QED modules.

The explicit interpretation which is applied to standardizing BIT is very strongly determined by the on-line versus off-line nature of the BIT approach. This section is organized around the three major classification categories defined in 2.1; i.e., continuous on-line, sampled on-line, and off-line. The continuous on-line standard approaches are at a lower level functionally than the others. Specific contributions to this category will continue to be evolved as part of this study.

2.5.2 Continuous On-Line Standard BIT Circuits

No single standard module level BIT circuit has been found which can provide continuous on-line monitoring for a family of functional modules with propagation times in the range typified by the QED set. In order for a single standard to be suitable for an entire family, it must be capable of being personalized. The only practical way to provide a wide variance (as in a functional family) of changeability is through programmatic techniques. The major conflict which this creates is the resultant speed discrepancy between the operational hardware module and the programmatic monitor.

It can be seen that, if there existed a general device which could be programmed to adequately monitor a family of modules at full operating speed then, this general monitoring device would be a candidate to replace the entire module family with one standard device. The conclusion is then to argue by contradiction that no such device exists, and hence emphasis should be placed on identifying a functionally lower level set of circuits to perform BIT within modules.

A possible sample BIT monitor realization that comes to mind is a microprocessor. In order to fully understand the ramifications of this approach for module level built-in-test, a study has been conducted in which a candidate microprocessor sample monitor design is applied to a particular QED module. The results of this study are presented in Appendix C.

3.0 ANALYTIC MEASURES FOR COMPARISON AND EVALUATION OF BIT APPROACHES

The basic intent of the built-in-test circuits of concern in this report is to provide fault detection and diagnosis to a module level with a visual cue and/or a pass/fail logic indication to a system built-in-test equipment monitor. There is no automatic repair or reconfiguration intended at the present time.

The maintenance concept for QED modules identifies the major guidelines under which BIT must function and gives insight into the strategic decisions made in formulating an overall maintenance plan [3]. The purpose of this section is to give or derive some of the analytic measures suitable for making decisions as to the specific approach for BIT to be taken within the framework of this overall maintenance plan.

3.1 Basic Assumptions

The BIT circuitry envisioned here is anticipated to exist as part of the QED or other family of functional digital modules. It is intended to monitor and validate module operations and provide a go/no-go indication. The level of diagnosis in this situation is synonymous with the detection level. The case of module-level BIT which has off-line test insertion capability is considered to be unlikely for reasons discussed in the section concerning active BIT.

Module performance validation will focus on the detection of functional faults as opposed to electrical or timing faults. This basically identifies logic faults (gate behavior changes) which alter the output from a desired value. Transients will not be filtered at the module level. If a transient causes an error which is detected, then the no-go indication will be given.

3.2 Goals

For the type of monitoring circuits being considered the designer must be able to evaluate both the cost and effectiveness of various BIT schemes. The cost measures, involving time, space, power, and reliability

as defined for this report, are given in Section 3.3.

An important aspect of the analysis comes in trying to quantify the fault detection performance of a particular BIT approach. The primary goal of Section 3.4 is to define the measures used to evaluate the effectiveness of the recommended BIT techniques.

Section 3.4.3 suggests other performance measures that give additional information about the completeness of the BIT fault monitoring. These measures are included because it is apparent that the present performance criteria are not totally adequate for determining the overall effectiveness of on-line, concurrent fault monitoring techniques.

3.3 Cost Criteria

If a broad interpretation is applied to cost, then there are a significant number of parameters which might be considered. This report focuses on cost parameters in four basic areas: 1) time, 2) number of packages, 3) power consumption, and 4) reliability. Each parameter in each of the basic areas will be defined in the following subsections.

3.3.1 Time

The time required to apply an n-cycle off-line test is a measure of the amount of time the module is used for self-testing and is not available to do normal processing. This parameter is defined as follows:

$$\begin{aligned} \text{Number of Cycles for Test} &= \text{The number} \\ &\text{of clock periods necessary to perform an} \\ &\text{off-line module self-test.} \end{aligned} \quad (3.1)$$

The actual cost in lost processing time is dependent on the specific application, since in many systems there are times when a module is idle. This time can be used for module self-testing with no reduction in overall processing capability. However, except for the microprocessor module, none of the BIT techniques recommended in this report involve an off-line test method; therefore the number of cycles for test for all modules

except the microprocessor will be zero.

3.3.2 Number of Packages

The number of integrated circuit packages required to implement the recommended BIT approach must be determined. The basic idea of this parameter is to describe the portion of the module that is identifiable as BIT circuitry. This measure, expressed as a percentage is defined as

Ratio of BIT Packages to Total Module Packages =

$$\frac{100 \cdot [\text{Number of Packages in BIT}]}{\text{Total Number of Packages (including BIT)}} \quad (3.2)$$

The strongest tie to actual cost in terms of dollars and space is given by package count. The final cost of a board assembly is usually less sensitive to the cost of a package than it is to the cost of placing the package on the board. Size is also somewhat more related to packages than gates or other circuit complexity measures.

One can develop a scaled package count with the pins per package as a scale factor. This approach holds board space as the most important aspect of this measure, although larger packages do tend to cost more. While this is a defensible argument, it seems that, for the added difficulty, little additional information is gained.

3.3.3 Power Consumption

The determination of the power consumption of the BIT circuitry can be treated in a straightforward manner. Even at the functional design stage, a fair estimate may be made as to the explicit packages required for a particular BIT approach. Once this is done, the typical DC power requirements for the required packages may be summed to obtain the total BIT power requirements. This leads to the following definition

$$\text{Power Consumption of BIT} = \sum_i PB_i \quad (3.3)$$

where the sum is taken over all of the packages in BIT and PB_i is the typical power consumption (product of typical current and typical voltage supply) for the i -th BIT package.

Another cost measure was defined to give an indication of the proportion of the power consumption of BIT relative to the whole module. This parameter, expressed as a percentage, is defined below

Ratio of BIT Power Consumption to Total Module
Power Consumption =

$$\frac{100 \cdot \sum_i PB_i}{\sum_j PM_j + \sum_i PB_i} \quad (3.4)$$

where PB_i is the typical power consumption of the i -th BIT package and $\sum_i$ is the sum over all of the BIT packages. PM_j is the typical power consumption of the j -th module package (non-BIT), and $\sum_j$ is the sum over all the non-BIT module packages.

3.3.4 Failure Rate

The addition of hardware to affect fault detection without repair adversely affects the reliability (MTBF and failure rate) of the module. This is tolerable only if the added circuitry improves the repair time and hence system availability. The purpose of this measure is to indicate the impact of the BIT on the MTBF or failure rate. It is much more difficult to quantify, without extensive system operational field data, the resultant mean-time to repair (MTTR) improvement.

The failure rates (FR) for individual packages found in QED modules are summarized in Appendix A. The failure rate for each QED module has been derived using the sum of component failure rates and is given in reference [4]. This simplification, which judges packaging (primarily connector) failures to be small with respect to component failures, can be utilized for the BIT circuit calculations.

The failure rate of the BIT circuit is the sum of the failure rates of the BIT packages. The measures used to express the failure rate information are defined as

$$\text{Failure Rate without BIT} = \sum_j \text{FRM}_j, \quad (3.5)$$

$$\text{Failure Rate with BIT} = \sum_j \text{FRM}_j + \sum_i \text{FRB}_i, \quad (3.6)$$

where FRM_j is the failure rate of the j -th module package or discrete component (non-BIT), and $\sum_j$ is the sum over all of the non-BIT packages and discrete components. FRB_i is the failure rate of the i -th BIT package or component, and $\sum_i$ is the sum over all of the BIT packages and components.

To indicate explicitly the relative proportion of the BIT failure rate to the failure rate of the whole module, a new cost measure was defined. This parameter, expressed as a percentage, is defined below

$$\text{Ratio of BIT FR to Total Module FR} = 100 \cdot \frac{\sum_i \text{FRB}_i}{\sum_j \text{FRM}_j + \sum_i \text{FRB}_i} \quad (3.7)$$

where the variables are defined as above. The mean-time between failure (MTBF) is related to the failure rate by

$$\text{MTBF} = \frac{1}{\text{FR}} \quad (3.8)$$

3.4 Performance Criteria

This subsection is concerned primarily with understanding how "good" a job a particular BIT approach does. Clearly a sound quantitative measure is required if reasoned decisions are to be made. The nature of complex digital systems testing is such that something more than intuition must direct decision making. The focus of this section will be on BIT with a passive monitoring function only. However, many of the ideas and results can readily be applied to the evaluation of active BIT capable of

input control (test word insertion).

The two performance measures used in this report are percent of packages monitored and percent of gates monitored. These measures generally correspond to two other measures sometimes used, namely, percent of function tested and confidence level, respectively. The measures used in this report are intended to have names that suggest their definitions and to be defined more explicitly than previously used measures.

3.4.1 Percent of Packages Monitored

Fault monitor coverage can be related to the percent of the total module packages monitored. This performance measure is defined as

$$\begin{aligned} \text{Percent of} \\ \text{Packages Monitored} = & \frac{100 \cdot \text{Number of Packages Monitored(Including BIT)}}{\text{Total Number of Packages(Including BIT)}} \end{aligned} \quad (3.9)$$

A package is said to be monitored if faults within that package that cause an erroneous result cause the pass/fail output signal to indicate a fault. In practice this definition is relaxed somewhat to include all those packages in an array whose "data" outputs are fully monitored, but have a small number of "status" outputs which are not verified. (Example: a string of counters whose final ripple carry output is not checked.)

3.4.2 Percent of Gates Monitored

Because of the wide range of integrated circuit complexities within the TTL family used to implement the QED modules, a more realistic indication of the percentage of total possible faults that can be detected may be given by the measure defined below.

$$\text{Percent of Gates Monitored} = \frac{100 \cdot \sum_i G_i}{\sum_j G_j} \quad (3.10)$$

where G_i is the number of equivalent gates in the i -th package, and

$\sum_i$ is the sum over all of the monitored packages (including BIT). $\sum_j$ is the sum over all of the module packages (including BIT). A package is defined as monitored in the same manner described above. In determining the percent of gates monitored it is possible and even desirable to look at the individual gate level diagrams of the partially monitored packages, decide how many gates are indeed monitored with each BIT circuit and add these numbers to the numerator of the definition. This approach was not taken because the additional accuracy is not essential to the objectives of this study.

3.4.3 Other Possible Performance Measures

Since the monitoring circuit considered here is passive, it cannot initiate a test for the presence of a given fault. Therefore, it seems reasonable to only attempt to measure how effectively it will monitor and validate module results given the data inputs which occur. The name monitoring capability index (MCI) is chosen for the measure to be defined subsequently. It is important to understand that a high monitoring capability index and a valid pass/fail line indicate proper module operation only for that portion of the module that the input word set has exercised. Under these conditions the user can be confident that the output data is valid, but the user should not assume that the module is fault-free. The MCI indicates the potential to detect failures given the input conditions necessary to produce detectable errors.

The level of monitoring capability depends upon three things:

- (1) The percent of the function monitored (function coverage);
- (2) The percent of the module fault conditions which can be detected given proper inputs (fault coverage), for the monitored portion of the circuit, and
- (3) The percent of time that the results are monitored (cycle coverage).

Each of these three properties is now described in more detail.

3.4.3.1 Function Coverage

A package is said to be covered if all of its inputs or predecessors of its inputs are monitored and its outputs or successors of its outputs can be monitored and verified.

In order to develop some scale of monitored versus unmonitored behavior it is necessary to count packages, gates or another quantity. In fact, the number of faults which fall within the monitored area would be of highest interest. It is often difficult to provide this number. A reasonable compromise is to use failure rate as a scale of fault presence and to define the percent function monitored (PFM) as

$$\text{PFM} = \frac{\text{FR of covered packs} \cdot 100}{\text{FR of the Module}} \quad (3.11)$$

Both of the FR numbers include the BIT circuitry itself.

3.4.3.2 Fault Coverage

An example of complete fault coverage is given in Figure 3.1 and partial input coverage which may result in partial fault coverage is shown in Figure 3.2. The BIT of Figure 3.1 does all that the operational module does for every input combination and this will detect all faults assuming no failure of the BIT. The approach demonstrated in Figure 3.2 is motivated by its flexibility and reduced expense. However, only a limited number of input cases are checked by the monitor. Hence, presumably less than full fault detection results. A parity checker is an example of 100% input coverage but less than 100% fault coverage.

In the most detailed view of this parameter, the fault coverage is more dependent upon coverage of input values which check a large number of faults than inputs which do not check many faults. That is to say the index is a measure of the number of faults which can be monitored with respect to the total number of faults. A weaker measure, but one which is easier to obtain, is to count the number of input cases which can be monitored versus the total number of input cases. In general, the number of input patterns required for a complete test will be smaller than

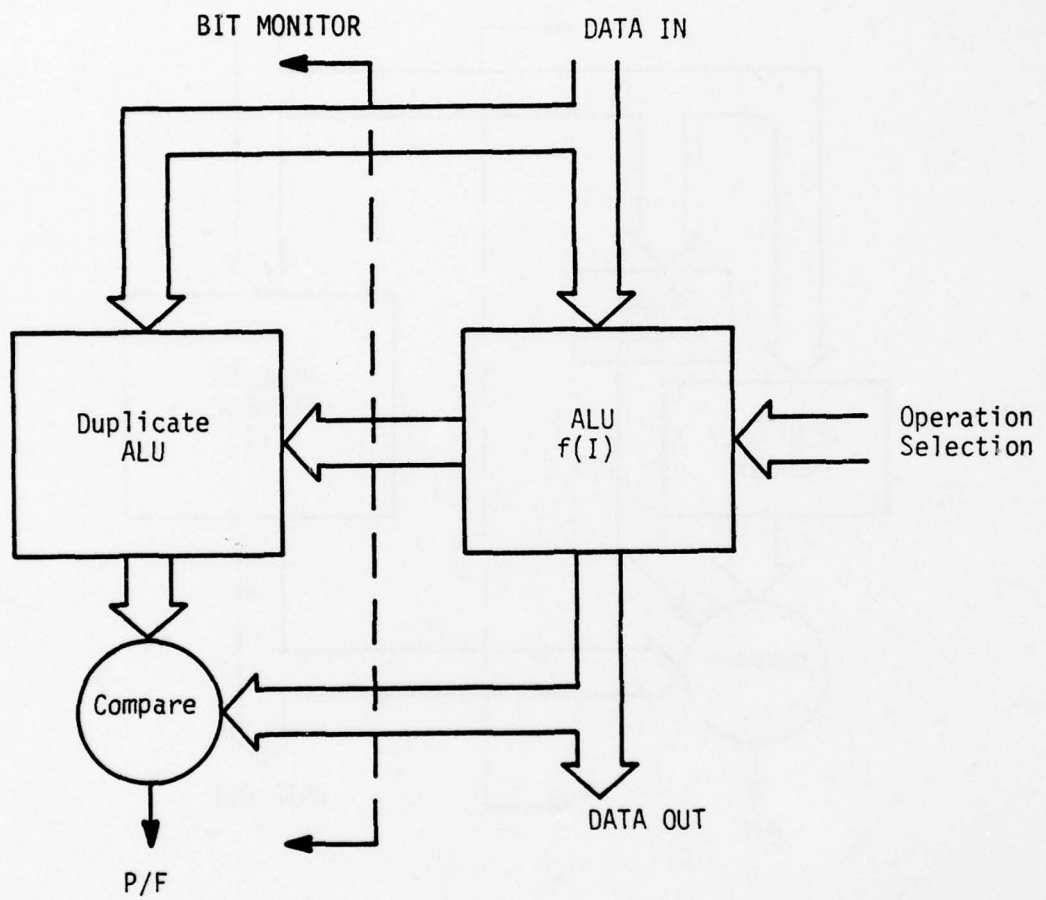


Figure 3.1. Monitor with 100% Input Coverage

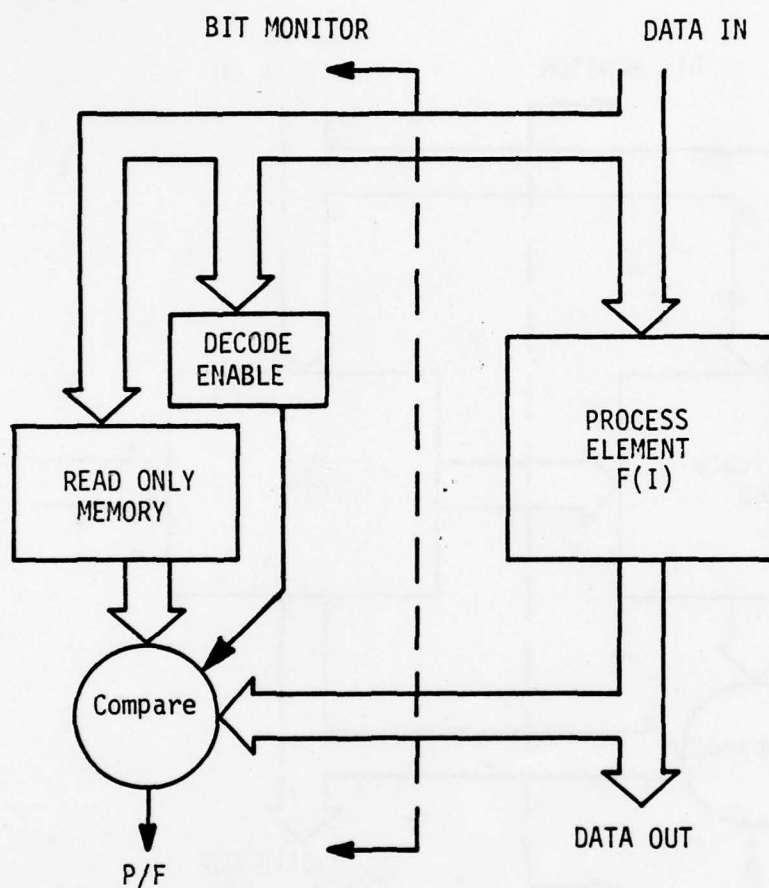


Figure 3.2. Monitor with Partial Input Coverage

the total number of input patterns but harder to determine. These two points of view are summarized by the following definitions:

Percent Input Coverage (PIC) =

$$\frac{100 \cdot \text{Number of Inputs, } I, \text{ for which } f(I) \text{ can be Verified}}{\text{Total Number of Inputs for which } f(I) \text{ is Defined}} \quad (3.12)$$

and

Percent Fault Coverage (PFC) =

$$\frac{100 \cdot \text{Number of Faults which can be Detected}}{\text{Total Number of Faults in } f(I)} \quad (3.13)$$

3.4.3.3 Cycle Coverage

For a module which monitors values continuously in time (as was the case in Figures 3.1 and 3.2) the present cycle coverage is 100%. When a sampling scheme is used (Figure 3.3), then there will be results which could be verified (with respect to fault coverage) but will not be because they are not sampled.

The primary motivation in considering a sampling scheme is that by reducing the time demand on the monitor, it is possible to consider using a programmable device. Programmable devices are typically slower but more flexible and thus applicable to a wider range of monitoring tasks.

If the monitoring device has a sampling rate of S samples per second and the device has an operating rate of C cycles per second, then the percent cycle coverage is given by

$$\text{Percent Cycle Coverage (PCC)} = \frac{100 \cdot S}{C} \quad (3.14)$$

When the device being monitored is asynchronous, then the cycles per second measured may be taken as the maximum number of cycles possible for

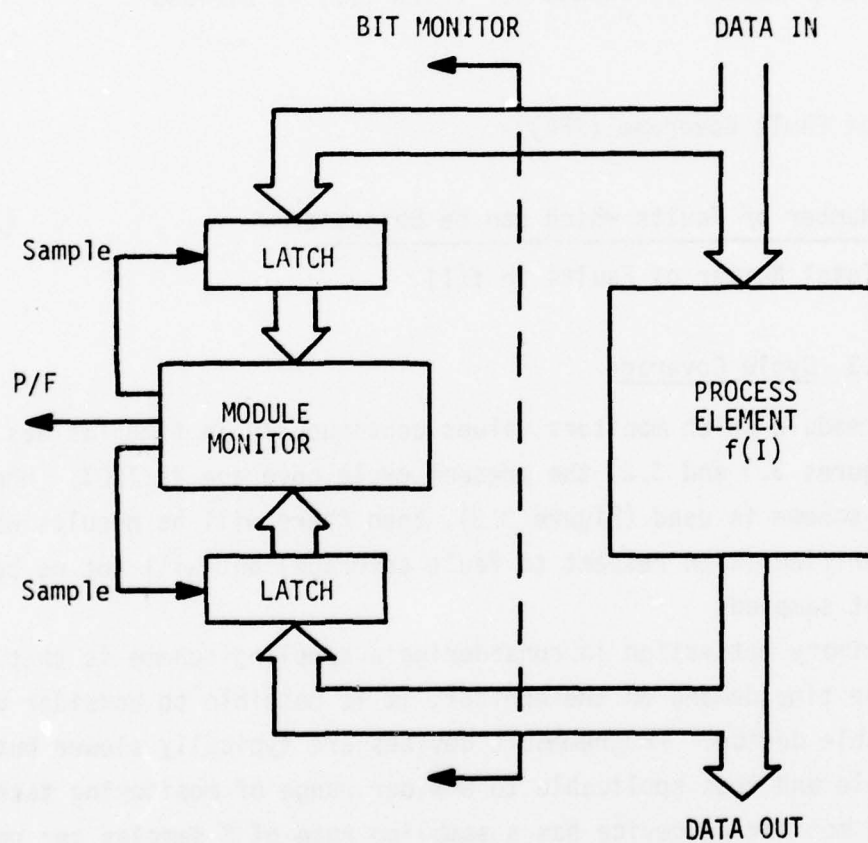


Figure 3.3. BIT Monitor with Partial Time Coverage

the device propagation delay.

3.4.3.4 Composite Measure

At this point it would be useful to attempt to define the overall idea of module monitoring capability as a composite of PFM, PIC, and PCC. For example one might consider a weighted sum of these components such that,

$$MCI = \alpha \text{ PFM} + \beta \text{ PIC} + \gamma \text{ PCC.} \quad (3.15)$$

The problem then becomes one of choosing appropriate values for the coefficients α , β and γ . In such a formulation the weights are directly determined by the relative importance of the percent of the function monitored, the percent input coverage and the percentage of the total number of cycles covered. Additional work needs to be done to both validate this basic equation and to determine α , β and γ or their equivalents.

As an example of an application of a composite measure, consider the following values:

$$\begin{aligned} \text{PFM} &= 84\%, \\ \text{PIC} &= 100\%, \\ \text{PCC} &= 100\%. \end{aligned}$$

One might consider that the composite monitoring capability provided is given by an index of 0.95 (ie., $\alpha = \beta = \gamma = 1/3$). The confidence level, however, is no better than the total fault coverage of the data supplied to the module by the system.

For the second example (Figure 3.3) the entire module is monitored resulting in a PFM of 100%. The other measures are taken as hypothetical. Sampling hardware can, in general, readily provide 100% input coverage and must produce less than 100% cycle coverage. An example of cycle coverage can be given by typifying the sampling device as a micro-processor. Estimates of performance are given in Table 3.1.

Estimate of the Number of CPU Cycles to run an ALU emulation	=	600
Cycle time (Z-80 example)	=	120ns
Total Emulation time	=	72 μ s
S = Sample rate	=	14 $\cdot$ 10 ³ /s
Worst case (fastest) delay of ALU	=	110ns
C = Worst case (most) cycles/s	=	9 $\cdot$ 10 ⁶ c/s
Cycle Coverage = 100 $\cdot$ S/C	=	0.15%

Table 3.1 Example Cycle Coverage

For the ALU then

MCI: PFC = 100%
 PIC = 100%
 PCC = 0.15%

Intuitively, the cycle coverage seems less significant since the same amount of testing may be obtained by allowing more elapsed time. As long as the elapsed time does not become significant with respect to system time constants, the cycle coverage reduction may not practically affect the usefulness of the BIT.

Since considerations such as this are so dependent on system factors, it may be found that the composite measure can best be defined in a system dependent way.

In conclusion if meaningful statements can be made about the data seen by a module in a certain period of time, then something can be said about the confidence level. For example, if a module with BIT which has PFM = 100%, PIC = 100%, PCC = 100%, and for which input data constituting a complete test set has been processed (within certain time limits), then the pass/fail indication may be interpreted with 100% confidence. Remove any of the above qualifiers and it is not clear what can be said about confidence level. Confidence level is generally a measure of active, not passive behavior. Therefore, if a BIT circuit has an active mode in which it is possible to insert test words, only then would it be possible to define a meaningful confidence level.

4.0 APPROACHES TO BIT FOR QED MODULES

The preceding sections have presented an overview of possible approaches for built-in-test of functional digital logic modules. Various general approaches to module level fault detection have been suggested including replication and a standard BIT sampling technique as well as several general purpose BIT circuits.

This section considers specific approaches to built-in-test. In particular, the members of the QED module family have been grouped according to the logical functions which they perform and recommended test approaches proposed.

The resulting functional module classifications for the QED family are:

PROCESS CLASS

- Arithmetic Logic Unit
- Parallel Multiplier
- Index Counter
- Microprocessors

MEMORY CLASS

- Random Access Memory, TTL
- Read-Only Memory, TTL
- Dual FIFO Memory
- Random Access Memory, MOS
- Read-Only Memory, MOS

INTERFACE CLASS

- Asynchronous Serial Interface
- Dual Parallel 8-bit Interface
- Dual 8-bit Switch
- NTDS to TTL Buffer
- TTL to NTDS Buffer

CONTROL CLASS

- Programmable Timing Generator
- Priority Encoder

4.1 Built-In-Test for Memory Class Modules

There are five QED modules in the memory class. These modules will be examined in three groups according to function, as listed below:

Group I
Random Access Memory (RAM) TTL
Random Access Memory MOS

Group II
Read-Only - Memory (ROM) TTL
Read-Only - Memory MOS

Group III
First-In-First-Out Memory (FIFO)

The modules within each group provide the identical function to the user and therefore will be treated as one type of module. Any BIT technique that is beneficial to a TTL type memory will be equally beneficial for MOS memory monitoring.

The memory class modules are characterized by their ability to store data. This data can be program object code, numerical data, character representations, etc. The data storage format is in eight-bit bytes for all memory modules.

The BIT approach recommended in this section for the memory modules includes a parity generator/checker as a standard approach to monitoring module interface circuitry and interconnections. This standard BIT circuit is described in Section 4.1.5. Additional BIT techniques which provide monitoring of data within the module are examined in the following subsections.

4.1.1 Random Access Memories

There are two basically different approaches to checking memory modules that have contrasting effects on the system designer. One approach is off-line testing, which is when the module is tested, it is put into a test mode so that normal processing cannot be done. The other approach is on-line. That is the module monitoring is performed on real data in real time with no resulting limitation on system throughput

or overall speed. Of the various methods of on-line testing, replication and coding are the most practical approaches for RAM testing.

4.1.1.1 Duplication

Replication of the memory circuits is a straightforward technique for providing error detection. In the write mode each data word is stored in two separate memory circuits and when the data is read out of the memory module, the two are compared. Thus any differences indicate a fault has occurred. This method of fault checking detects all errors in the memory including multiple bit faults and addressing errors. A block diagram of this expensive approach is shown in Figure 4.1.

4.1.1.2 Parity

Coding techniques offer a significant reduction in hardware over duplication at the expense of a decrease in fault detection capability. Single bit per word parity, the simplest coding technique, is also the most common memory error detection technique. To implement parity an additional bit is added to each data word and the value of this bit is determined so as to make the sum of the number of one bits in the word an odd number for odd parity, an even number for even parity. When the data is read out, the parity is again generated for the data bits and compared to the stored parity bit. If these two parity bits are not the same, a fault is indicated. This coding detects all errors in an odd number of bits which include all single bit errors. A block diagram of a RAM module using parity is shown in Figure 4.2.

The QED memory modules are constructed using RAM circuits such that each memory integrated circuit contributes only one bit to the data word. It is therefore possible to detect both addressing errors and memory cell failures as long as there are an odd number of total errors in the data word. Single bit parity cannot detect faults in an even number of bits. It is possible to detect a greater number of multiple errors if a greater number of check bits are used. Unfortunately, these alternatives do not increase the fault detection capability as fast as hardware

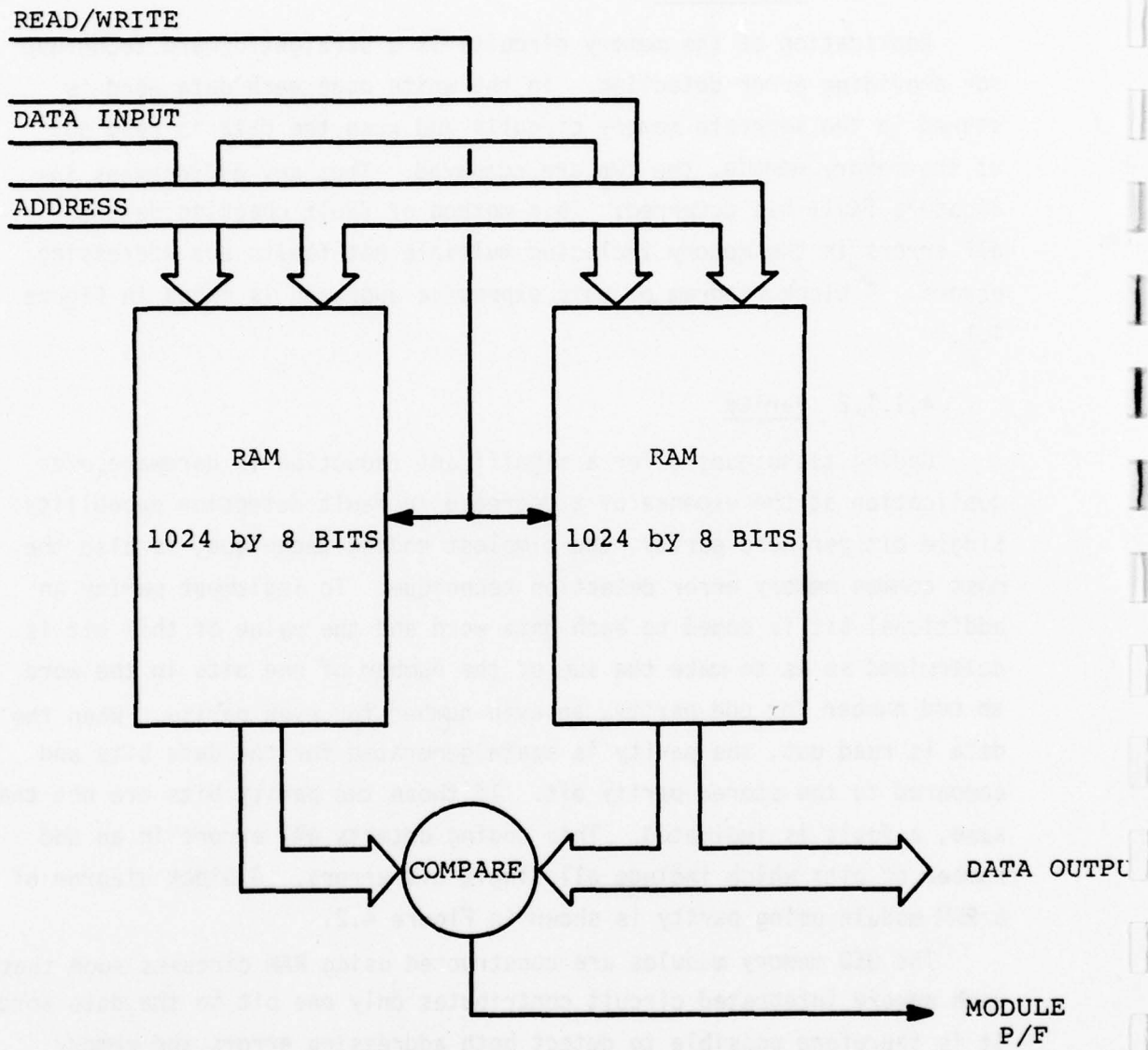


Figure 4.1 RAM BIT by Duplication

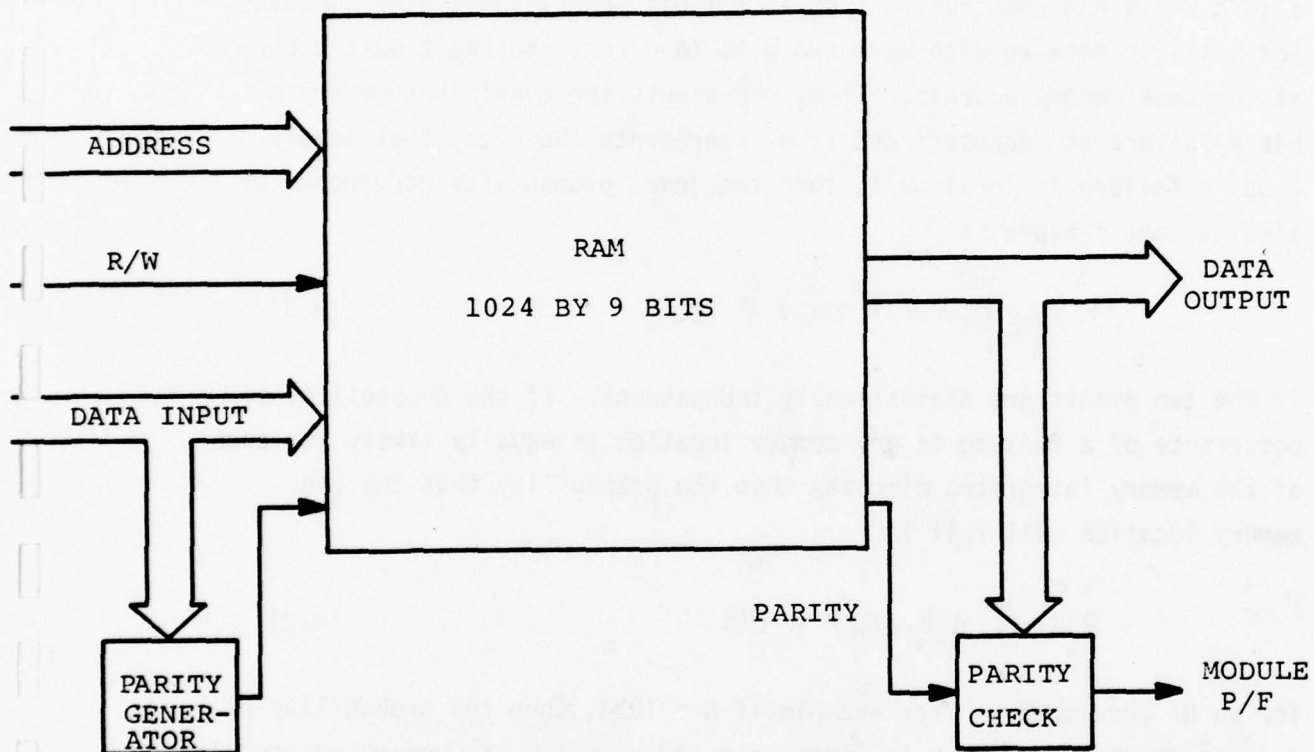


Figure 4.2 RAM BIT by Parity

costs and will not be pursued at this time.

The question must be asked as to the likelihood of multiple bit failures in the memory modules currently being considered. The following provides some insight into the likelihood of multiple fault occurrence.

To compute the probability of occurrence of an even number of failures, consider the case of two simultaneous memory chip outages in a 1K X 9-bit RAM constructed from 1K X 1-bit memory integrated circuits. For a single data word to have two bits in error, the fault must occur at the same memory address. If m_i represents the event that memory m has a failure at address i and if n_i represents the event that memory n has a failure in location i , then the joint probability occurrence of simultaneous failure is

$$P(m_i | n_i) = P(m_i) P(n_i) \quad (4.1)$$

if the two events are statistically independent. If the probability of occurrence of a failure in any memory location is equally likely for each of two memory integrated circuits then the probability that the i th memory location will fail is

$$P(m_i) = P(n_i) = 1/N, \quad (4.2)$$

for an N -word memory. For example if $N = 1024$, then the probability of two simultaneous failures in memories m and n in the i th memory address is

$$P(m_i | n_i) = \left(\frac{1}{1024}\right) \cdot \left(\frac{1}{1024}\right) \approx 1 \times 10^{-6} \quad (4.3)$$

Since double bit errors can occur in any pair of memories, the combination of nine things taken two at a time, which is equal to 36, must be multiplied with the result from equation (3.4) to obtain the probability of a double bit error, P_α . Therefore,

$$P_\alpha = 36 \cdot \left(\frac{1}{1024}\right) \cdot \left(\frac{1}{1024}\right) = 3.43 \times 10^{-5} \quad (4.4)$$

which means that of the possible errors in the memory locations only about three in 10,000 will be double bit errors.

One can readily see that the occurrence of more than two simultaneous, even number of failures is much less likely under the same assumptions. Thus multiple, even number of bit failures are not of major concern in parity checking approaches for properly organized memories.

4.1.1.3 Single BIT Error Correction

It is possible to correct errors in memory systems with the use of multiple check bits. For single bit error correction the number of check bits required may be determined by the inequality:

$$2^k \geq m + k + 1 \quad (4.5)$$

where m = number of data bits, and k = number of Hamming parity bits or check bits [6]. Solving this equation where $m = 8$, shows that a minimum of 4 check bits are needed to correct single bit errors in 8 bit data words. While it may appear that 4 bits to check 8 bits is a considerable amount of overhead, error correction can improve the MTBF. This is something that simple error detection can never do. Error correction works by generating and storing k (Equation 4.5) parity bits from specific subsets of bits from the data word. When the data is read out, the parity generation process is repeated on the data bits. If these two sets of bits are different, an error has occurred. The bit pattern made by the exclusive OR-ing of the two sets of parity bits can be decoded to indicate in which bit position the error has occurred. This bit is complemented which corrects the error. Error correction as described cannot correct multiple bit errors and cannot detect most multiple bit errors. A block diagram illustrating this approach for a QED RAM module is shown in Figure 4.3.

4.1.1.4 Off-Line Testing

To provide built-in-test in an off-line mode, it is necessary to provide a test pattern generator on the module. The most common type of bit pattern used to check a RAM is a "checker board" (alternating ones

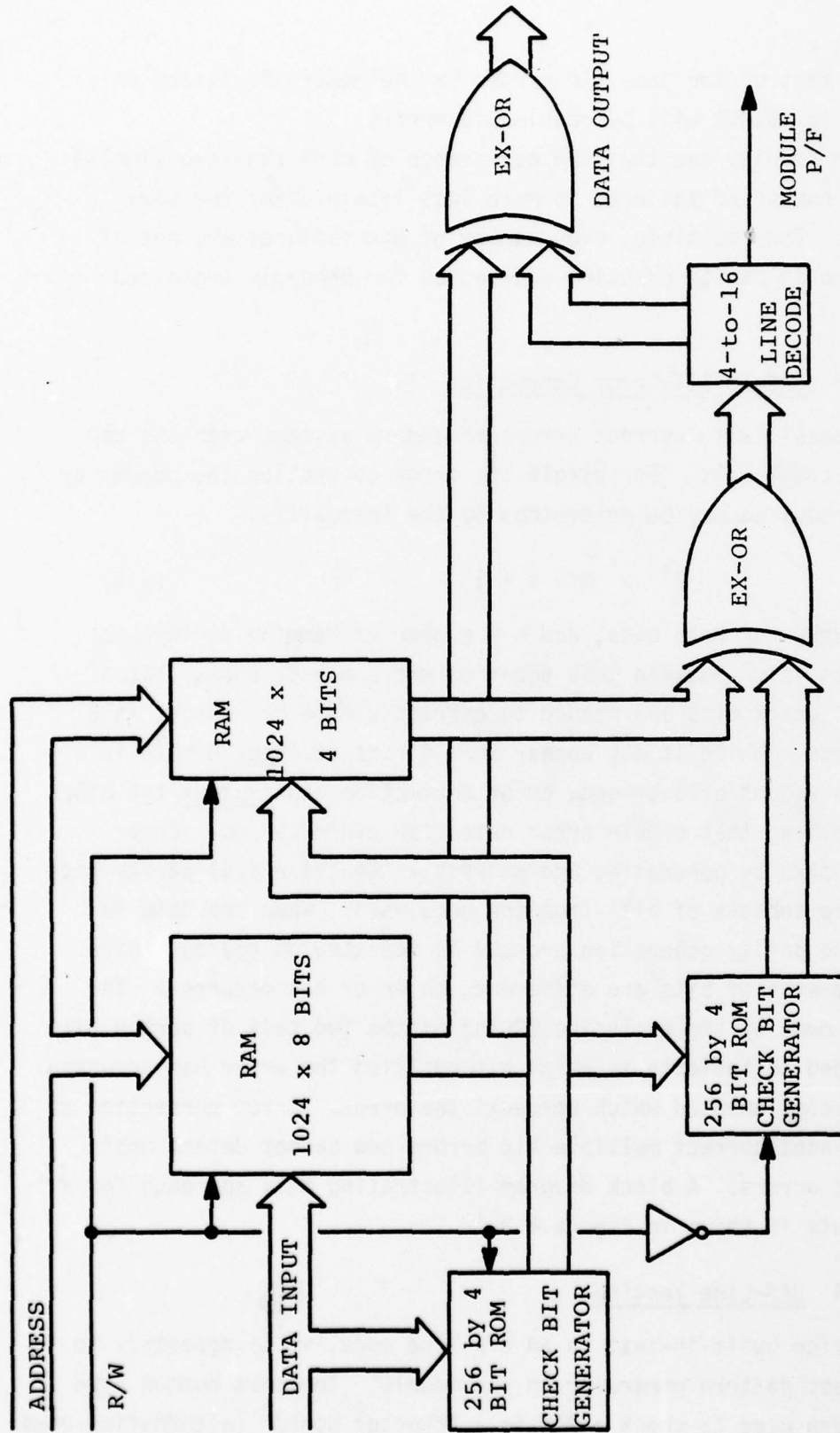


Figure 4.3 RAM BIT with Error Correction.

and zeroes) pattern which is easily generated. While a cyclic pseudo-random generator can also be utilized as an input sequence, a simple shift register can be used to generate an alternating one and zeroes test pattern with less hardware. To check for all stuck-at-faults, it is necessary to write and read back a one and a zero in each bit position so actually two tests patterns (one the inverse of the other) are needed. After the test pattern has been written into memory, the pattern generator is restarted and the bit pattern read from the memory is compared to the regenerated input bit pattern. Any pair of bits that are different indicates a fault. A block diagram of a RAM module with off-line BIT is shown in Figure 4.4.

In using test pattern techniques, it is necessary to provide a method of bus isolation within the module so that the test patterns are not propagated throughout the whole system. Other necessary functions are a clock, an address counter, and some basic control for the test mode operation. In all off-line testing that is performed while the system is operating, the system designer must make provisions for restoring the data in the tested modules so it will not be lost because of the test.

4.1.1.5 Recommendations

The recommended approach for the built-in-test of the RAM modules is word parity. A block diagram of a module with word parity is shown in Figure 4.5. This method is recommended because it uses the least additional hardware while providing a high error detection capability. It is clear that the parity technique uses the least number of additional memory circuits, since the parity approach uses only one additional memory circuit, the error correction approach uses four, and the duplication approach uses eight. The off-line approach requires no additional memory circuits, but it does use a substantial number of MSI and SSI packages. Because of the strong correlation between the number of packages (especially complex memory circuits) and costs (board space, power and failure rate) it is of fundamental concern to minimize this parameter. The parity approach certainly satisfies this goal. The

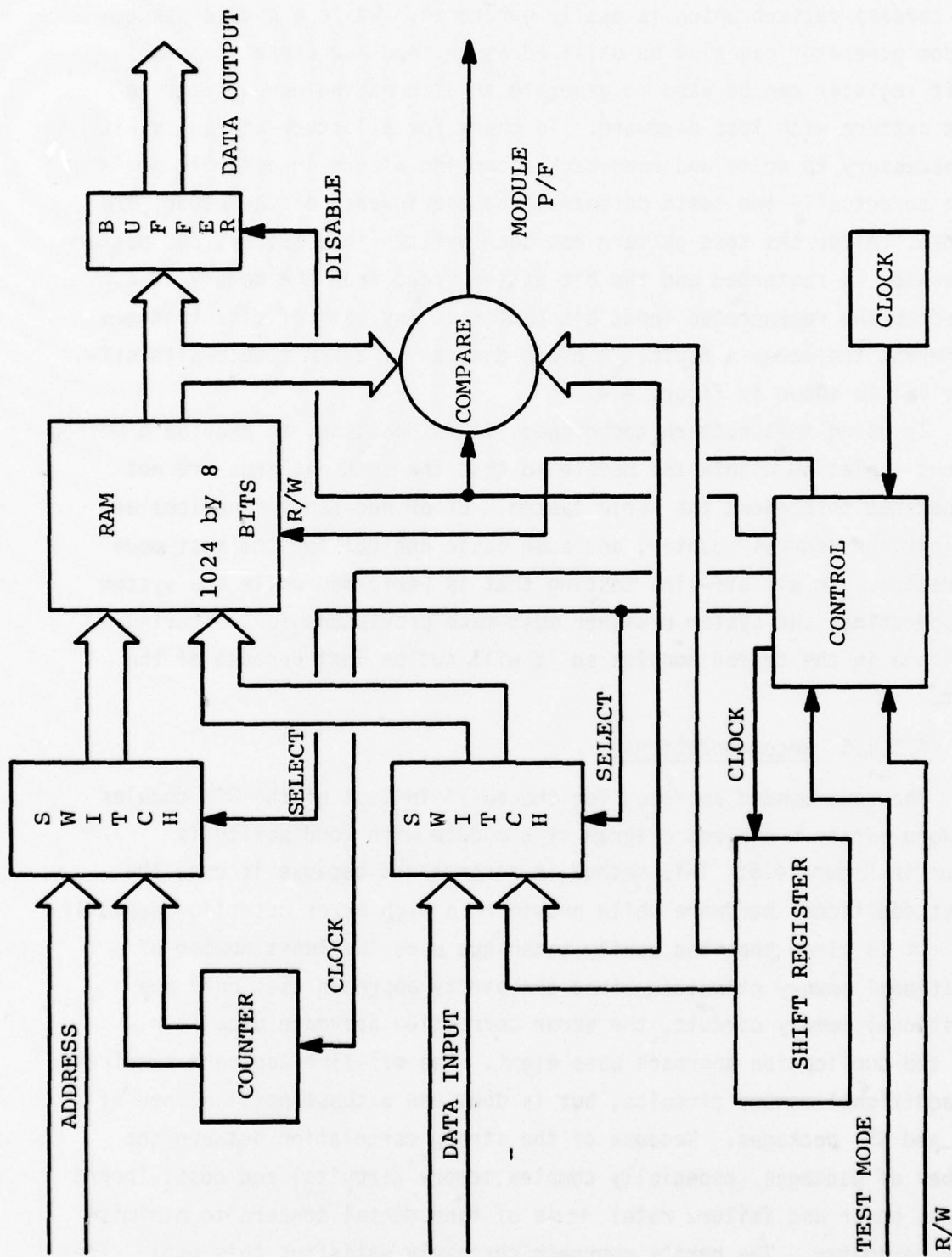


Figure 4.4 RAM Off-line BIT.

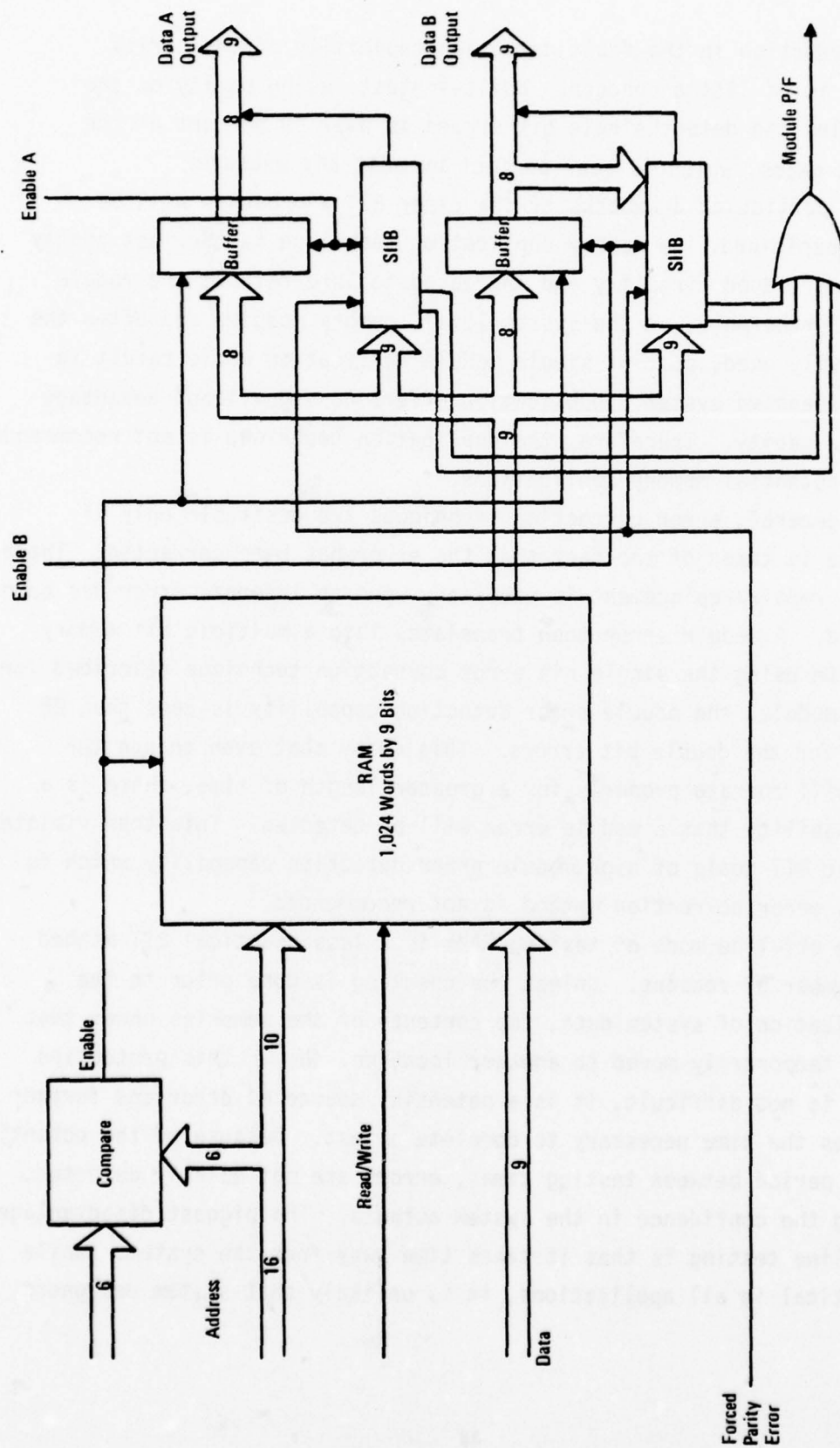


Figure 4.5 RAM Module With Recommended BIT

slight reduction in the fault detection capability of the parity approach is of little concern. Built-in-test using parity on the RAM modules can detect single bit errors in over 99 percent of the module's gates, which is near perfect by most any measure.

The particular drawbacks of the other BIT approaches will be further explained. The memory duplication technique is the most costly in terms of added circuitry and increased failure rate at the module level. Furthermore, at the system level, memory modules are often the most heavily used, so that simple module duplication would result in a very expensive system. Duplication offers no significant advantage over word parity. Therefore, the duplication technique is not recommended for any potential memory applications.

In general, error correction techniques are desirable only if advantage is taken of the fact that the error has been corrected. Therefore, no repair/replacement is necessary when an internal error has been corrected. A module error then translates into a multiple bit memory error. In using the single bit error correction technique described for the RAM module, the module error detection capability is less than 25 percent for the double bit errors. This means that even though the module will operate properly for a greater length of time, there is a low probability that a module error will be detected. This then violates the basic BIT goals of high module error detection capability which is why this error correction method is not recommended.

The off-line mode of testing RAMs is a less practical BIT method for a number of reasons. Unless the checking is done prior to the initialization of system data, the contents of the memories under test must be temporarily moved to another location. While this protection of data is not difficult, it is a potential source of error and further increases the time necessary to complete a test. Because of the potentially long period between testing times, errors are not quickly detected, reducing the confidence in the system outputs. The biggest disadvantage of off-line testing is that it takes time away from the system. While not critical in all applications, it is unlikely that system designers

would welcome these restrictions and the added design effort needed to control the timing of the tests.

4.1.2 Read Only Memories

Read-only memories (ROMs) function in the same manner as RAMs during read cycles and can be tested using similar techniques. The on-line modes of providing BIT to RAMs are theoretically directly applicable to testing ROMs. A full description will not be repeated but there are a few modifications that must be noted. With respect to duplication of memory circuits, the only functional change from the RAM diagram is duplication of the package enable decoder so that each decoder can drive a separate ROM array. The coding and storage of the check bits for error correction is the same as the RAM except that the code is programmed in other ROMs and there is no code generation done on the module (only code checking).

A problem of geometry arises when implementing parity on a ROM memory in a method similar to a RAM memory. A RAM circuit may only be one bit wide; that is to say for each address only one bit may be read or written. To provide a memory with multiple bit words, a number of RAM circuits are used with their address lines bused together so that each RAM integrated circuit supplies one bit of the data word. To add a parity bit the designer simply adds one more RAM circuit to the address bus to store the parity bit for each data word. This point is illustrated in Figure 4.6.

In contrast to the RAM case, typical commercially available ROM circuits are either four or eight bits wide, so that for each address four or eight bits are output at one time. Since ROMs are not commercially available in one bit wide (or nine bit wide) configurations, the addition of a ROM in a manner like the RAM is impossible. Parity can still be added in an efficient manner as shown in Figure 4.7. The specific arrangement of the ROM circuits on the QED module is four circuits, each having 512 eight-bit wide words. Four enable lines are decoded from the address so that only one of the four circuits provides the output of the module. Parity can be added to the module with the addition of a 512 by 4 bit ROM. The nine address lines that are connected to the four data ROMs

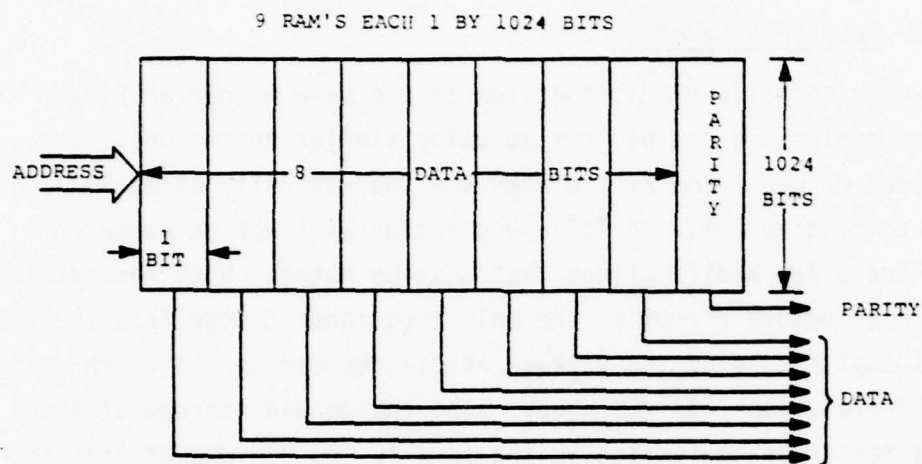


Figure 4.6 Parity Implementation in RAM

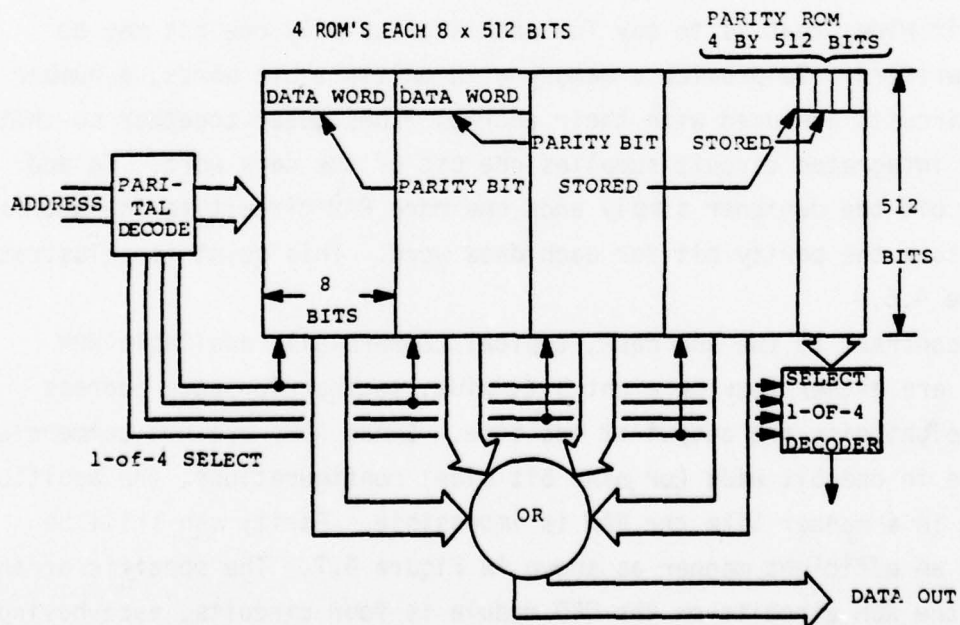


Figure 4.7 Parity Implementation in ROM

will also be connected to the parity ROM. The single parity bit for each data word can easily be decoded from the 4 available from the parity ROM by a select 1 of 4 decoder connected to the circuit enable lines. Using this technique all ROM storage is utilized and the additional BIT circuitry is not excessive.

The off-line technique described for use in the RAM module is not suitable for use with the ROM module because the testing circuitry cannot write data into the ROM. However, another off-line approach, longitudinal parity, can be used to provide BIT for the ROM modules. This is achieved by stepping through each address of the ROM and calculating the parity on each bit position. Using eight modulo-2 counters, a longitudinal parity word is generated which can be compared with a fixed constant to verify the correctness of the data in the ROM. The fixed constant would most economically be stored in one of the ROM locations, so that the 2048 word ROM module would become a ROM module with 2047 useable locations. A diagram of this approach is shown in Figure 4.8.

The recommended BIT approach for the ROM module is word parity and a block diagram of such is shown in Figure 4.9. The methodology for selecting the best BIT method is identical to the RAM module. The word parity implementation provides a check on almost the entire memory module with a relatively small increase in hardware. This BIT technique can detect all single bit stuck-at-faults within the ROM integrated circuits and some internal addressing decoder errors. While this approach cannot detect all addressing errors, it will generally detect half of them. This reduction in error detection capability affects the overall fault detection capability only in an inappreciable way.

The major drawbacks of each of the other BIT methods will now be considered. The major disadvantage of the off-line approach is the large number of additional packages necessary for BIT. This does not affect the failure rate as much as may be expected because the added packages are relatively simple and therefore have low failure rates. (However, the parity approach has a lower failure rate than the off-line approach.) Another drawback to this BIT approach is that the testing function and system use cannot occur at the same time. However, it is possible to

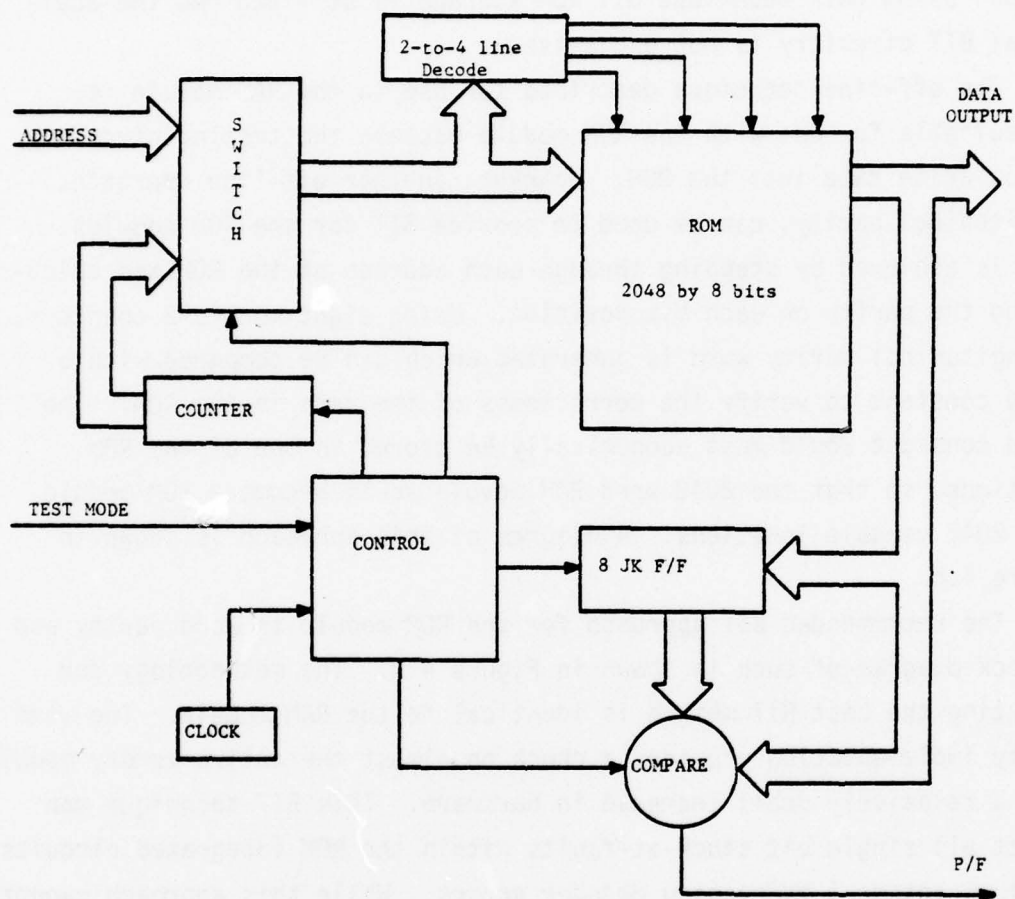


Figure 4.8 ROM BIT by Longitudinal Parity.

perform the testing, a part at a time, during the period that the system is not using the module (i.e., cycle stealing). With proper system timing, in many applications it is possible to check the ROM module and not slow down system throughput. However, this control of timing would make the system more complicated. The error correction technique is not desirable because of the low module error detection capability as explained for the RAM module. Duplication offers no significant benefits to BIT; it only increases cost.

4.1.3 First-In-First-Out Memory

The FIFO memory is similar to the other memory modules when considering built-in-test. The RAM on-line error detecting techniques have an analogous method in the checking of the FIFO. Memory duplication with a comparator checking the outputs can be applied to the FIFO just like the RAM. The coding techniques also can be applied to the FIFO. In fact, provisions for parity are made by the manufacturers of the FIFO in that the circuits are designed to store nine-bit data words. This fact makes parity checking the most economical of all BIT approaches because a very small amount of additional hardware is needed. A block diagram of the FIFO module with parity is shown in Figure 4.10.

An off-line approach that functions much like the RAM BIT is applicable to the FIFO. Using this approach, a test control signal enables a test pattern generator which writes into the memory until it is full. The memory then outputs the information which is compared to the output of the test pattern generator. Using this scheme, the system designer must use caution in the timing of the test mode signal because in order not to lose any data, the test mode signal should be given only when the FIFO is empty.

The FIFO module is best suited to use parity as a built-in-test method because the integrated circuits are made to handle nine bit words. This fact makes other built-in-test methods much more costly (with only slight gain in error detection capability) because of the higher number of additional circuits necessary to implement them. Additional circuits also increase the failure rate and power consumption of the module.

4.1.4 Further Discussion of Built-In Error Correction Techniques for Memory Class Module

It is envisioned that functional digital modules like the QED family will be used in applications where manual intervention to effect system repair may not be possible. For example, tactical fighter aircraft and helicopters require highly reliable electronic systems for relatively short duration missions. In these applications it is of little consequence to be able to note that an electronic module has failed, if action to repair that fault cannot be taken. Fortunately, in the case of digital memory functions, relatively simple codes exist which can be used to automatically correct errors in a straightforward way without operation intervention. This discussion considers a particular error correction code and its application to memory class functions.

To take full advantage of error-correction codes and achieve minimum maintenance and built-in-test, it is necessary to provide detection capability for double bit errors in addition to the correction of single bit per word errors. Since the single bit errors are only errors within the modules, the operation of the module as far as the system is concerned, is faultless. Therefore the module need not be replaced when a single error has occurred. It must be replaced only when a double error has occurred. In this way the MTBF of the module can be increased, but only if double-bit errors can be detected. In general it is possible to convert a single-bit error correcting code into a single-bit error correcting with double bit error detecting code by adding one additional bit which is a parity bit over all of the data and code bits. Using the Hamming single-error correcting code described previously, five check bits would be needed to provide single error correction with double error detection on an eight bit data word. The coding of the check bits would be identical to the single error correction case (with the addition of the overall parity bit) as would be deciding and error correction. Double errors are indicated by the overall parity of the word being correct, but the error correcting check bits indicating an error has occurred. Single (correctable) errors are indicated by an error in both the overall parity and the error correcting codes indicating an error. No error, of course, is indicated by no errors in any of the parity checks.

To more easily quantify the increase in MBTF in order to justify the additional cost in space, power, and dollars needed to implement an error correcting code, it is necessary to make several assumptions. The single error correcting, double error detecting code will correct all single bit per word errors and detect all double bit per word errors in the storage elements. The following analysis will assume all errors are independent. A single bit per word parity scheme as previously described will be used as a basis for comparison. The parity approach will detect all single bit word errors and will assume that these errors are independent. The differences in detection/correction of three or more errors will be ignored because of the small probability of their simultaneous occurrence.

The following data for each type is based on a module storing 1024 eight bit words using 1K by 1 bit RAMs for storage and MSI/SSI level support circuitry. In the computation of the reliability of the memory using error correcting codes, the memory module is divided into two parts. One part contains the memory chips themselves and their reliability is computed based on the fact that only 12 of the 13 memory chips need to be working in order for the whole module to be working. For this portion of the module the reliability does not follow the exponential model (failure rate independent of time) that most electronics do, but rather is described by the equation

$$R_C = 13e^{-12\lambda t} - 12e^{-13\lambda t} \quad (4.6)$$

where R_C is the reliability of the array of memory chips, λ is the failure rate of one memory chip and t is time.

The other portion of the memory module is made up of all the other circuitry in the module. This portion of the module must be completely operational for the memory module to function properly. The reliability model for this portion of the module is the common exponential decay based on a uniform failure rate, (the failure rate is independent of time) and can be expressed as

$$R_S = e^{-\lambda_s t} \quad (4.7)$$

where R_s is the reliability of the support circuitry, λ_s is the composite failure rate of the circuitry (i.e., the sum of the failure rates of all the individual support chips) and t is time.

To compute the reliability of the entire memory module, the reliability of the memory chips is multiplied by the reliability of the support circuitry. This gives a reliability equation of the form

$$R_m = R_c R_s = 13e^{-(12\lambda_c + \lambda_s)t} = 12e^{-(13\lambda_c + \lambda_s)t} \quad (4.8)$$

where R_m is the reliability of the entire memory module, λ_c is the failure rate of one memory chip, λ_s is the failure rate of all of the support circuitry and t is time. The reliability of the memory modules with parity is determined in the conventional manner and can be expressed as

$$R_p = e^{-\lambda_p t} \quad (4.9)$$

where R_p is the reliability of the module with parity, λ_p is the composite failure rate of the module (the sum of the failure rates of the individual ICs) and t is time.

Figure 4.11 shows the reliability as a function of the time for the QED RAM module for the two built-in-test approaches just described (parity and single error correction with double error detection). The important question in evaluating the error correcting memory module is to quantify the gain in failure rate. With this information, it is possible to make a well founded decision on whether the additional costs of error correcting are worth the increased reliability and reduced maintenance.

Because the reliability curve of the error correcting memory module is not of form $R = e^{-\lambda t}$, it is impossible to assign a single number to the failure rate. However, it is possible to define a factor, F_R (that is a function of time), that realistically indicates the improved reliability achieved using the error correcting technique. This factor, F_R , can be defined as:

$$F_R(t) = \frac{\ln(R_p)}{\ln(R_{ec})} \quad (4.10)$$

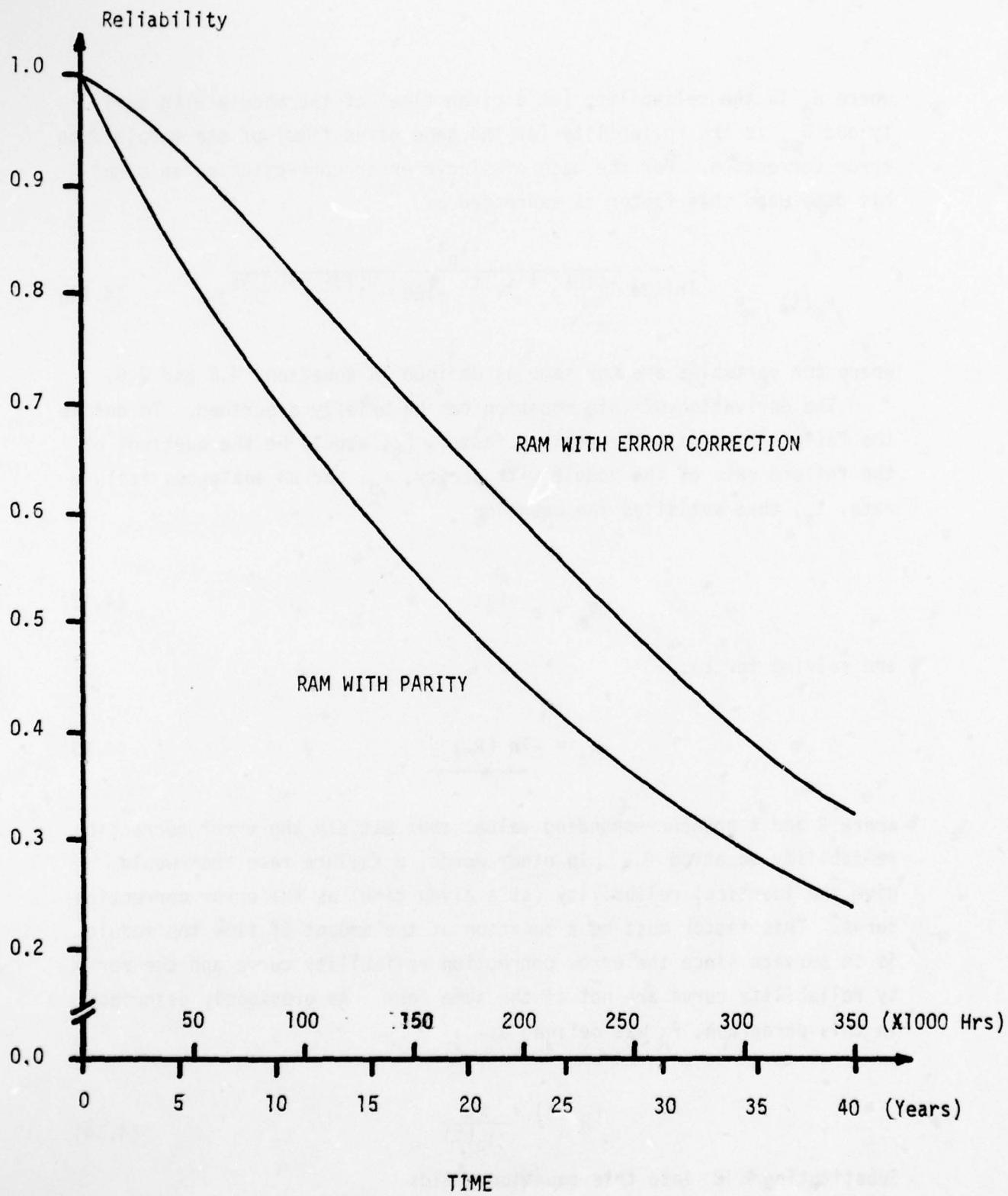


Figure 4.11 Reliability of RAM Modules

where R_p is the reliability (at a given time) of the module with parity and R_{ec} is the reliability (at the same given time) of the module with error correction. For the case of single error correction on an eight-bit data word this factor is expressed as

$$F_R(t) = \frac{e^{-\lambda_p t}}{\ln[13e^{-(12\lambda_c + \lambda_s)t} - 12e^{-(13\lambda_c + \lambda_s)t}]} \quad (4.11)$$

where the variables are the same as defined in equations 4.8 and 4.9.

The derivation of this equation can be briefly described. To define the failure rate gain the desired factor, F_R , should be the quotient of the failure rate of the module with parity, λ_p , and an analogous failure rate, λ_a , that satisfies the equation

$$R_m = e^{-\lambda_a t} \quad (4.12)$$

and solving for λ_a ,

$$\lambda_a = \frac{-\ln(R_m)}{t} \quad (4.13)$$

where R and t are corresponding values that satisfy the error correction reliability equation 4.8. In other words, a failure rate that would give the identical reliability (at a given time) as the error correction curve. This factor must be a function of the amount of time the module is in service since the error correction reliability curve and the parity reliability curve are not of the same form. As previously described in this paragraph, F_R was defined as

$$F_R(t) = \frac{\lambda_p}{\lambda_a(t)} \quad (4.14)$$

Substituting 4.12 into this equation yields

$$F_R(t) = \frac{-\lambda_p t}{\ln(R_m)} \quad (4.15)$$

which is identical to equation 4.11 once the equation (4.8), for R_m , the reliability of module with error correction, is substituted into it.

Figure 4.12 shows this increased reliability factor F_R as a function of time for the QED RAM modules as previously described. One can see that from an initial failure rate gain of a little over three, the reliability increase falls to a gain of about two after ten years. This shows the substantial reliability increase that can be achieved for quite a number of years. In addition, through preventative maintenance, it is possible to keep this failure rate gain over three. This achievement is made simply by an annual replacement of the modules that have internal failures that are being corrected. This replacement reduces the probability of a double bit error (and, therefore, increases the reliability of the module) because the modules that have already suffered single bit error are removed.

4.1.5 Standard Interconnection and Interface BIT

A special circuit has been designed to provide the parity generation and checking necessary to implement the recommended BIT technique. In addition, it was designed to provide parity generation and checking for the interconnecting data buses of all the QED modules. This capability provides a BIT technique to check the module input and output circuits as well as to detect wiring and connector faults. This special circuit is called the Standard Interface and Interconnection BIT (SIIB). The SIIB provides a module level, on-line (concurrent) fault monitoring capability which can supply module pass/fail information to a system fault monitor.

The SIIB, as a standard BIT circuit, was designed to be used on a large number of different modules. Because of its multi-function design, it can be used in alternate ways to check different circuits, thus reducing the number of necessary standard BIT circuits. In particular, the SIIB is designed to provide an error detection capability for the input latches and output buffers of the QED modules. In addition, the SIIB provides an error detection and isolation capability for interconnecting circuitry including logic card connectors and backplane wiring.

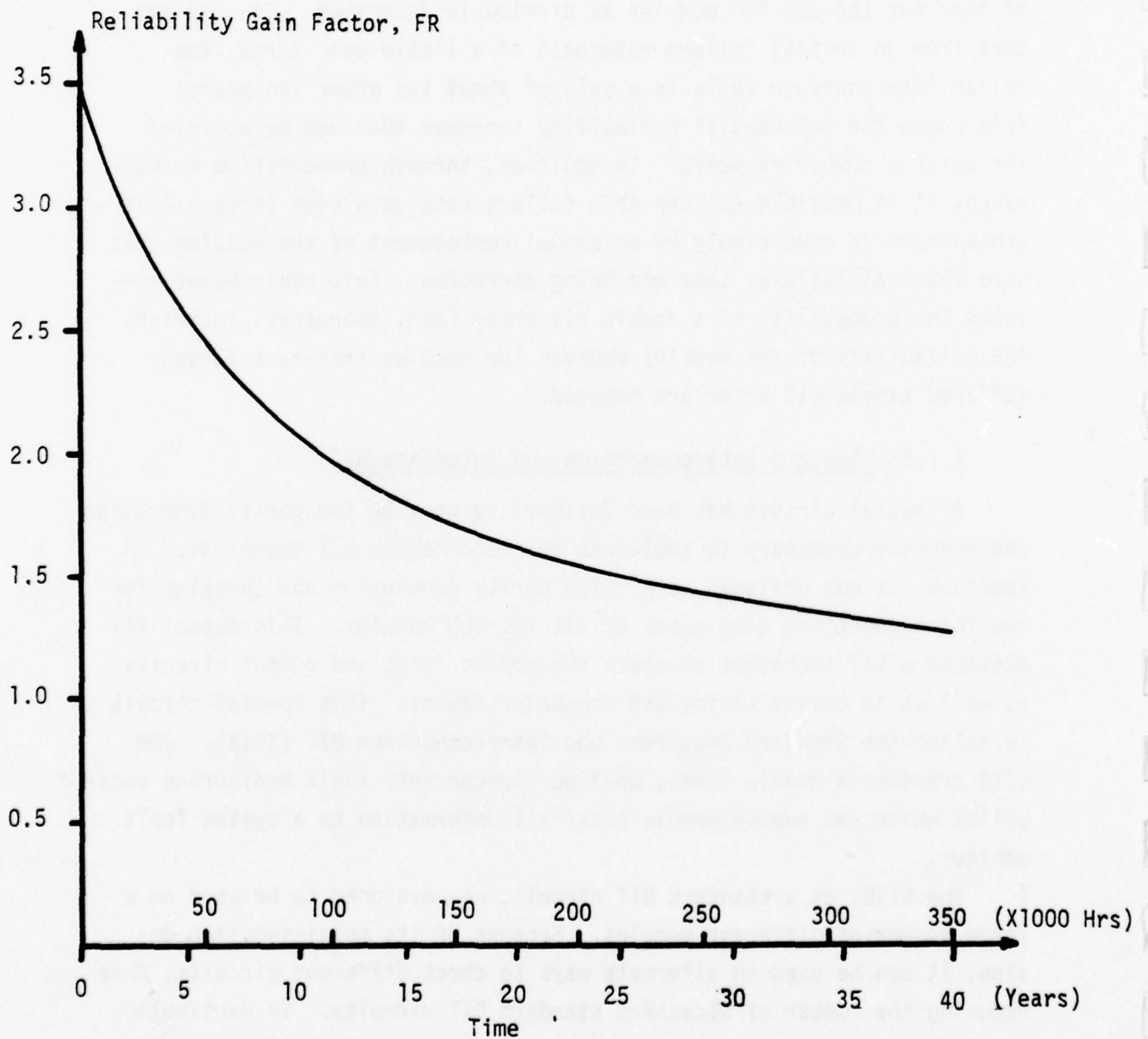


Figure 4.12 Reliability Gain Using Error Correction

On certain QED modules the SIIB also provides part of the error detection capability for the functional portion of the module as in the case of the Random Access Memory (RAM) module. The BIT functional approach embodied in the SIIB, is to check parity on incoming data and to check and generate parity for outgoing data. The following section presents a detailed functional description of the SIIB.

4.1.5.1 Functional Description

A block diagram of the SIIB circuit is shown in Figure 4.13. It consists basically of two 8-bit odd parity generators and checkers. This BIT circuit is used to check incoming data parity and to verify the performance of the input latches as shown in Figure 4.14. When used in this manner the P/F_a line indicates the results of a comparison of the parity of A Data and the transmitted parity bit. A failure of the comparison indicates a fault in either the logic card connector or in the intermodule connecting wiring. The P/F_b line indicates the results of the comparison of the parity of the output data of the latches and the parity of B Data. Thus, P/F_b verifies the proper operation of the input data latches. For timing purposes, these comparisons are made only when the latch is enabled. This approach not only indicates when errors occur in the interconnection wiring and the input latches, but it facilitates fault localization by distinguishing between such faults.

In general, the parity-out line is not used. However, in the memory modules and certain I/O modules the parity-out data can be used to check the functional portion of the module. In this case, the SIIB circuit takes the place of a parity generator. Also, in this application, the SIIB aids fault localization.

The SIIB circuit can also be used to check the output buffers as shown in Figure 4.15. When applied in this manner, the P/F_a line in general has no meaning and is, therefore, left unconnected. The P/F_b line gives an indication of the comparison of the data parity bits on each side of the output buffer. This comparison is made only when data is enabled out. A failure of the comparison denotes a failure in the output buffers. In addition to checking the output buffers, the SIIB

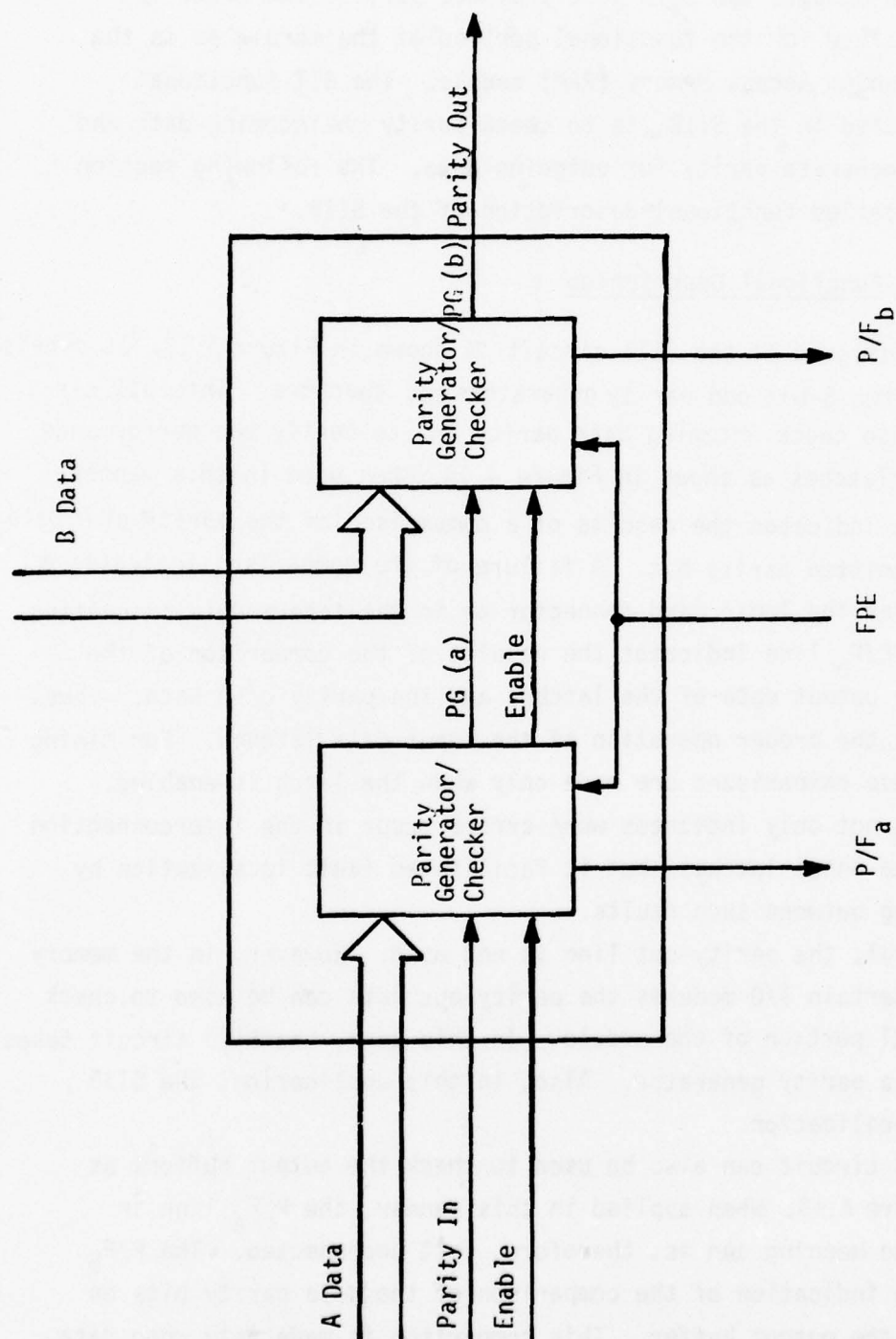


Figure 4.13 SIIB Functional Block Diagram

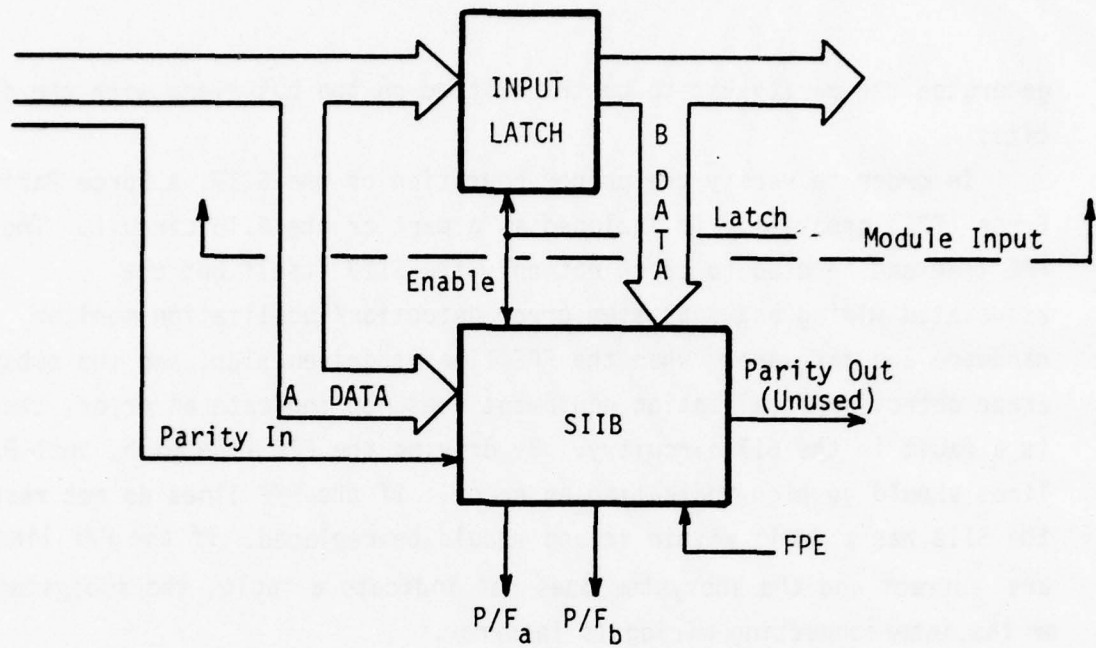


Figure 4.14 Input Checking

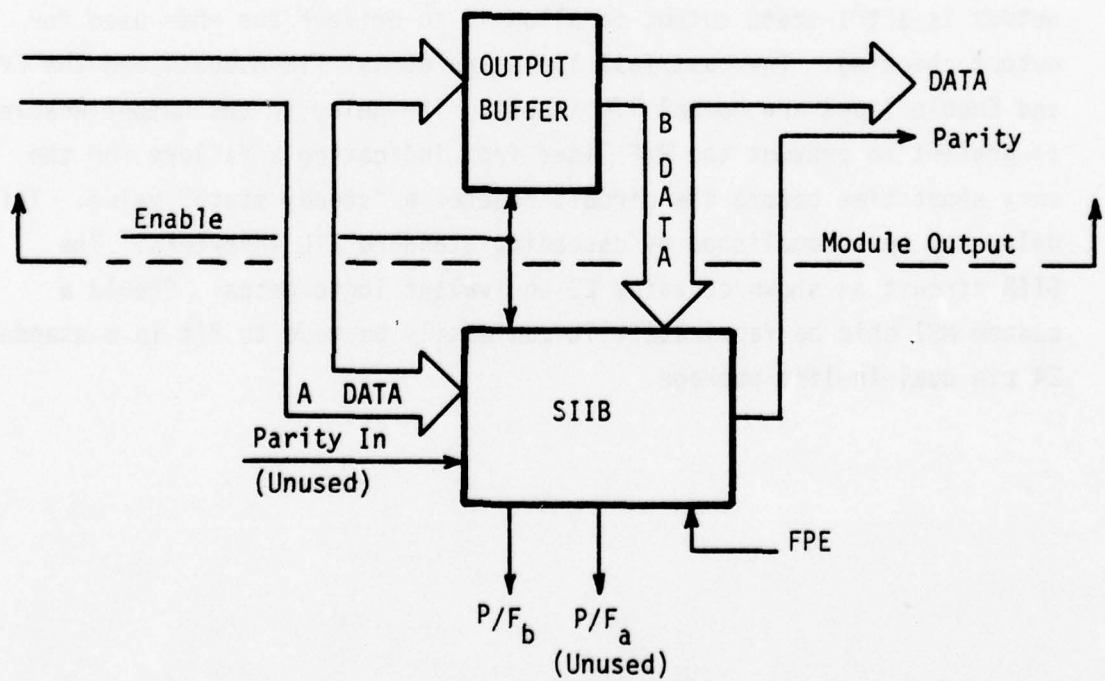


Figure 4.15 Output Checking and Parity Generation

generates the parity bit to be transmitted on the bus along with the data bits.

In order to verify the proper operation of the SIIB, a Force Parity Error (FPE) capability is included as a part of the SIIB circuit. The FPE line can be used to check not only the SIIB itself but the associated wiring and subsystem error detection/localization monitor hardware and software. When the FPE line is driven high, and the subsystem error detection/localization equipment does not indicate an error, there is a fault in the BIT circuitry. By driving the FPE line high, both P/F lines should go high indicating an error. If the P/F lines do not respond, the SIIB has a fault within it and should be replaced. If the P/F lines are correct and the subsystem does not indicate a fault, the subsystem or the interconnecting wiring is in error.

4.1.5.2 Logic Diagrams

Figure 4.16 is a gate level logic diagram of the SIIB. The B-parity output is a tri-state output to allow it to drive a bus when used for output checking. The pass/fail lines are normal TTL outputs and the FPE and Enable lines are normal TTL inputs. The delay to the output enable is present to prevent the P/F lines from indicating a failure for the very short time before the circuit reaches a "steady state" value. This delay may be accomplished by cascading standard TTL inverters. The SIIB circuit as shown contains 22 equivalent logic gates. Should a custom MSI chip be fabricated, it can easily be made to fit in a standard 24 pin dual in-line package.

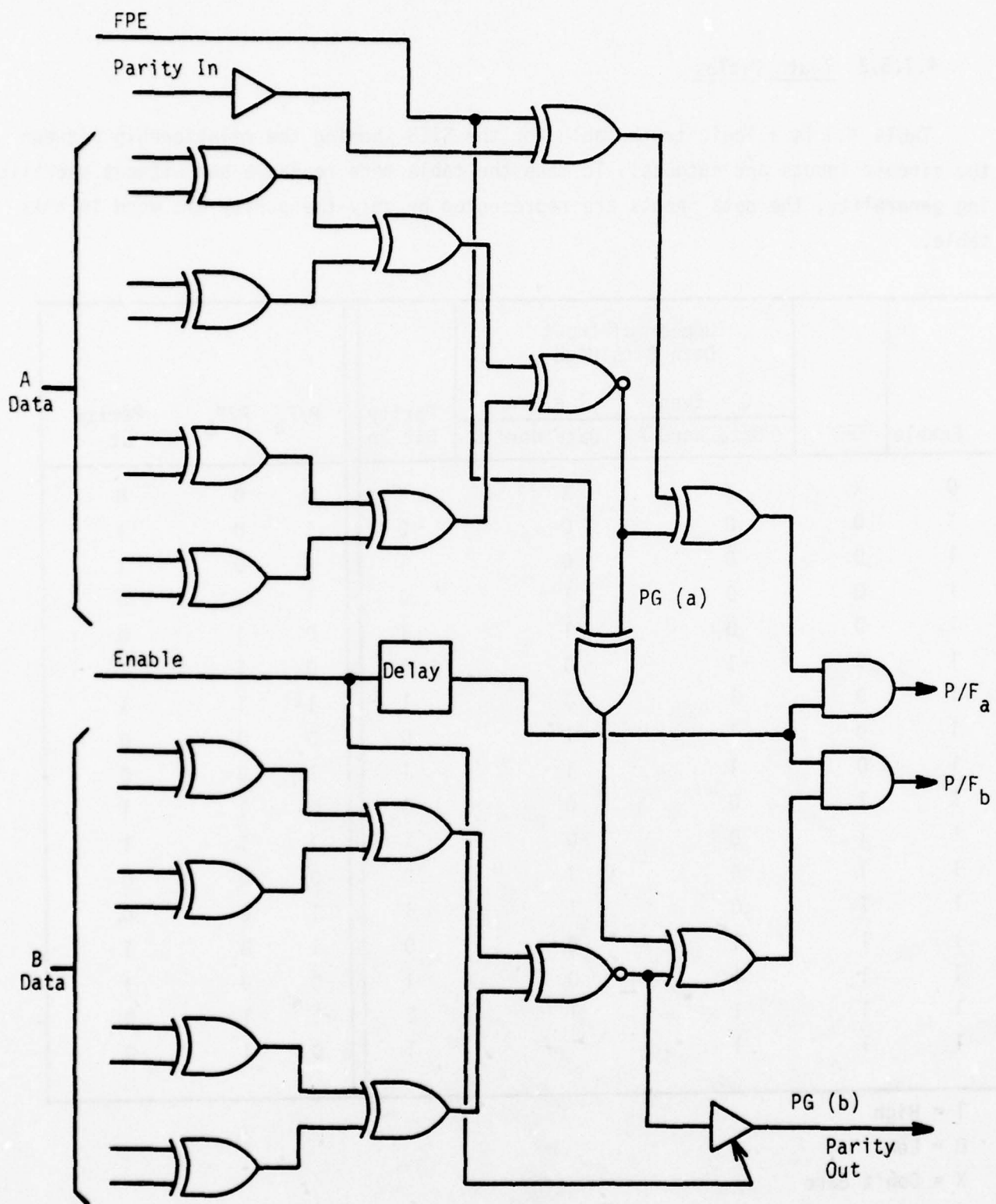


Figure 4.16 SIIB Logic Diagram

4.1.5.3 Truth Tables

Table 4.1 is a logic truth table for the SIIB showing the relationship between the circuit inputs and outputs. To make the table more readable and without sacrificing generality, the data inputs are represented by only the parity bit word in this table.

Enable	FPE	Number of Input Data Bits High		Parity Bit In			
		0 = Even	1 = Odd		P/F _b	P/F _b	Parity Out
		Data Word A	Data Word B				
0	X	X	X	X	0	0	H
1	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	0	1	1	0	1	0
1	0	1	0	0	0	1	1
1	0	1	0	1	1	1	1
1	0	1	1	0	0	0	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	1	1
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	0
1	1	0	1	1	1	0	0
1	1	1	0	0	1	0	1
1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	0
1	1	1	1	1	0	1	0

1 = High

0 = Low

X = Don't care

H = High Impedence State

Table 4.1 Truth Table for SIIB

4.1.5.4 Circuit Implementation

The SIIB may be implemented with currently available off-the-shelf small and medium scale integrated circuits. An alternate approach would be to design a single custom MSI chip to realize the SIIB function. The characteristics of each of these implementation alternatives are given in Table 4.2. Although either option is theoretically possible, in a practical sense the single custom circuit is by far the most viable. Not only is the package count drastically reduced, but so is the failure rate (0.065 versus 0.187 failures in 10^6 hours) and the power dissipation (50 mw versus 400 mw, assuming it will be made in a low power Schottky version).

It should be noted that the proposed standard BIT circuit is a candidate high volume IC. That is, due to its universality, it may be used in a wide variety of applications and therefore can be produced in large quantities. The resulting advantage, of course, is that high volume ICs tend to be very inexpensive and candidates for multiple sourcing.

4.1.5.5 Critical Parameters

The only potentially critical SIIB timing problem involves the length of the delay between the input enable and the pass/fail output data valid period. The delay must be long enough to allow the latches and buffer outputs to become valid and for the propagation delay within the SIIB to take place. By delaying the outputs as indicated in Figure 4.16, the P/F lines never give "false alarm" spikes while the data is becoming stable.

The SIIB circuit adds no critical timing requirements to any of the operations of the modules. This results from the fact that the SIIB is basically a monitor and causes no processing to start or stop. It is recommended that the SIIB be implemented with an integrated circuit technology such as low power Schottky which minimizes gate loading and power consumption.

4.1.5.6 QED Module Test Equipment Requirements

The recommended SIIB circuit is self-checked through the use of the

SIIB	Implementation Using Existing MSI	Implementation Using Custom MSI
No. of Gates	48	22
Packages	4	1
No. of Pins	14 pins/package	24
Failure Rate	$0.387/10^6$ Hours	$0.265/10^6$ Hours
Power Dissipation		
Typical	750 mw	50 mw
Max	1200 mw	90 mw

Table 4.2 SIIB Implementation Alternatives

Forced Parity Error (FPE) input. When this input is driven high and the module is enabled, the P/F outputs indicate a fault. It is recommended that the module input and output SIIBs be checked separately by enabling only the input latch and then enabling only the output buffer. It is necessary to have a valid data word on the input of the SIIB when performing these tests. This simply means that data supplied to the module under test must be valid.

The QED module test equipment must also perform a check of the parity output. Actually, two tests are necessary to determine proper operation for a data word with both even and odd parity. Proper operation is defined by Table 4.1. On output buffers, the high impedance state must be verified by not enabling the output and performing standard electrical checks on the parity output. This testing can be performed simultaneously with the electrical testing of the output data. This would involve an additional process but no new procedure would be required.

4.1.6 Application of Analytic Measures to the Recommended BIT

To quantify the gains and costs of each of the BIT approaches it is necessary to apply the analytic measures described in Section 3. While these measures are not claimed to be the optimum measures to evaluate BIT, they do provide a good indication of the additional costs involved. These measures also provide a guide to the effectiveness of the BIT techniques.

A summary of the BIT evaluation for the memory class is shown in Table 4.3. One can see that the recommended BIT technique, word parity, is quite effective since it can detect an error in over 99% of the module's gates. On a package basis the BIT can detect an error in over 60% of the packs. The BIT can also detect errors from wiring and connector faults, which is not indicated by the numbers in this table. In addition the cost of word parity is low because, for each of the modules in the class, less than 20% of the module's failure rate, less than 30% of the module's packages (typically 23%), and less than 20%

PARAMETER	RAM	ROM	FIFO	RAM WITH ERROR CORRECTION	UNITS
Percent of Gates Monitored	99	99	99	86	%
Percent of Packages Monitored	83	80	63	61	%
Number of Cycles for Test	0	0	0	0	-
Ratio of BIT Packages to Total Module Packages	22	27	21	55	%
Failure Rate without BIT	3.5	2.2	1.7	3.5	/10 ⁶ Hr
Failure Rate with BIT	4.1	2.5	2.0	3.1*	/10 ⁶ Hr
Ratio of BIT F.R. to Total Module F.R.	13	17	13	-200**	%
Power Consumption of BIT	0.8	0.8	0.2	4.5	Watts
Ratio of BIT Power Consumption to Total Module Power Consumption	11	19	5	43	%

* Comparable Failure Rate at a Reliability of $e^{-1} = .367879$

** For one year, see Section 4.1.4 for more complete discussion

Table 4.3 Analytic Measures Tabulation for Memory Class

of module's power consumption is accountable to BIT. These are quite reasonable costs for the level of error detection achieved. It is important to note that word parity is concurrent fault detection technique.

Also included in Table 4.3 is the BIT evaluation of the RAM module using single bit error correction with double bit error detection described in Section 4.1.4. This provides a direct comparison of this coding technique with the parity coding technique. It appears that error correction is a desirable approach when designing memory modules. For approximately twice as many packages, twice the power consumption and a slight decrease in monitoring capability, the error correction technique achieves a reliability three times better than a similar module with parity. There are always physical limitations, such as board space and power availability that may preclude the use of error correction on a particular module, but the advantages and disadvantages should be carefully examined before reaching a final conclusion on memory systems.

Appendix B gives a detailed package description of the particular QED Memory Class modules with the recommended BIT. The data necessary to compute the analytic measures are also given in Appendix B.

4.2 Built-In-Test for Process Class Modules

4.2.1 Definition of Process Class

The process class modules consist of combinatorial and sequential logic configured to store operands, perform arithmetic computations, manipulate data and output resultants. The process class QED modules are

Parallel Multiplier,
Arithmetic Logic Unit,
8-bit Index Counter,
Microprocessor Modules.

The process class modules are distinguished by their ability to perform arithmetic and logic operations on data in 2's complement representation. These modules are further characterized by their high speed capabilities. The process class modules are organized to accept 8-bit words and are generally expandable in 8-bit increments.

The recommended approach to built-in-test of the process class modules is to check the module interface circuitry using the standard I/O parity approach described in Section 4.1.5 and to use arithmetic coding techniques for checking the module's computational circuitry. In the following discussion it will be shown that a large percentage of the total circuitry on each of the QED process class modules can be checked using I/O parity and arithmetic codes. It will also be shown that the modules can, for the most part, be checked concurrently (on-line) in their standard operating mode using these techniques.

In some instances the percentage of circuitry checked concurrently may not be as high as warranted by the particular system application. In such cases provisions can be made at the system level to facilitate off-line testing at the option of the system designer. It is the intent in providing an off-line BIT alternative, to maximize the percent of each QED function tested while minimizing the hardware and software as well as the off-line processing time necessary to do a thorough job of checking each module. An example of this approach is the microprocessor module which will be discussed later in this section.

The following discussion considers some particular arithmetic coding options which may be used to check the arithmetic circuitry on the process class modules. Following a general discussion on applicable arithmetic coding techniques, specific circuitry for checking the arithmetic functions on each of the process class modules is presented. The reader is referred to Section 4.1.5 for a discussion of the standard I/O parity portion of the tests which is included on each module.

4.2.2 Approach to Process Class Modules

Various techniques for checking computational modules have been considered. Of particular interest are the coding techniques which are widely discussed in the literature. This discussion considers some of these techniques and singles out those most applicable to functional module built-in-test.

4.2.2.1 Arithmetic Coding Theory

The theory of arithmetic codes useful for error detection and correction is discussed extensively in the literature with the classic works being Peterson and Brown [5] and Peterson [6]. Other significant work is presented by Szabo and Tanaka [7]. All of these works present derivations based primarily on number theory. Theorems are given which are useful in quantifying the error detecting and correcting properties of various arithmetic coding schemes. However, techniques for the reduction to practice of the arithmetic coding schemes presented in the references cited above are not so widely understood. The application of coding schemes to both combinatorial and sequential logic circuits which use weighted number systems is presented in works by Szabo and Tanaka [7] and Ramamoorthy and Han [8].

Much of the theoretical work on arithmetic codes deals with the use of residue numbers systems which have inherent error detection and correction capabilities. Examples of computational schemes which use residue arithmetic are given by Garner [9], Watson and Hastings [10], and Barsi and Maestrini [11]. While such approaches are theoretically attractive, their use with the QED process class modules is not practical since the QED modules are designed to perform arithmetic operations using a weighted

binary number system (i.e., 2's complement arithmetic). The main distinction between the approach presented in the works referenced above and that presented in this report is the assumption in the present work that only weighted binary arithmetic computations are performed and that error checking is not necessarily an integral part of the basic arithmetic used.

The primary thrust of the process class module BIT approaches presented in this report is directed toward concurrent fault detection (on-line error checking). However, it is well known that powerful mathematical approaches for off-line fault detection do exist and would be useful for QED module acceptance testing. In particular, the calculus of D-cubes, as described by Roth [12], Roth, Bouricins and Schneider [13] and Putzola and Roth [14] is one off-line testing scheme having a strong mathematical basis. In addition, cyclic codes have been investigated for the Advanced Avionics Fault Isolation System (AAFIS) and are reported by Benowitz et al. [15] as an off-line module testing scheme.

Other useful, and at the same time, less mathematical treatments of built-in-test approaches have been presented in the literature. One such treatment with emphasis on the use of redundancy is given by Dandapani and Reddy [16]. A non-mathematical treatment which addresses the question of built-in-tests for LSI is given by Williams and Angell [17]. Their approach centers around the placement of test points throughout the LSI chips. However, the same ideas can be applied to MSI logic within a circuit module.

Because of the extensive work which has been done in the area of arithmetic coding and is available in the literature, no attempt will be made in this report to document all arithmetic coding approaches. Instead, only those techniques and codes which are readily applicable to the problem of concurrent error detection and off-line testing of QED process class modules will be presented. It is important to note that in this study "applicability" is very nearly synonymous with "ease of implementation" since it is the intent of this BIT study to minimize the hardware required to test the QED modules.

The following sections of this report present some specific arithmetic coding approaches for built-in-test of the Process Class QED modules. Special emphasis is given to those candidate approaches

which are easiest to implement in hardware. Tradeoffs are made between ease of implementation and fault detection effectiveness. No attempt is made in the present study to evaluate applicable fault correction techniques for the process class modules.

4.2.2.2. Arithmetic Coding Techniques for Concurrent Error Detection

This section considers arithmetic coding approaches to built-in-test for the process class QED modules. Codes which are of special interest are those designed for checking weighted number systems since the QED modules use binary 2's complement arithmetic. For the binary number system x is represented by

$$x = 2^0 b_0 + 2^1 b_1 + 2^2 b_2 + \dots 2^{n-1} b_{n-1} \quad (4.16)$$

where the coefficients are either 0 or 1.

An alternate approach to built-in-test of the process modules is to use a number system with an inherent error checking capability. An example of such a system is the residue arithmetic approach reported by Garner [9], Walton and Hastings [10] and Barsi and Maestrini [11]. However, the application of residue number systems to the QED modules would require potential QED module users to learn a number system other than 2's complement. This approach is thus deemed undesirable because of the obvious impact on system level design. However, residue arithmetic theory can be applied to the QED process modules for concurrent error detection in the form of a separate code which is described in a later section of this report.

There are two basic approaches to on-line binary arithmetic error checking. These are referred to in the literature as separate and non-separate codes [18]. These approaches are described in the following sections of this report along with examples of each.

4.2.2.3 Non-separate Codes

Non-separate codes are those codes which are combined with data words in such a way as to require that separation between the two take place, i.e., additional processing, in order to recover the information bits which are being checked. Such codes have the advantage that they

can be transmitted as a part of the data and thus pass through the same hardware which they are checking. A general block diagram of a non-separate code implementation is shown in Figure 4.19. It may be observed from this block diagram that additional processing of the data is required before the information which is being checked can be recovered. An example of a non-separate code is the AN codes described by Peterson [6]. In such a code the data, N , is represented by AN where A is a constant chosen using a criteria which maximizes the error detecting capabilities and minimizes the hardware required. Another important characteristic of such a code is that the coded form of the sum of two numbers is the sum of the coded numbers, i.e.,

$$AN_1 + AN_2 = A(N_1 + N_2). \quad (4.17)$$

Also, it can be shown that

$$(AN_1) \cdot (AN_2) = A^2 [N_1 \cdot N_2] \quad (4.18)$$

Thus, AN codes require a multiplication step in the encoding process and division in the decoding process. Also, it is implicit in the encoding scheme that the arithmetic hardware being checked must have sufficient capability to handle the resulting larger word sizes without detrimental data overflow.

One important variation of the AN codes is the AN + B code [6]. This code allows the use of 2's complement arithmetic, for example. This follows from the fact that binary and 2's complement arithmetic are related by an additive correction term.

The effectiveness of the AN codes, like that of the residue codes which will be described in a later section, is directly dependent upon the choice of A . Specifically, the set of undetectable error magnitudes, $|E_m|$, for AN codes can be found from [18],

$$|E_m| = KA, K = 1, 2, \dots, [(r^n - 1)/A], \quad (4.19)$$

where A = check modulus,
 r = radix,
 n = number of bits.

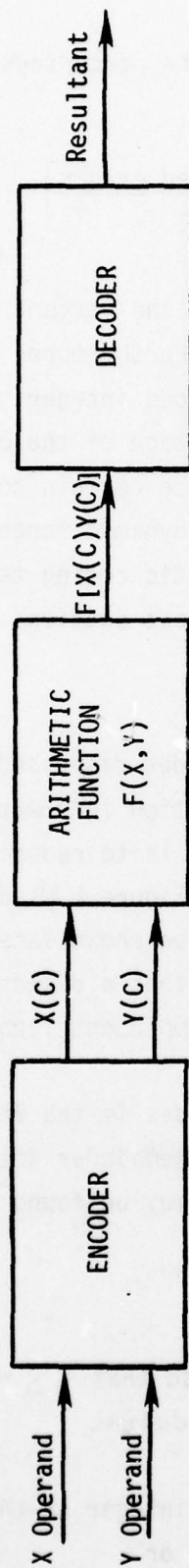


Figure 4.17 Non-separate Arithmetic Code Implementation

For radix-2 (binary) arithmetic, the percentage of single errors detected for AN codes can be found from

$$\% \text{ Checked} = \left[1 - \frac{\text{No. undetected errors}}{2^n - 1} \right] \times 100\%. \quad (4.20)$$

Table 4.4 presents a tabulation of the percent of combinations checked as a function of the check modulus A. Furthermore, Avizienix [18] points out that if the check modulus A is any odd integer, an AN code will detect all single bit errors and a high percentage of the burst errors.

As discussed earlier, AN codes place certain constraints on the system designer by limiting his usable dynamic range because of required operand product operations. An arithmetic coding technique which does not have this problem is presented in the next section.

4.2.2.4 Separate Codes

In contrast to the non-separate codes discussed in the previous section, separate codes treat the detection information as a separate entity. An immediate advantage of this is to reduce the impact of the built-in-test on the system designer. Figure 4.18 depicts a generalized separate code implementation. It will be shown later in the discussion of the mechanization of separate codes that a disadvantage is the fact that more hardware may be required to implement separate codes as compared to non-separate codes.

One of the most useful separate codes is the family of codes called residues. A residue is defined as the remainder after a division. The residue representation of an integer x may be found from

$$x = q m + r \quad (4.21)$$

where $q = \text{integer so that } 0 \leq r < m$
 $m = \text{check modulus.}$

The quantity, r, is the least positive integer of the division of x by m. It is called the residue x modulo m or

$$r = x \text{ modulo } m \quad (4.21)$$

CHECK MODULUS	% WORDS CHECKED
3	66
5	80
7	86
11	91
13	92

Table 4.4 Percent of Words Checked as a Function
of the Check Modulus

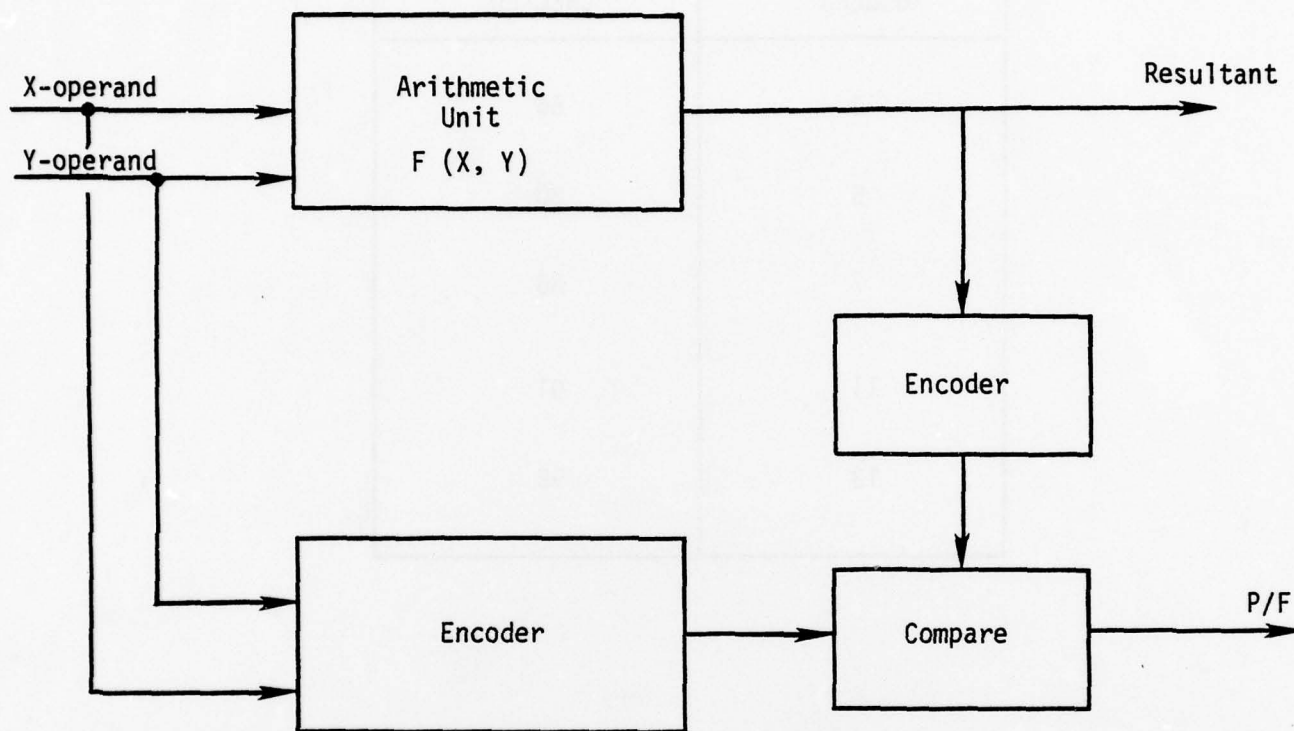


Figure 4.18 Separate Arithmetic Code Implementation

Many properties described above for the AN codes hold true for residue codes. Essentially, all of the AN code properties for misses (undetected errors) hold for residue codes. Likewise, all single bit errors are detected for odd moduli with residue codes. This fact coupled with the use of algorithms which can efficiently compute the residue resulting from modulo m division make residue codes quite attractive for arithmetic function error checking.

In order to evaluate the difficulty of the residue code mechanization problem, assume that an integer x is specified by

$$x = 2^n b_n + \dots + 2^2 b_2 + 2b_1 + b_0. \quad (4.23)$$

Using the property that the residue of the sum equals the sum of the residues yields

$$|x|_m = ||2^n|_m b_n + \dots + |2^2|_m b_2 + |2|_m b_1 + b_0|_m. \quad (4.24)$$

As an example, each term on the right hand side of equation 4.24 will be evaluated using $m = 3$. Results are presented in Table 4.5 for the 8-bit binary number, 11111111. For this example, x equals 255 while the sum of the residues equals 12. Taking each of these modulo 3 yields,

$$|255|_3 = |12|_3 = 0, \quad (4.25)$$

which is the desired result.

It is interesting to note from Table 4.5 that all odd powers of two terms have modulo 3 residues equal to 2 while all even powers of two have modulo 3 residues equal 1. This and similar properties will be exploited in order to realize low cost implementations of separate residue arithmetic fault detection codes.

An important class of systematic codes are the cyclic codes described by Peterson [6]. These codes are frequently used in peripheral memories such as disks and drums for error detection and correction. In addition to their widespread use in sequential access computer peripheral memory systems

b^n			$b^n \bmod 3$
1	x	$2^7 \longrightarrow 128$	2
1	x	$2^6 \longrightarrow 64$	1
1	x	$2^5 \longrightarrow 32$	2
1	x	$2^4 \longrightarrow 16$	1
1	x	$2^3 \longrightarrow 8$	2
1	x	$2^2 \longrightarrow 4$	1
1	x	$2^1 \longrightarrow 2$	2
1	x	$2^0 \longrightarrow 1$	1

Table 4.5 Modulo 3 Results for Each Bit of an 8-Bit Byte Number

these codes have been applied to random access memory systems where high reliability is important. One such application has been described by Toshi and Watambe [19] using a special case of cyclic codes called Hamming codes.

There are two distinct advantages of cyclic codes which should be mentioned. The first of these are their very well understood nature which can be described in concise mathematical form as coefficients of sets of primitive polynomials [6]. As a result, quantification of the error detecting capability of cyclic codes is easy. Secondly, binary cyclic and a small amount distributed memory is required.

However, while cyclic codes do a good job of detecting errors, they are most widely used in systems where error correction is required. As mentioned earlier, cyclic codes find widespread use in memory systems as concurrent error detectors/correctors. The application of cyclic codes in arithmetic fault detection has been mostly limited to off-line (non-concurrent) fault detection applications. This is the case with Hughes' AAFIS approach [15].

Of the arithmetic coding approaches described above including AN codes, cyclic codes, residue arithmetic and residue codes, the easiest to implement in terms of hardware is the residue code approach. A block diagram of the separate error detection code family of which residue codes are members is given in Figure 4.18. The generalized approach shown in this block diagram has been expanded and coupled with the SIIB and Standard Timer Process to become the built-in-test for the class modules.

Avizienis [18] has shown that any odd integer can be used to generate residues which will detect all single bit errors. It is felt that the detection of single bit errors is a reasonable objective for the built-in-test for the process class modules since it is desirable for the checking circuitry to be a small percentage of the circuitry being checked. It should be pointed out that the numbers Avizienis gives are for single words.

A particularly easy residue code to generate is the one using the odd integer, 3 (modulo 3 code). This is seen from Table 4.5 which illustrates that all even powers of 2 produce residues of unity and all odd powers of 2 have residues of value two. By recognizing this property and using the fact that the sum of the residues equals the residue of the sum [7], a residue generating algorithm like that shown in Figure 4.19 results.

A straightforward combinatorial logic implementation of the modulo 3 residue algorithm depicted in Figure 4.19 is given in Figure 4.20. This circuit works by interconnecting even powers of 2 to a residue 1 generator and odd powers of 2 to a residue 2 generator. The results of the residue generators are added together modulo 3 to form the final sum of the residues. This operation is performed on each operand and resultant. The coded operand results are multiplied together and compared with the residue of the resultant.

However, while the module 3 code is fairly easy to implement, it does have its limitations. In particular, it only detects single bit errors and as seen in Table 4.4 the modulo 3 residue code does not detect errors which are multiples of 3 (called misses). Table 4.4 shows that modulo 3 misses represent about 33% of all possible combinations. This percentage can be decreased at the expense of more complex circuitry since the residues resulting from odd integers greater than 3 are, in general, more difficult to generate.

4.2.3 Standardization of Process Class Module BIT

The circuits described in this section provide an error detection capability for the process portion of the QED modules. A modulo-3, separate residue arithmetic code was chosen because of its implementation advantages as discussed above.

Residue codes may be implemented as separate codes and therefore do not affect the design of the system in any detrimental manner. The basic approach is to compute the residue of each of the operands of a module and the residue of the answer. The module process operation is performed on the residues of the operands and then compared to the residue of the answer. Using modulo-3 residues, all single bit errors are detected.

AD-A056 147

RESEARCH TRIANGLE INST; RESEARCH TRIANGLE PARK NC SYST--ETC F/G 14/2
A STUDY OF A STANDARD BIT CIRCUIT.(U)

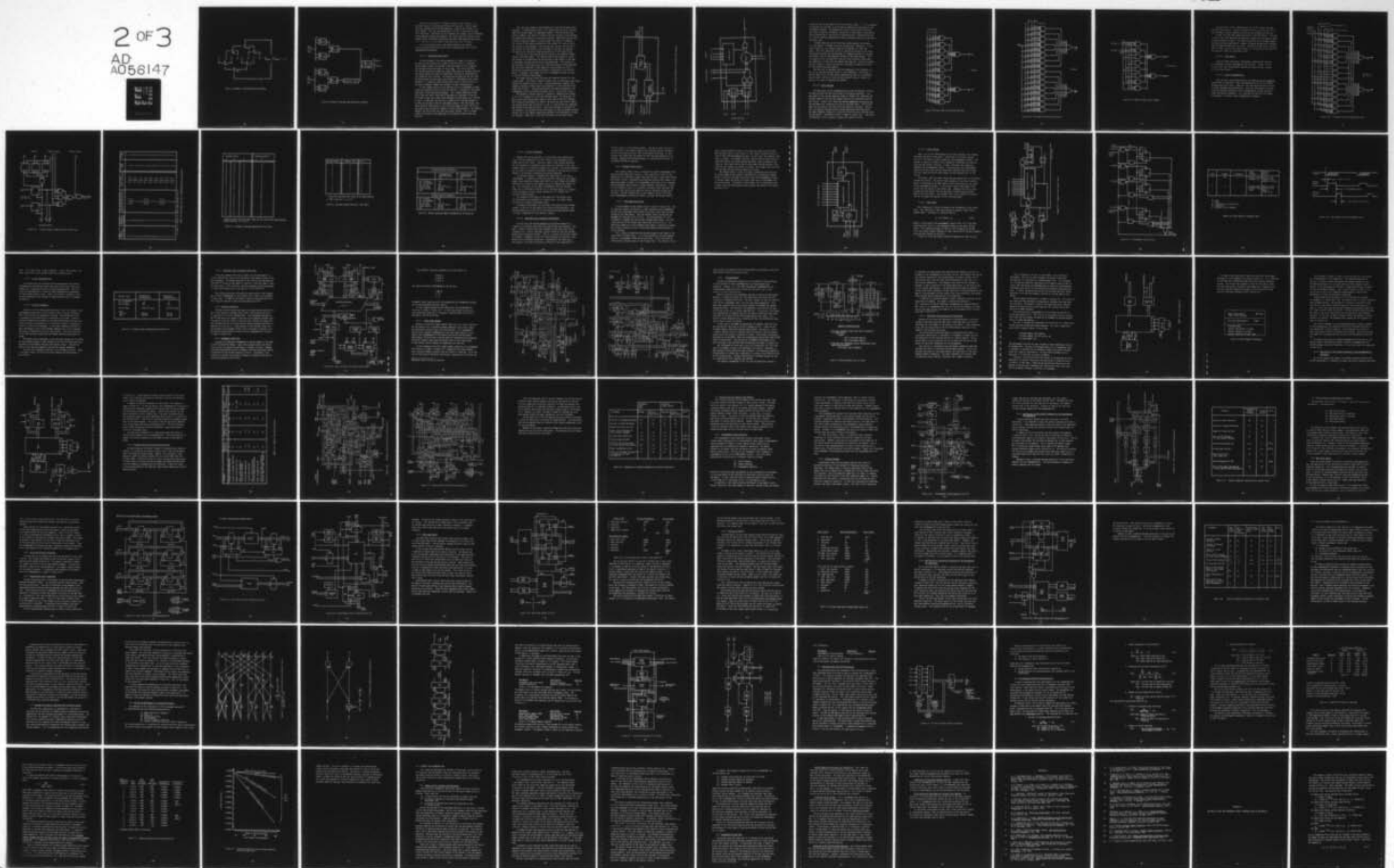
FEB 77 J B CLARY, J W GAULT, S J WEIKEL
RTI-430-1269

N00163-76-C-0231
NL

UNCLASSIFIED

2 OF 3

AD
A056147



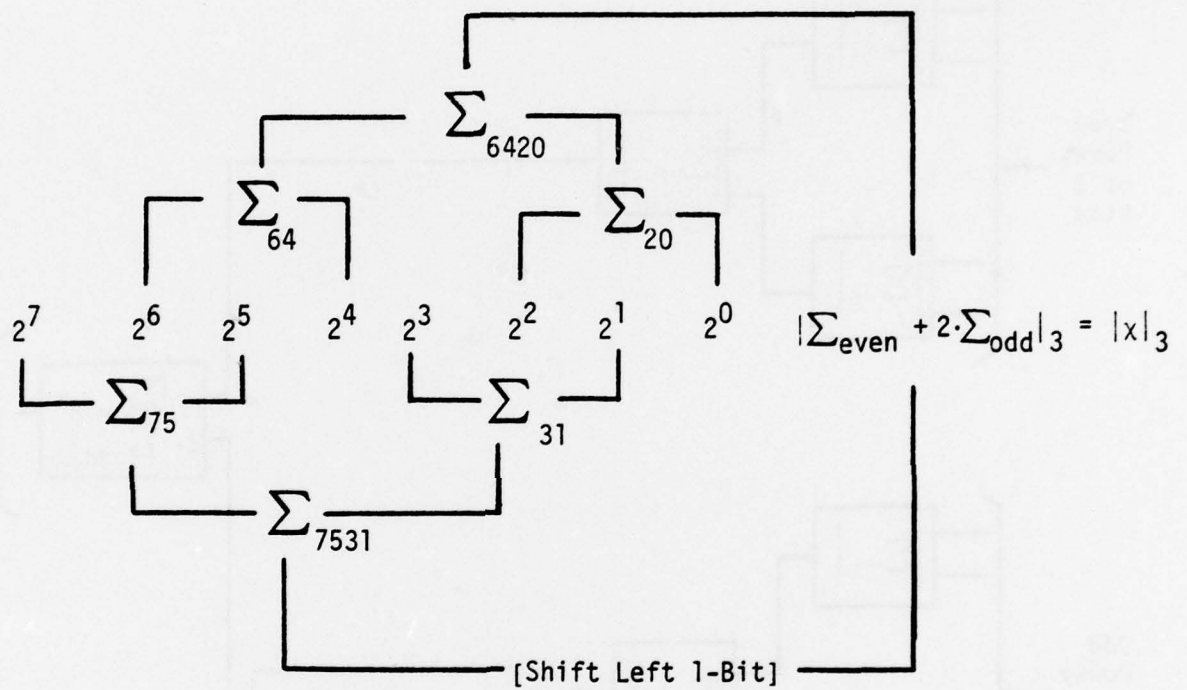


Figure 4.19 Modulo 3 Code Generation Algorithm

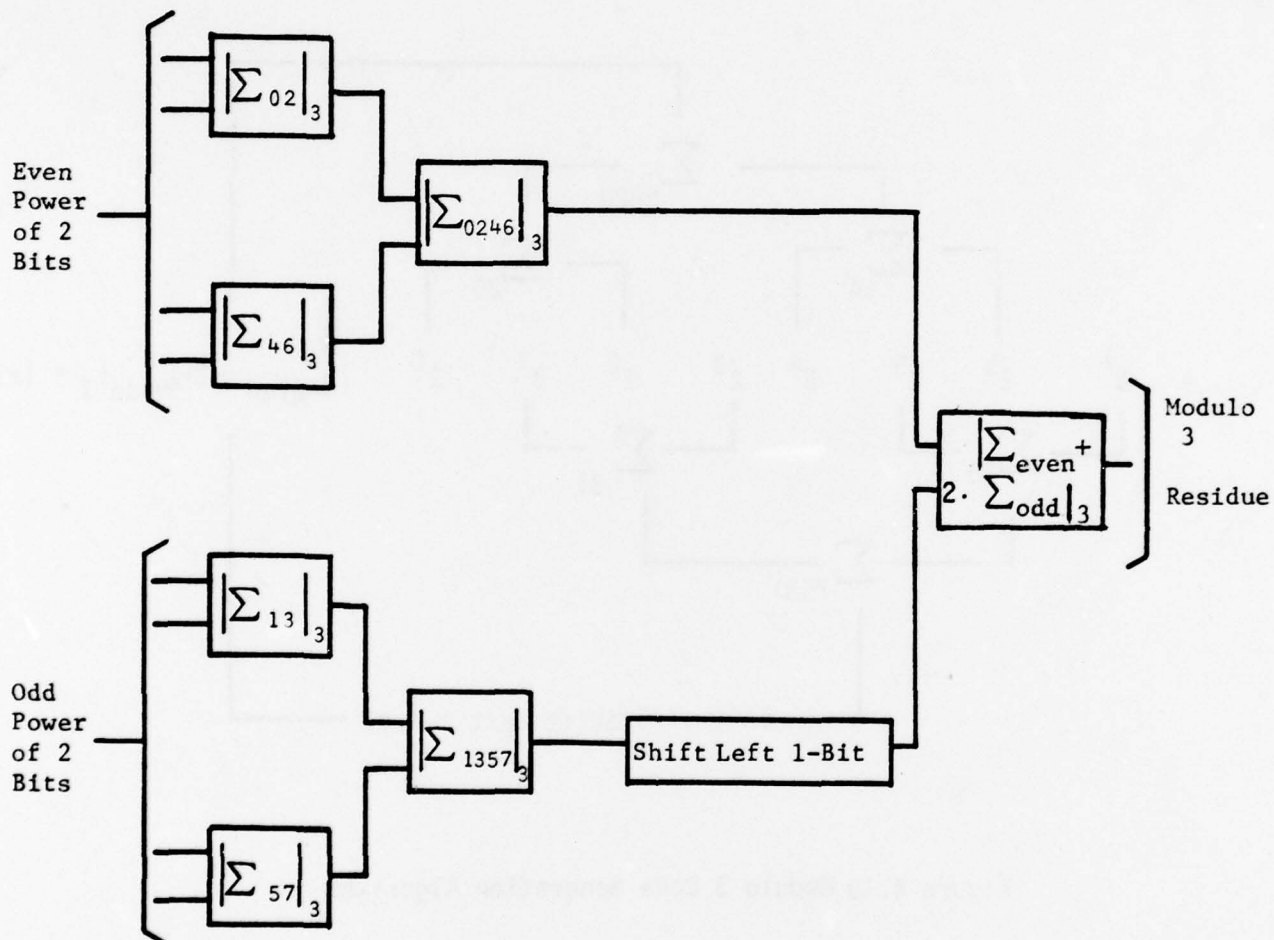


Figure 4.20 Modulo 3 Residue Code Generation Hardware

Because of the number of different potential applications, it is desirable to split the testing function into two parts. One of these circuits, denoted as the residue generator, computes the residue of an 8-bit number. One such residue generator circuit is required for each 8-bit data word. The second circuit computes the product, sum or difference of the operand residues and compares this result with the residue of the product sum or difference computed by the normal QED module process operation. The result is displayed by the module pass/fail indicator.

In the following section, the residue generator and checker circuits are described functionally and a proposed gate level circuit implementation presented.

4.2.3.1 Functional Description

A block diagram of the residue generator is shown in Figure 4.21. As shown in the diagram, use is made of the fact that in a weighed binary number system, the residue of each odd power of two is two and the residue of each even power of two is one. This fact simplifies the residue generation scheme and reduces the amount of hardware required. Once these partial residues are determined, they may be added modulo-3 to produce the total residue. Allowance must be made for the fact that the QED modules use both binary and signed 2's complement binary representation. A scheme has been devised which allows a single circuit to be used to properly compute the residue of either representation.

In this scheme, which will be referred to as 2's complement residue (TCR), the residue of all bits but the sign bit is computed in the manner described in the preceeding paragraph. This residue computation makes use of the fact that a digit which represents an odd power of two has a residue of two, and a digit that represents an even power has a residue of one. In addition, the sign bit is added to this residue (modulo-3) to form the residue of the 2's complement number in the TCR scheme. Because the most significant bit of a positive number is zero, the TCR is the same as the residue of a number in binary representation. For negative numbers, the TCR is one less than the binary residue. This in effect yields the residue of the magnitude of the negative number times two modulo-3.

This last point leads to the mathematical justification behind this approach. To understand why this is so, examine the four cases involved when multiplying signed 2's complement numbers. When multiplying two positive numbers, the result is positive and the residues of the operands and product are identical to the binary residue. When multiplying a positive and a negative number (in either order) the product is negative. The residue of the negative operand and the product will each be twice the residue of the magnitude. Multiplying each residue (the residue of the product and the product of the residues) by two does not destroy the equality nor the error detecting properties of the residue code.

The last case involving the multiplication of two negative numbers yields a positive product. The residue code of each operand is twice the residue of the magnitude and the residue of the product is the same as the binary residue (the residue of the magnitude). The product of the residues is two times two or four times the residue of the magnitude. But since this operation is done modulo-3, multiplication by four is equivalent to multiplication by one which is equivalent to no multiplication. Therefore, the product of the 2's complement residue is equal to the TCR residue of the product.

The second residue circuit provides the checking function for the residue codes. The circuit, shown in Figure 4.22, in functional block form, accepts the outputs from up to four residue generators, computes the algebraic residue of these inputs and compares it to the residue of the normal QED function. Timing considerations have been considered in the recommended circuit to provide an output free of "false alarm" spikes.

The algebraic residue function is capable of computing the residue of $(A \cdot B) + K$ where A, B, and K are residue inputs. By using only inputs A and B and setting the K input to zero, the circuit will compute the residue of the product of A and B. By using only the A and K inputs and setting the B input to one ($B_1 = 0, B_2 = 1$) the circuit will generate the residue of the sum of A and K. Since this residue is available to the outside, it is possible to use this output to input to another residue checker and in this manner compute the residue of any combination of sums and products. When used in this mode the enable of the unused input

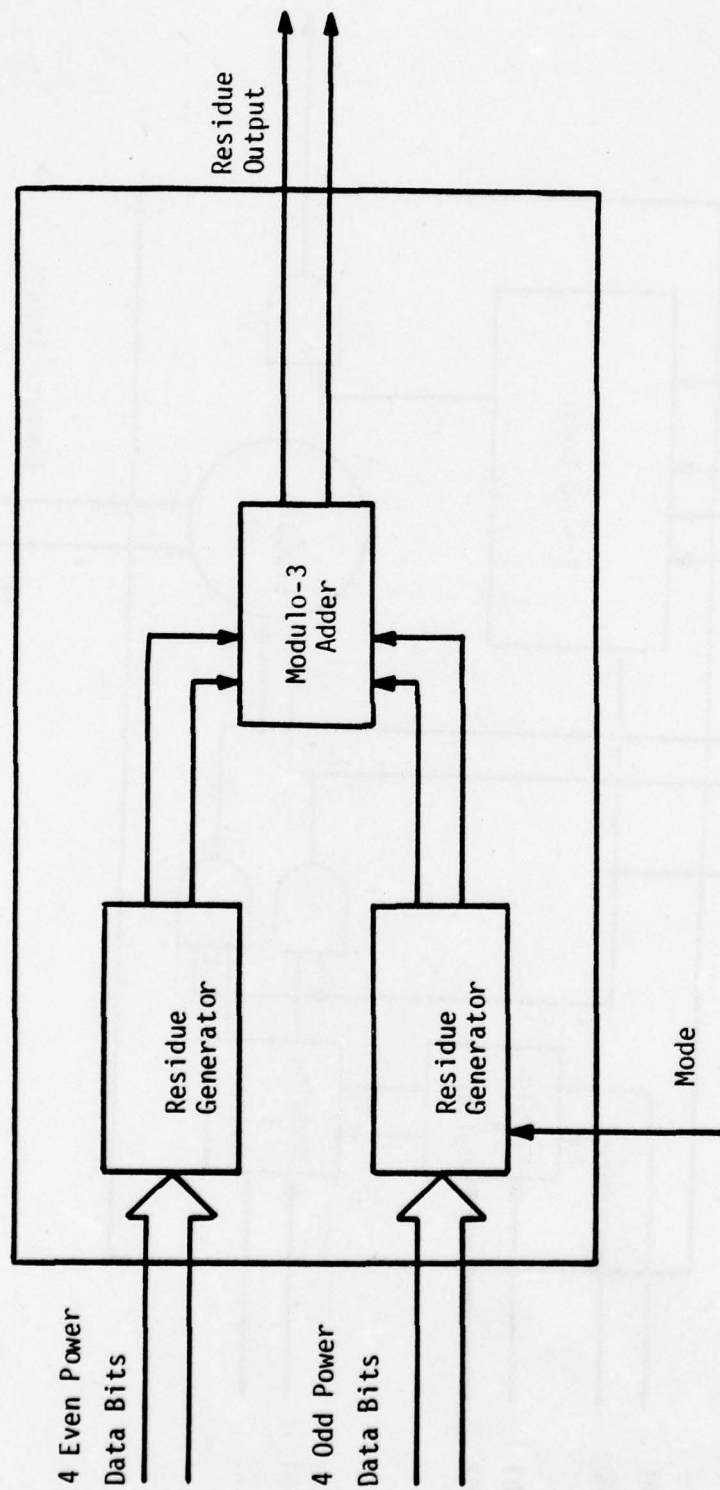


Figure 4.21 Residue Generator Functional Block Diagram

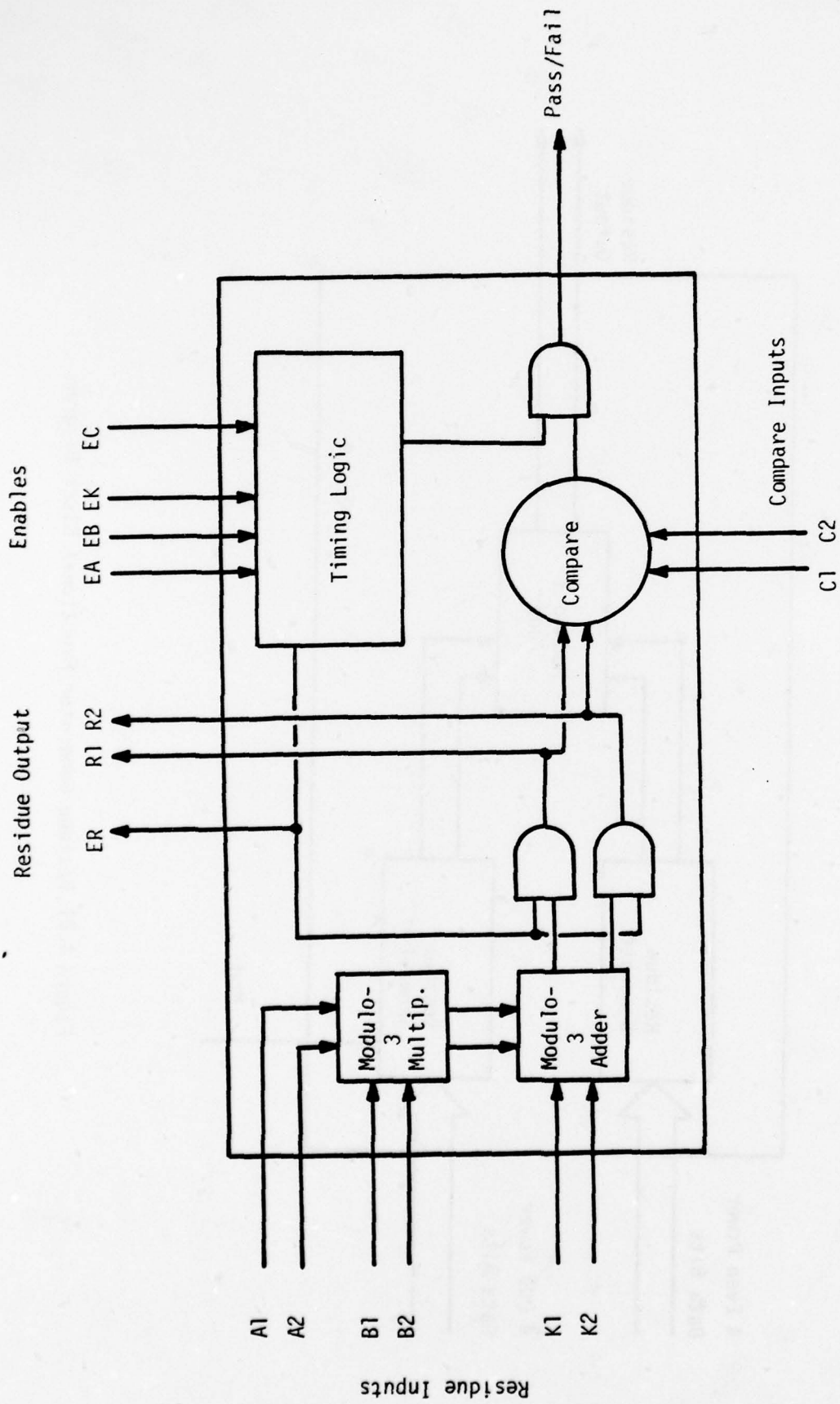


Figure 4.22 Residue Checker Functional Block Diagram

should be tied to the enable of one of the other inputs. If it is necessary to subtract two residues it can be done by adding with the digits of the subtrahend reversed. The mathematical justification results from the fact that in modulo-3 arithmetic, adding one is equivalent to subtracting two and adding two is equivalent to subtracting one. Reversing the bits of a two bit residue representation merely converts a two into a one and a one into a two.

Although there may be more than one residue checker on a module, only one checker is needed to provide the pass/fail indication. In the other checker(s), the compare inputs are connected to a logic one level. This will cause the circuit to indicate a failure for any input. The pass/fail output can then be used as a composite enable to indicate the proper timing of the generated residue to the other checker. This is necessary since the pass/fail line will only indicate a fault when the comparison is false and the enables have been set. Since the comparison will always be false, the pass/fail generates an output only after all the inputs have been enabled.

The fact that an improper input will generate a fail response, as well as an improper residue output, is very important. This allows a partial check on all of the residue generators as well as a convenient way to cause a failure in order to check the BIT circuit - which, in effect, is a means of checking the checker.

4.2.3.2 Logic Diagrams

The gate level circuit representations of the TCR generator circuit has been divided into the three blocks as discussed previously. The even power of two residue generator shown in Figure 4.23 generates the residue, modulo-3, and has two outputs that are not accessible outside the circuit. Figure 4.24 shows the residue generation realization for the odd power bits including the sign bit and mode input. This circuit again has two output lines that are not externally available. The two pairs of outputs from the partial residue generators described above are added together, modulo-3 in the third circuit block to form the final residue output. The addition block is shown in Figure 4.25. The entire TCR generator circuit contains 41 gates and requires 15 pins.

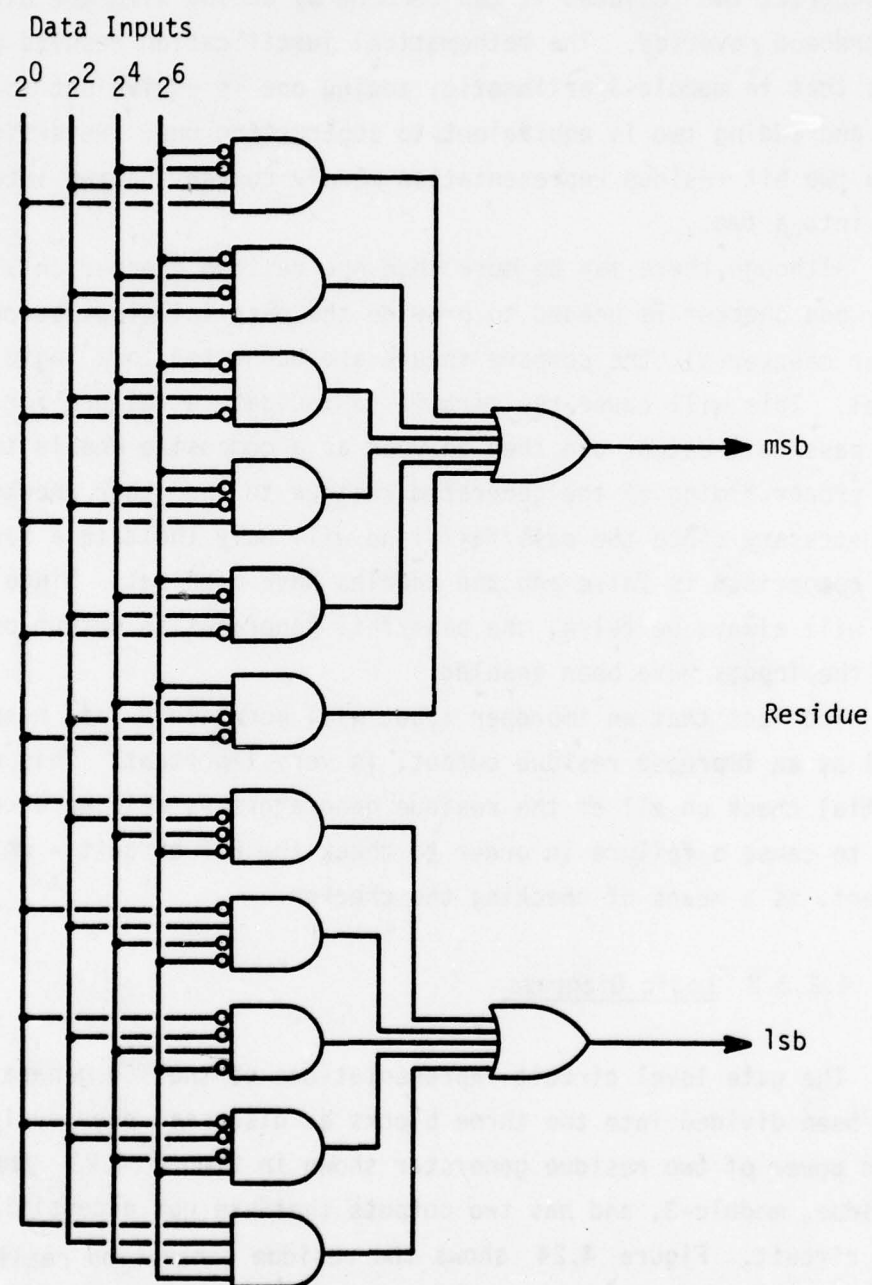


Figure 4.23 Even Power Bit Residue Generator

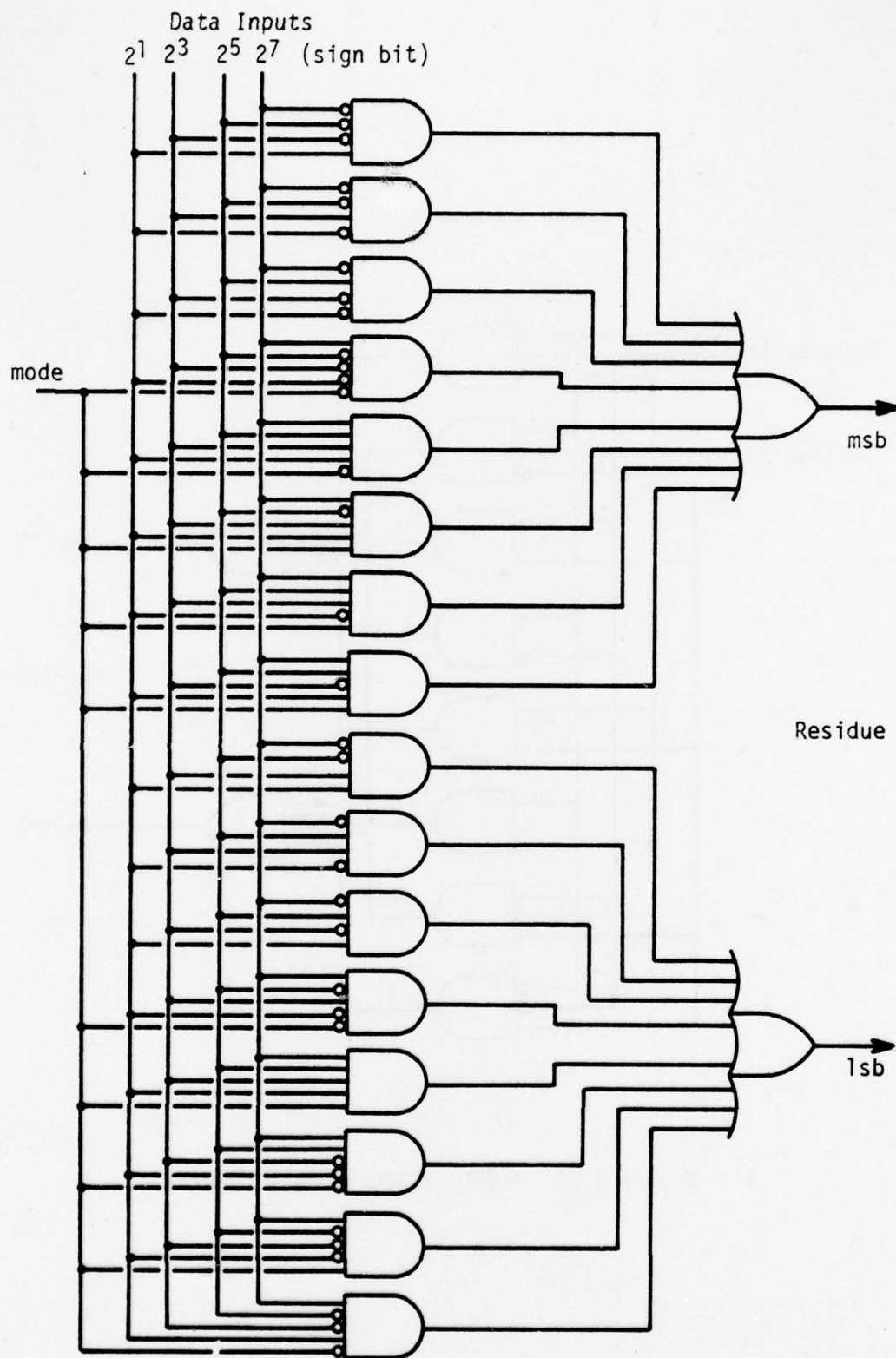


Figure 4.24 Odd Power Bit Residue Generator

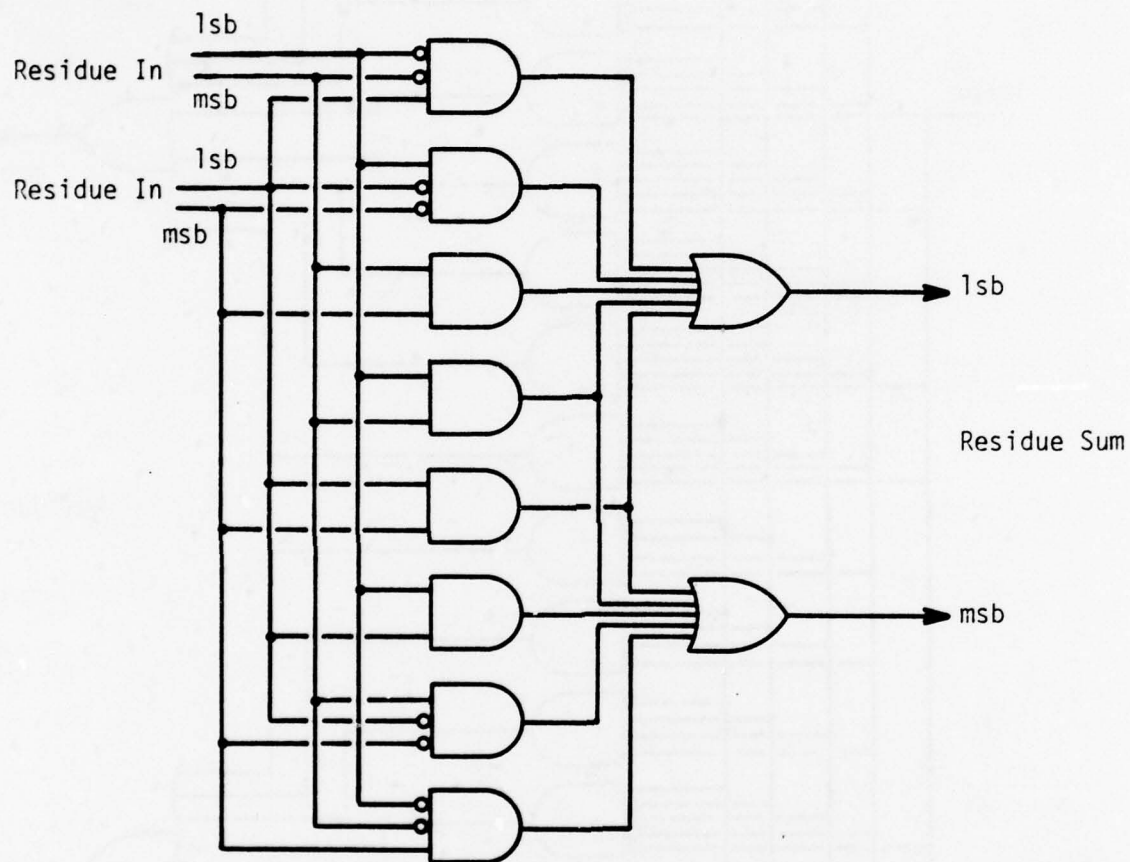


Figure 4.25 Modulo-3 Adder Logic Diagram

The gate level circuit representation of the TCR checker has been divided into two parts. Figure 4.26 shows the first part of the circuit. The arithmetic residue computation logic has two products and one sum.

Figure 4.27 shows the remaining portion of the TCR checker. Included is the circuit that generates the pass/fail output from the compare inputs and the arithmetic residue. The remaining portion of the residue checker provides the timing and control signals to enable the pass/fail indicator when the residue comparisons are valid. This part of the circuit contains three latches, one for each residue input, all of which must be set before the pass/fail output can indicate a fault.

4.2.3.3 Truth Tables

Table 4.6 shows the logical input/output relationships that the 2's complement residue code generator must satisfy. Tables 4.7 and 4.8 give the logic relationships between the inputs and the outputs of the residue checker.

4.2.3.4 Circuit Implementation

Like the SIIB, the residue generator and checker may be implemented using either off-the-shelf, small and medium scale ICs or with custom designed monolithic chips. Table 4.9 lists the characteristics of each of these implementation alternatives. Obviously the custom IC realizations are the most attractive from a package count, failure rate and power dissipation standpoint. As in the case of the SIIB, the custom IC approach is the most viable implementation approach.

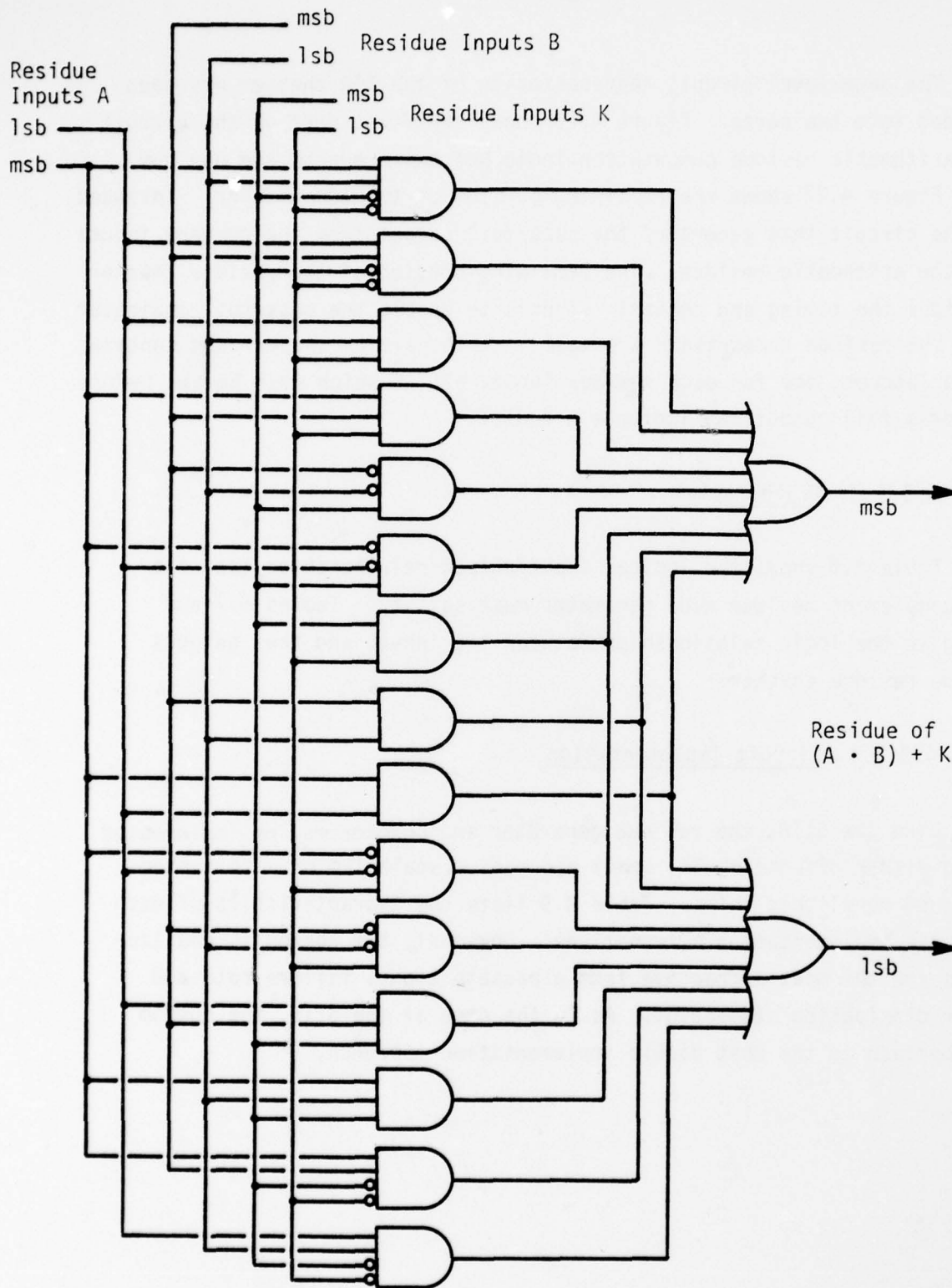


Figure 4.26 Arithmetic Residue Computation Logic

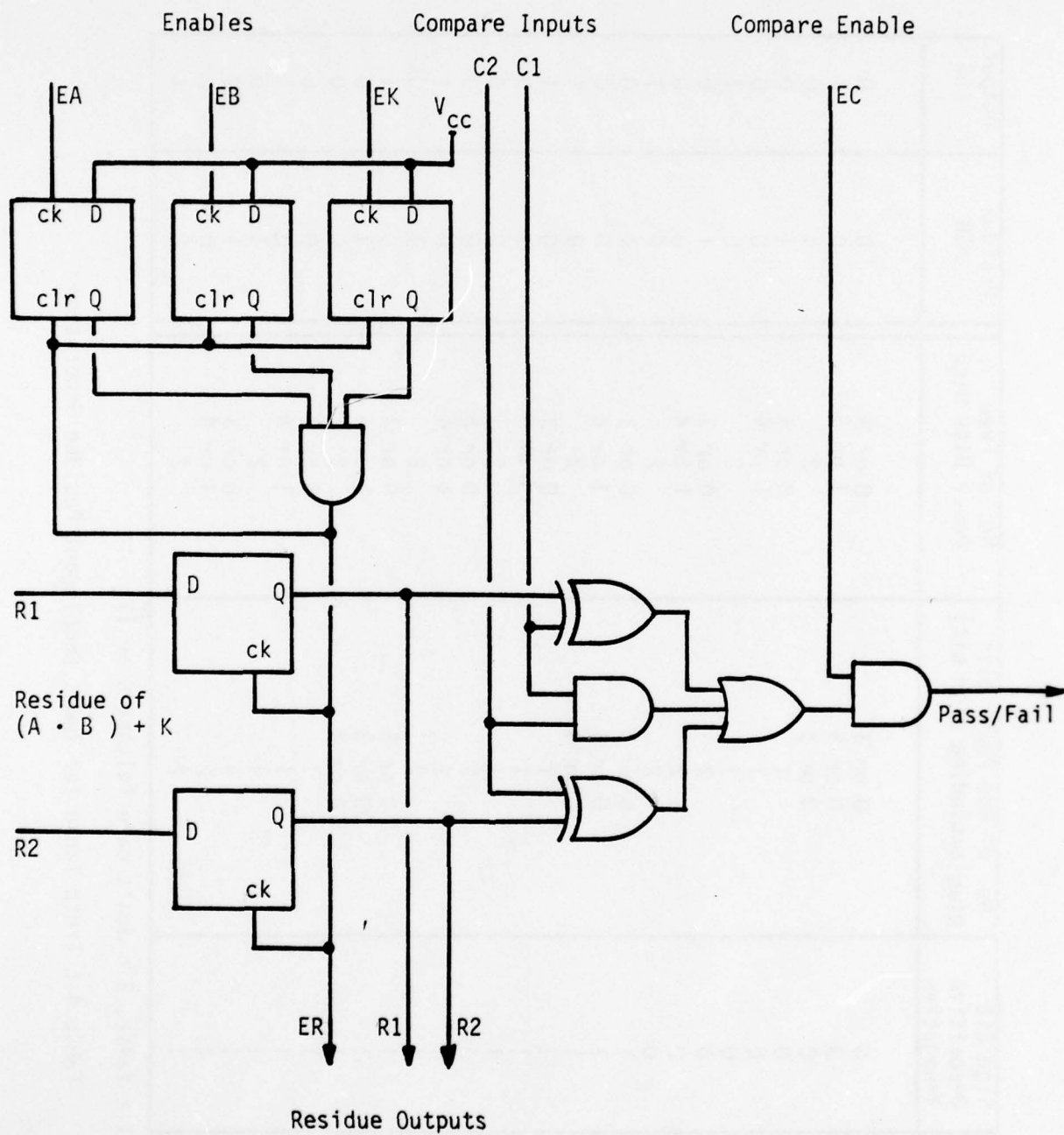


Figure 4.27 Residue Checker Comparison and Timing Logic

Mode 0=binary 1=2's comp	Sign Bit 0=positive 1=negative	No. of Odd Power Bits High(excluding sign bit)	No. of Even Power Bits High	Residue MSB	Output LSB
X	0	0 or 3	0 or 3	0	0
X	0	0 or 3	1 or 4	0	1
X	0	0 or 3	2	1	0
X	0	1	0 or 3	1	0
X	0	1	1 or 4	0	0
X	0	1	2	0	1
X	0	2	0 or 3	1	0
X	0	2	1 or 4	0	0
X	0	2	2	0	1
0	1	0 or 3	0 or 3	1	0
0	1	0 or 3	1 or 4	0	0
0	1	0 or 3	2	0	1
0	1	1	0 or 3	0	1
0	1	1	1 or 4	1	0
0	1	1	2	0	0
0	1	2	0 or 3	0	1
0	1	2	1 or 4	0	0
0	1	2	2	0	1
1	1	0 or 3	0 or 3	1	0
1	1	0 or 3	1 or 4	0	1
1	1	0 or 3	2	0	0
1	1	1	0 or 3	0	0
1	1	1	1 or 4	0	1
1	1	1	2	1	0
1	1	2	0 or 3	0	0
1	1	2	1 or 4	1	0
1	1	2	2	0	1

0 = True, 1 = false, X = don't care (either 0 or 1)

Table 4.6 Truth Table for Two's Complement Residue Generation

RESIDUE INPUTS			RESIDUE OUTPUTS *
A	B	K	R
3	X	X	3
X	3	X	3
X	X	3	3
0	0	0	0
0	0	1	1
0	0	2	2
0	1	0	0
0	1	1	1
0	1	2	2
0	2	0	0
0	2	1	1
0	2	2	2
1	0	0	0
1	0	1	1
1	0	2	2
1	1	0	1
1	1	1	2
1	1	2	0
1	2	0	2
1	2	1	0
1	2	2	1
2	0	0	0
2	0	1	1
2	0	2	2
2	1	0	2
2	1	1	0
2	1	2	1
2	2	0	1
2	2	1	2
2	2	2	0

* Residue Output available only after EA, EB, and EK have been enabled.
X=Don't Care (0, 1, 2, or 3)

Table 4.7 Arithmetic Residue Computation Truth Table

Residue Output	Compare Input	P/F *
X	3	1
3	X	1
0	0	0
0	1	1
0	2	1
1	0	1
1	1	0
1	2	1
2	0	1
2	1	1
2	2	0

* Pass/Fail available only after EC has been enabled.

X= Don't Care (0, 1, 2, or 3)

Table 4.8 Residue Checker Pass/Fail Truth Table

RESIDUE GENERATOR (MODULO-3)		
	Implementation Using Existing MSI	Implementation Using Custom MSI
No. of Gates	55	41
No. of Packages	30	1
No. of Pins	420	14
Failure Rate	$.737/10^6$ Hours	$.082/10^6$ Hours
Power Typical	865 mW	60 mW
Max.	1769 mW	100 mW
RESIDUE CHECKER (MODULO-3)		
No. of Gates	1060	47
No. of Packages	4	1
No. of Pins	60	18
Failure Rate	$.316/10^6$ Hours	$.084/10^6$ Hours
Power Typical	760 mW	60 mW
Max.	1215 mW	100 mW

Table 4.9 Residue Coding Hardware Implementation Alternatives

4.2.3.5 Critical Parameters

Because the residue generator is constructed using combinatorial logic, the only potentially critical timing is the propagation delay within the circuit itself. Since the inputs to the residue generators are not connected to the module input (they are connected to the latch outputs) there is no impact on module input loading. The residue generator has no inputs or outputs to the module interface so the operation of the module is not directly affected by the residue generator.

The residue checker is somewhat more complicated in the timing requirements. To ensure an error-free pass/fail indication, the compare enable input should be strobed after the A, B, and K inputs are enabled. Time must be allowed for the compare circuit to compute the proper output. This time delay can be easily implemented using the standard timer that is described in Section 4.2.4. The total delay is a function of the arithmetic circuitry being checked and is a constant for a particular module design.

The loading on the inputs of the module will be minimal since only the enables are connected to a module input. All other inputs are driven by internally generated lines.

A necessary requirement for the proper functioning of the residue checker is that all of the enables be cycled for each operation. The residue checker waits until all of the inputs have been enabled before it allows a computation of the pass/fail output.

4.2.3.6 QED Module Test Equipment Requirements

To fully test the residue generators and residue checker(s) on a module, all possible input data word combinations are necessary. However, a greatly reduced input test sequence of data words each having residues of 0, 1, and 2, exercises a major portion of both the residue generating and checking circuits. That is, if there are two 8-bit inputs to a module, a sequence of nine (3^2) input patterns are required. Each pattern has a different residue pair, determined by the input data. The pass/fail line should indicate no failures for any combination if

the BIT circuit is functioning properly. One way to force an error is to drive both of the compare inputs to the residue checker high which would in turn cause the fail line of the module to go high. In this process, the inputs must be enabled for each new data pattern for all testing. Because the residue checking is a passive built-in-test, no additional testing is required.

4.2.4 Standard Timer Circuit

The standard timer circuit is designed to provide programmable time delays for the other BIT circuitry. This capability is vital to the generation of a valid/pass signal that does not have "false alarm" spikes. While this circuit is designed to provide the timing of BIT signals, its generality makes it useful as a programmable oscillator or as a programmable monostable multivibrator in other potential applications. A goal of this standard circuit is to provide a universal timer that does not utilize a high failure rate capacitor and can therefore be made quite reliable. The basic design uses a high frequency crystal oscillator and a programmable divide-by-N counter to achieve the desired results.

4.2.4.1 Functional Description

A block diagram of the standard timer is shown in Figure 4.28. The two separate outputs provide both a signal delayed by the programmable divider and a monostable multivibrator (one-shot) signal that has been delayed by the same amount. The nine divider inputs provide the programmability required of this circuit. The desired delay (in 10 ns increments) is loaded into the divider through these inputs. The nine lines provide 2^9 or 512 delays that range from 10 ns to 5.12 μ s. An additional capability of the standard timer makes it possible to connect two or more of these circuits in series and thereby achieve delays as long as desired.

This is done by connecting the one-shot output of one timer to the clock input of another timer. It is also possible to use the standard timer as a programmable free-running oscillator. This is possible by connecting the delayed output to the trigger input. The function of the

mode (internal/external) input is to select the input to the divider. When the mode input is high (logic 1), the internal 100 MHz divider is used to drive the programmable counter. In this case, the external clock input is unused. The trigger input will then allow the clock pulses to reach the counter to provide the desired delay. When the mode is low (logic 0), the signal driving the divider comes from the external clock input and the internal oscillator is unused. The trigger input will perform the same function without regard to the mode.

The delayed output is used to trigger a monostable multivibrator. The pulse length of the output is determined by a divide-by-4 counter which is driven by the same clock that drives the programmable divider. In typical usage, the delay of circuits under test is programmed into the divide inputs. The module's input enable line is connected to the trigger input and the one-shot output provides the enable to the module pass/fail line.

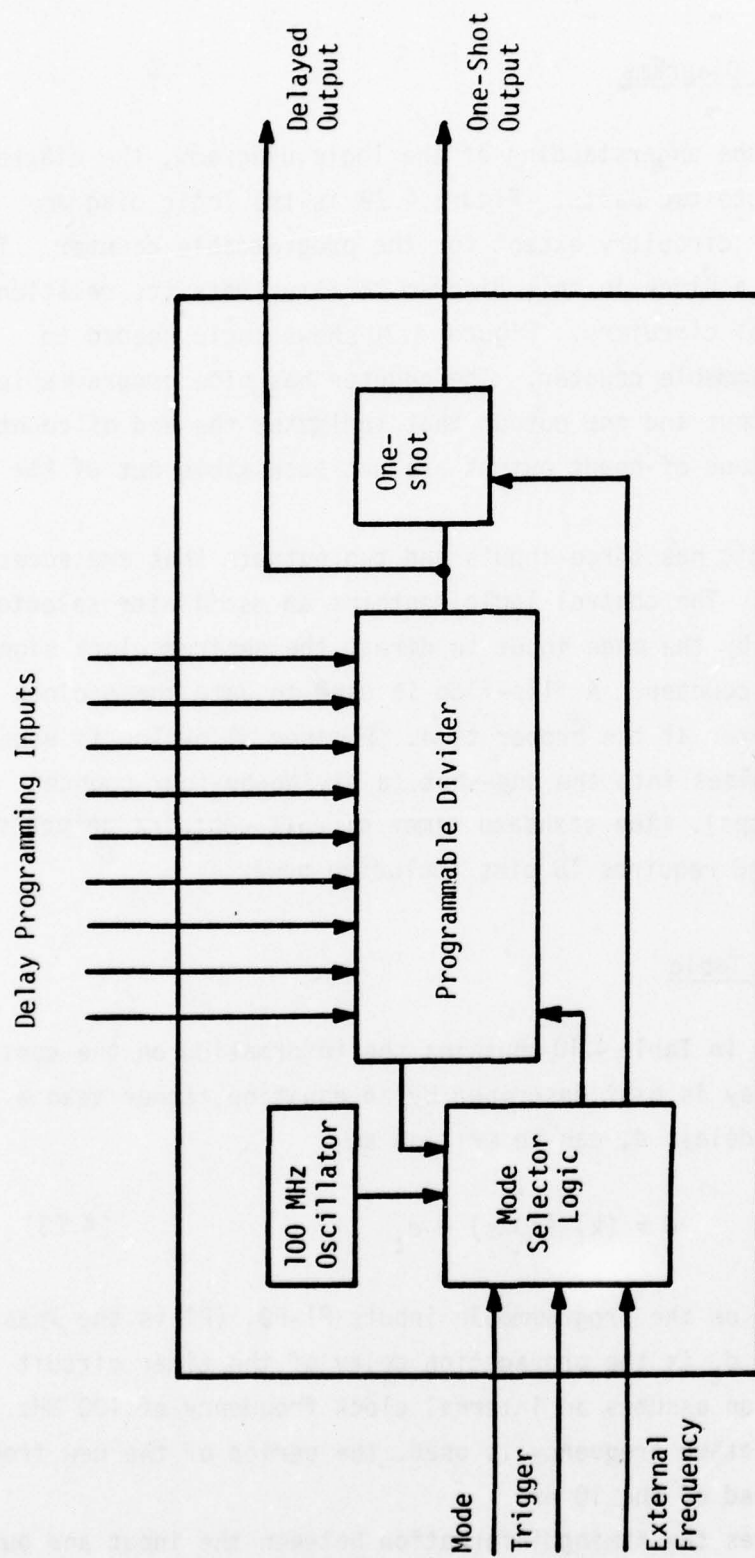


Figure 4.28 Functional Block Diagram of Standard Timer

4.2.4.2 Logic Diagrams

Again, to aid the understanding of the logic diagrams, the diagrams have been divided into two parts. Figure 4.29 is the logic diagram for all of the timer circuitry except for the programmable counter. The counter is shown as a block in this diagram to illustrate its relationship with the control circuitry. Figure 4.30 shows logic needed to implement the programmable counter. The counter has nine programmable inputs, one clock input and one output that indicates the end of count. The clock input and end-of-count output are not accessible out of the circuit.

The control logic has three inputs and two outputs that are accessible outside the circuit. The control logic contains an oscillator selector which is controlled by the mode input to direct the desired clock signal to the programmable counter. A flip-flop is used to gate these clock pulses into the counter at the proper time. Another flip-flop is used to gate the clock pulses into the one-shot (a divide-by-four counter made of two flip-flops). The standard timer circuit contains an equivalent of 114 gates and requires 16 pins including power.

4.2.4.3 Truth Table

The information in Table 4.10 contains the information on the control input. The time delay is best described by an equation rather than a lengthy table. The delay, d , can be written as

$$d = (k) (10\text{ns}) + d_t \quad (4.26)$$

where k is the count on the programmable inputs P1-P9, (P1 is the least significant bit) and d_t is the propagation delay of the timer circuit itself. This equation assumes an internal clock frequency of 100 MHz. If for any reason another frequency is used, the period of the new frequency should be used instead of the 10 ns.

Figure 4.31 gives the timing information between the input and out-

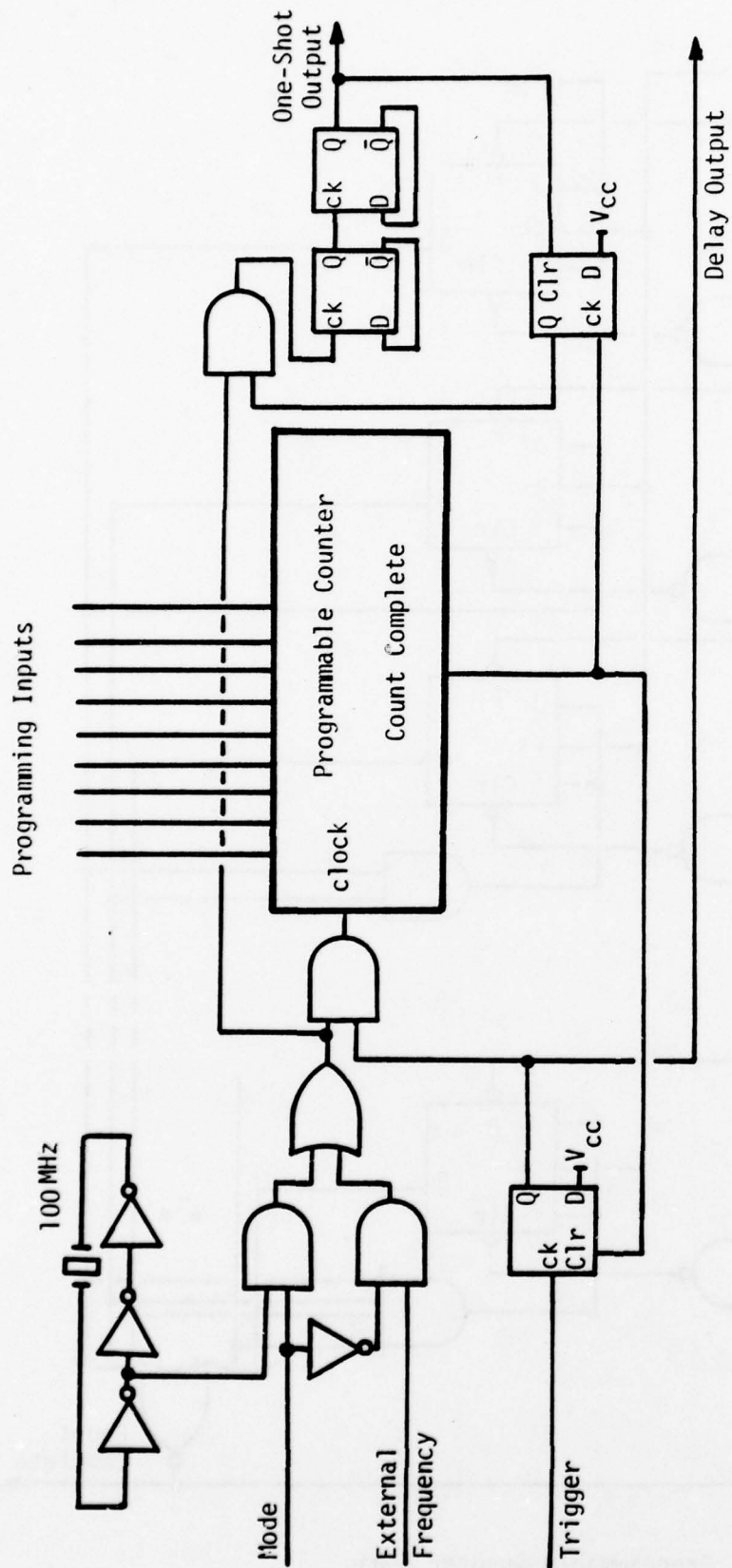


Figure 4.29 Standard Timer Control Logic

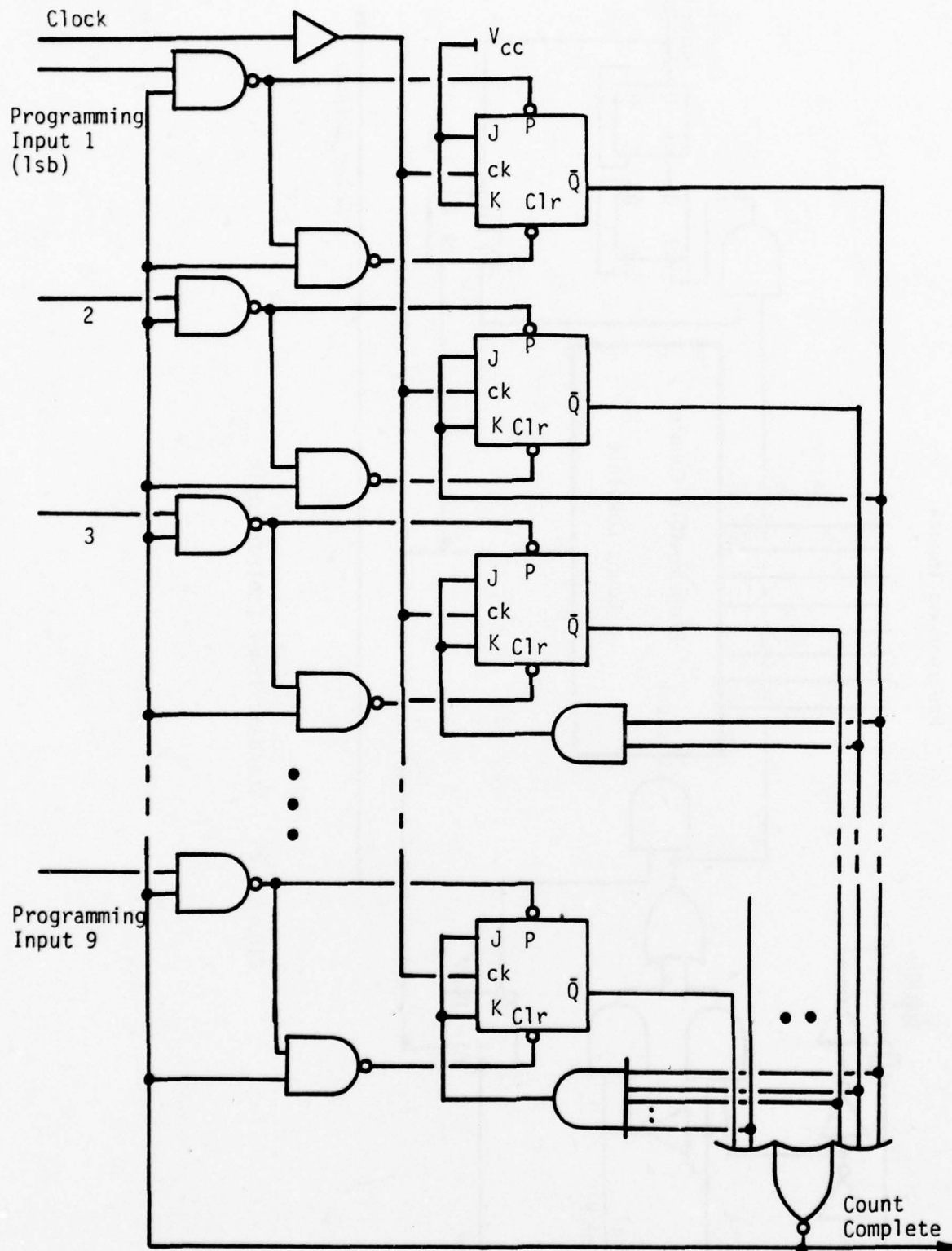


Figure 4.30 Programmable Counter Logic

Mode	Trigger	Frequency	Delay Output	One-Shot Output
1	↓	X	Enabled using internal ck	Enabled using internal clock Length is 40 nsec
0	↓	C	Enabled using external freq. input	Enabling using external freq. input, F. Length is $\frac{4}{F}$.

0 = Low
 1 = High
 ↓ = Transition from high to low
 X = Immaterial
 C = External clock frequency

Table 4.10 Truth Table for Standard Timer

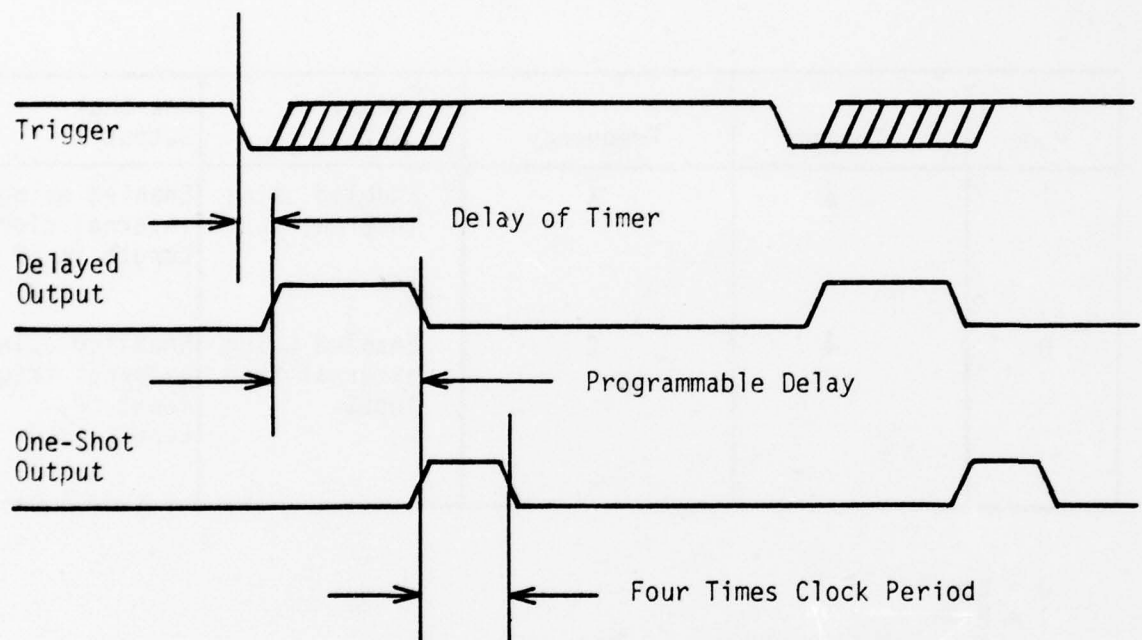


Figure 4.31 Input-Output Timing of Standard Timer

puts. The trigger input is edge triggered. In most applications, the other inputs will not be dynamic except the frequency input.

4.2.4.4 Circuit Implementation

As discussed for the SIIB and the residue generator and checker, implementation alternatives range from using off-the-shelf small and medium scale integrated circuits to custom designed chips. Table 4.11 lists the characteristics of each of these implementation approaches for the Standard Timer circuit. Again, the custom IC approach is most attractive because of the savings in space and power and in increased reliability.

4.2.4.5 Critical Parameters

The most critical timing questions arise in the first stages of the programmable counter and in the oscillator logic circuitry driving it. This is due to the high frequencies involved. The 100 MHz oscillator is necessary for the desired 10 ns resolution of the timer. There is presently commercially available an off-the-shelf 60 MHz oscillator that could be used as an alternative if a 100 MHz oscillator is impractical to put in a dual-in-line package. If the lower frequency oscillator is used, the design of the circuit becomes somewhat less critical because the limits of Schottky TTL are well above 60 MHz. Also, the resolution of the timer would be reduced and the longest possible delay would increase.

The counter must be designed so that the delay through the flip-flops and the necessary gates is small enough so that the flip-flops have time to preset or clear before the next clock pulse arrives. There are no other critical design parameters that must be treated separately.

The oscillator frequency stability is not extremely critical. There is no need to have a low drift oscillator because the timing delay is not critical.

Standard Timer	Conventional Implementation	Custom LSI Implementation
No. of Packages	7	1
No. of Pins	104	16
No. of Gates	167	114
Failure Rate	$0.608/10^6$ Hours	$0.32/10^6$ Hours
Power Typical	1300 mw	150 mw
Max.	2200 mw	275 mw

Table 4.11 Standard Timer Implementation Alternatives

4.2.5.6 QED Module Test Equipment Requirement

Since the standard timer has no output from the QED module in a typical application, there are no additional requirements placed on the test equipment. The proper operation of the timer is ensured when the entire BIT circuitry on the module is verified. When the timer is used for other than the BIT circuitry, the test equipment should be designed to verify the timing given in the Table 4.10 and Figure 4.31.

The following sections illustrate the application of the standard BIT circuitry recommended for the Process Class Modules to each member of that Class. In addition the analytic measures described earlier in this study are applied to the resulting modules with BIT.

4.2.5 Parallel Multiplier

The QED multiplier module has been fully described by Harrison [4] and in NELC Technical Document 434 so that a detailed description of its operation will not be repeated here. The recommended approach to BIT for the multiplier module is the use of the proposed standard interface parity circuitry (SIIB) to test the input latches and output buffers coupled with the use of the residue coding technique described in Section 4.2.2 and 4.2.3. Timing enables for the residue circuit can be provided by the standard timing circuit discussed in Section 4.2.4. The resulting QED multiplier along with the recommended BIT circuitry is shown in Figure 4.32.

4.2.6 Arithmetic Logic Unit

The basic BIT approach recommended for the ALU module is the same as for the multiplier module. Again, the standard parity and timing circuits are used along with the modulo 3 residue circuitry. However, since the ALU performs multiple arithmetic functions additional control is required for the residue checking circuitry. Fortunately, this circuitry is minimal since modulo 3 addition, subtraction and the exclusive OR operations are similar.

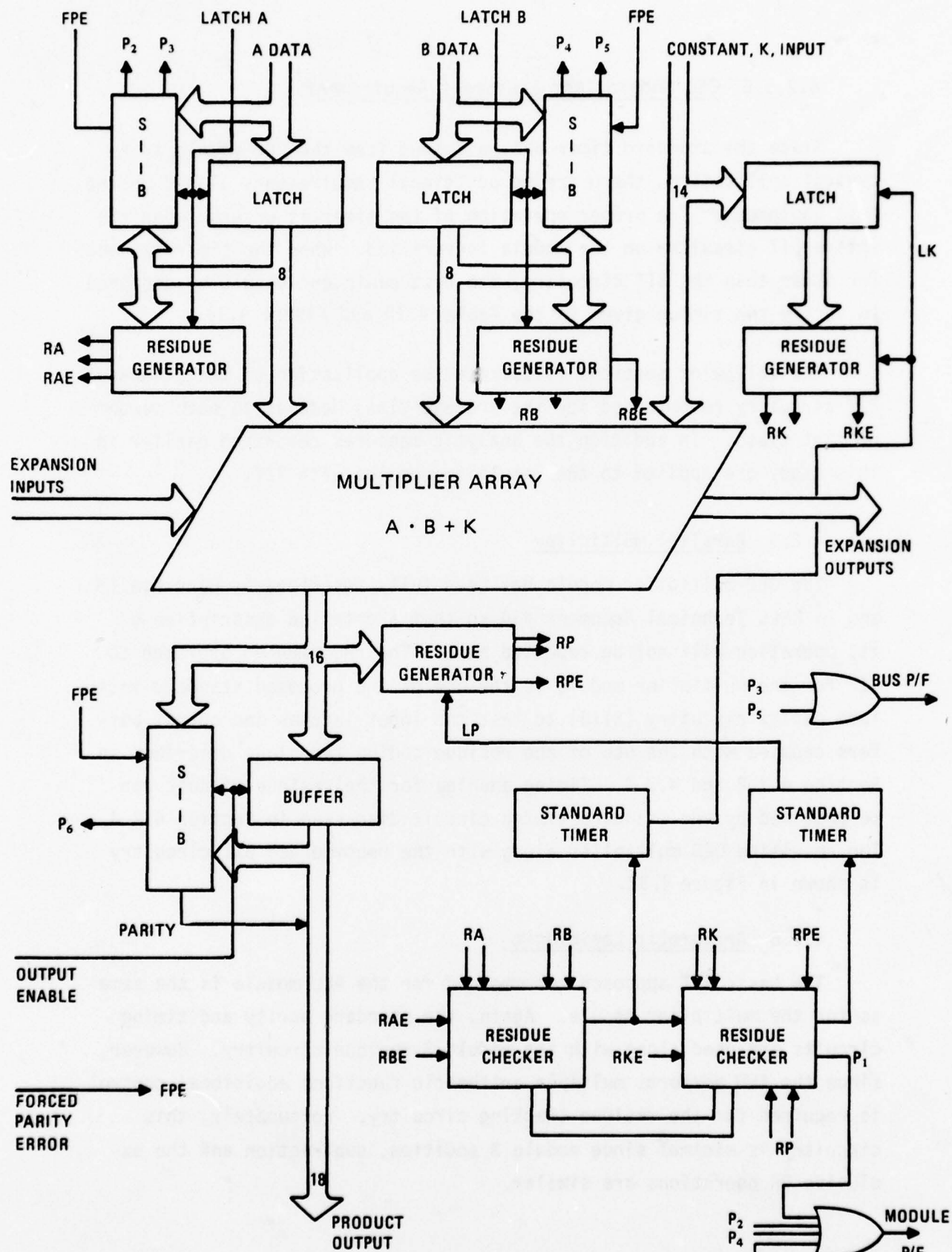


Figure 4.32 Built-In-Test for Parallel Multiplier

The arithmetic functions performed by the ALU module are:

B minus A

A minus B

A plus B

The logical operations implemented by the ALU are:

$A \oplus B^*$

$A + B$

$A \cdot B$

Arithmetic shift right and shift left operations are implemented through use of the accumulator register control.

A block diagram showing the ALU module plus the recommended BIT circuitry is shown in Figure 4.33. It should be pointed out that the error detecting properties of the modulo 3 residue code for the ALU is the same as for the multiplier.

4.2.6 8-Bit Index Counter

Analysis of the 8-bit index counter shows that it is very similar to the ALU module since it generates memory addresses using binary adders and subtractors starting from a user defined base. As a consequence, the basic BIT approach recommended for the 8-bit Index Counter is the same as that recommended for the parallel multiplier and ALU module which employ I/O parity and a residue code for concurrent error detection. There are, however, more input and output ports on the Index Counter so that several parity checkers and generators are required.

A block diagram of the 8-bit Index Counter along with its recommended BIT circuitry is shown in Figure 4.34. Because of the Index Counter's more limited arithmetic function set, less control is required. As in the case of the parallel multiplier and the ALU, standard parity circuits are

* $\oplus$ denotes the exclusive OR function.

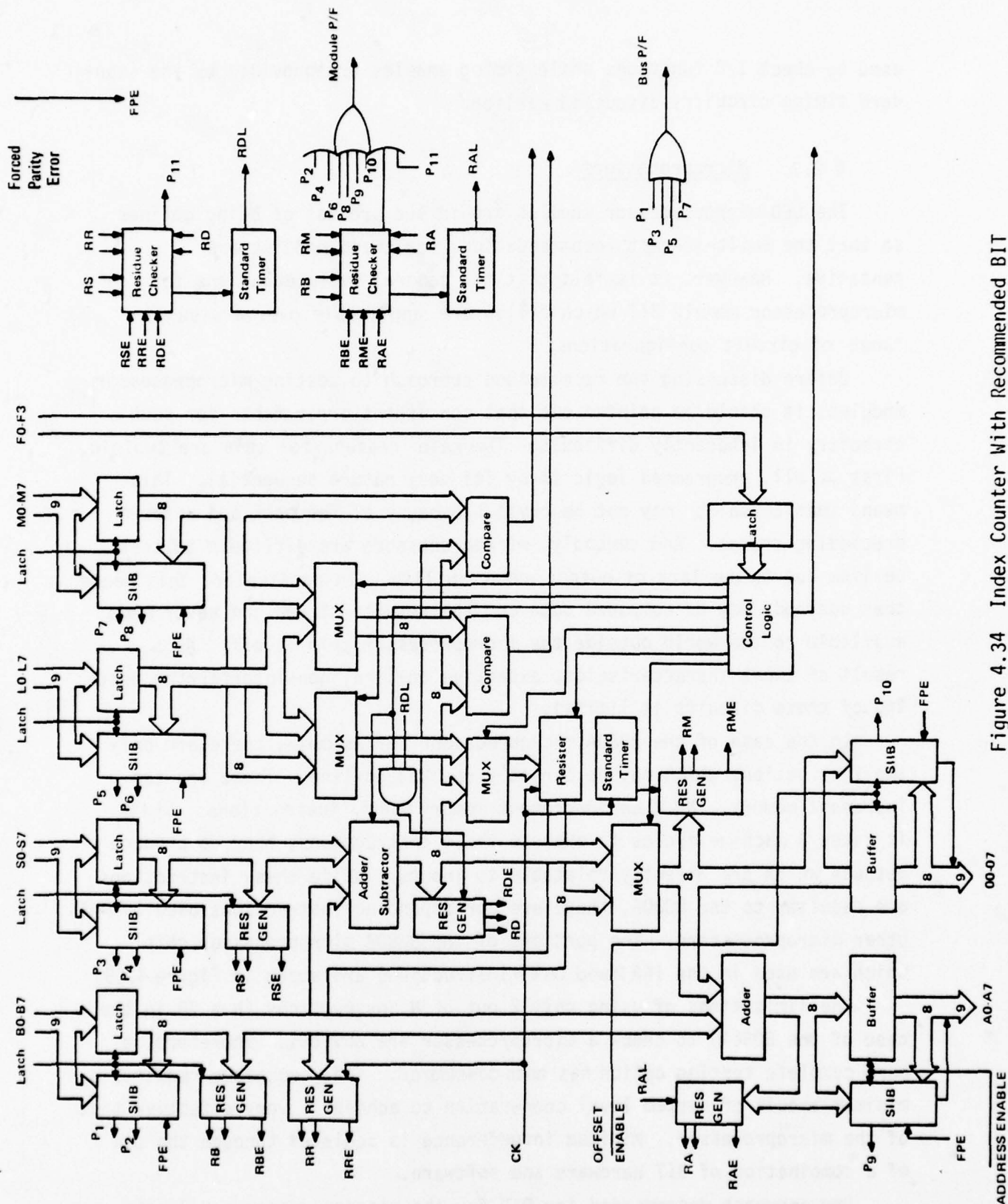


Figure 4.34 Index Counter With Recommended BIT

used to check I/O functions while timing enables are provided by the standard timing circuitry discussed earlier.

4.2.7 Microprocessors

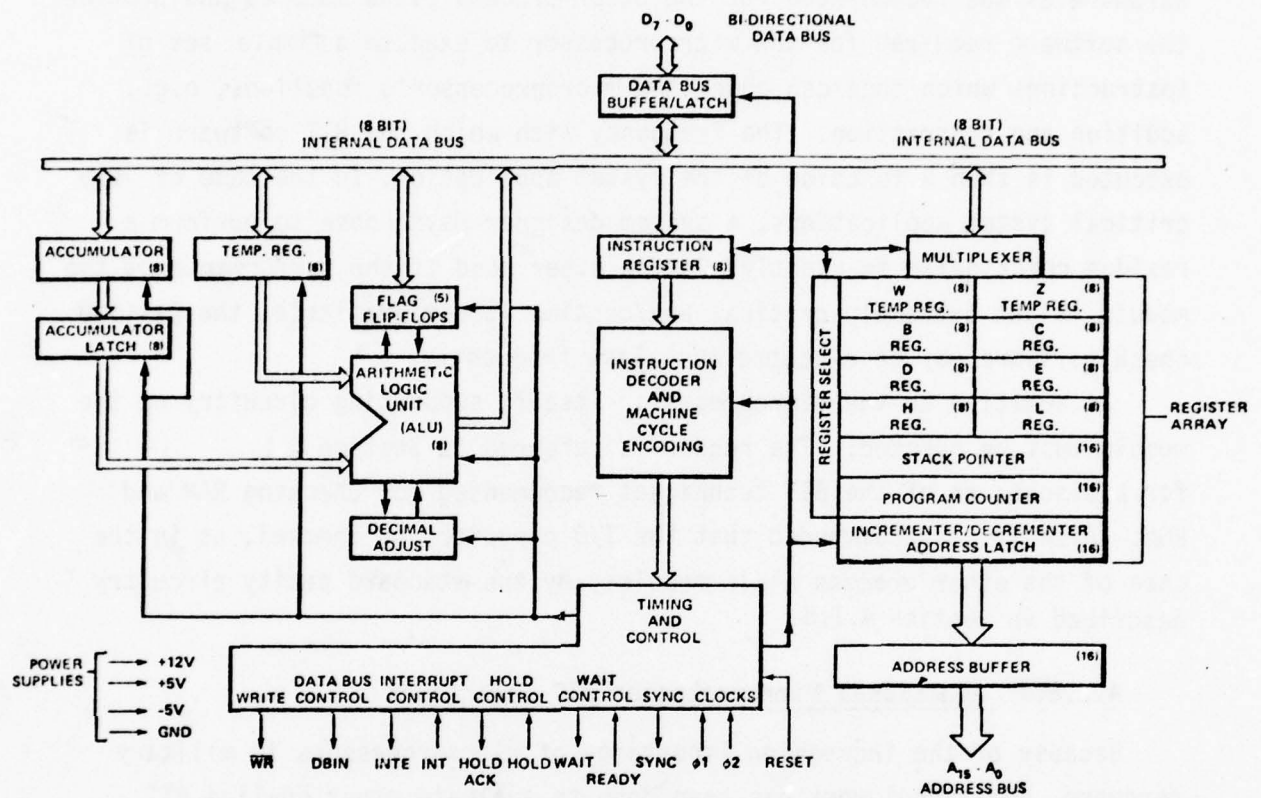
The QED microprocessor modules are in the process of being defined so that the built-in-test recommendations for these modules are tentative. However, it is felt that the general recommendations for microprocessor module BIT which follow are applicable over a wide range of circuit configurations.

Before discussing the recommended approach to testing microprocessor modules, it should be pointed out that concurrent error tests for such circuitry is inherently difficult. The main reasons for this are twofold. First of all, programmed logic is by its very nature sequential. This means that causality may not be obvious because of feedback and unknown preceeding states. And secondly, microprocessors are difficult to test on-line due to the lack of output observability. In particular, this means that operands may be computed upon and the results stored and never made available to the world outside the microprocessor chip itself. As a result of these characteristics, extensive on-line, non-interfering testing of these circuits is limited.

In the case of the 8080A microprocessor for example, there are only two instructions which may be directly checked on-line. These are the increment memory, INR M, and decrement memory DCR M, instructions. While it takes 3 machine cycles to execute these instructions, they do produce outputs which are directly relatable to inputs. While these instructions are peculiar to the 8080A, there are corresponding instructions used by most other microprocessors. The portions of the 8080A microprocessor chip which are used in the INR M and DCR M instructions are shown in Figure 4.35.

The limitations of using only 2 out of N instructions ($N = 78$ in the case of the 8080A) to check a microprocessor are obvious. Therefore, a more complete testing option has been considered. This technique requires a minimum amount of system level cooperation to achieve a very effective test of the microprocessor. Minimum interference is achieved through the use of a combination of BIT hardware and software.

One approach recommended for BIT for the microprocessor modules is



8080A MICROPROCESSOR

- UTILIZE INSTRUCTIONS FOR WHICH CAUSALITY IS OBSERVABLE
 - 8080A
 - INR M (Increment Memory)
 - DCR M (Decrement Memory)
- PROVIDE BIT HARDWARE WHICH MINIMIZES SOFTWARE REQUIREMENTS
 - RESIDUE CODE HARDWARE

Figure 4.35 Microprocessor Built-in-Test

to implement the same residue code generating and checking circuitry in hardware as was recommended for the other process class modules and provide the software required for the microprocessor to execute a simple set of instructions which then can check the microprocessor's functions, e.g., addition and subtraction. The frequency with which the BIT software is executed is then a function of the system application. In the case of very critical system applications, a system designer may choose to perform a residue check quite frequently. On the other hand if the performance of the module is not extremely critical and/or time is not available, the residue check software may be executed much less frequently.

In addition to the microprocessor itself, supporting circuitry on the module must be checked. The reader is referred to Section 4.1 for a discussion of the BIT techniques recommended for checking RAM and ROM. Also it is recommended that the I/O circuitry be checked, as in the case of the other process class modules, by the standard parity circuitry described in Section 4.1.5.

4.2.8.1 Additional Microprocessor BIT Techniques

Because of the increasing importance of microprocessors in military hardware, additional work has been done to evaluate other on-line BIT techniques for this member of the Process Class family. These techniques along with the application of the BIT effectiveness measures are presented in the following discussion.

The microprocessor-based QED module, as presently envisioned, is a multiscard module which is expandable in increments of memory and I/O. The complexity of this function results in a modular module. The approach taken in this discussion will be to define the CPU card as the module for which BIT techniques will be recommended.

This position is reasonable since the memory submodules (both RAM and ROM) are similar to the existing QED memory modules. The BIT approaches described for the memory modules are applicable to the microcomputer's memory submodules. The other submodules are involved with I/O expansion and/or specific interfaces for a number of devices.

The I/O submodules are not, at the present, fully defined or documented. The last of seven proposed I/O modules raises anew the question of philosophy of a standard module set. If the concept of a limited but generally applicable set of modules (such as QED) is valid, then the design and implementation of device to microprocessor interfaces which do not utilize these existing modules seems questionable. In short, the definition of the I/O functions and their implementation is not definite enough to be able to include them in this report.

The CPU module (without BIT) is shown in Figure 4.36. This design is not taken from the QED module microprocessor drawings since they were not available. The design is typical, has been fabricated, operates properly and the statistics used should be very similar to any CPU based on an 8080A processor.

The microprocessor's programmability facilitates the possibility of two approaches to BIT. The first to be considered involves software self tests and the second, the addition of hardware to provide monitoring as in other QED modules.

The cost of software BIT approaches can be tallied in a slightly different way from conventional added hardware. The cost of added software for BIT proposes results in costs from

- Increased memory requirements,
- Increased demand for CPU cycles, and
- The development cost.

The development cost has not been included in other comparisons and will be dropped here. The cost of increased memory (if an incremental jump in hardware is required for the diagnostic space) can readily be determined for a specific application. The application software size is known but it is difficult to access in general.

Similarly, the availability of CPU cycles for testing may be evaluated easily in specific cases when the application is known and not at all when it is not known. There may be applications which are so demanding of the CPU that inadequate self testing can occur; thus, self test as a general strategy is difficult to recommend.

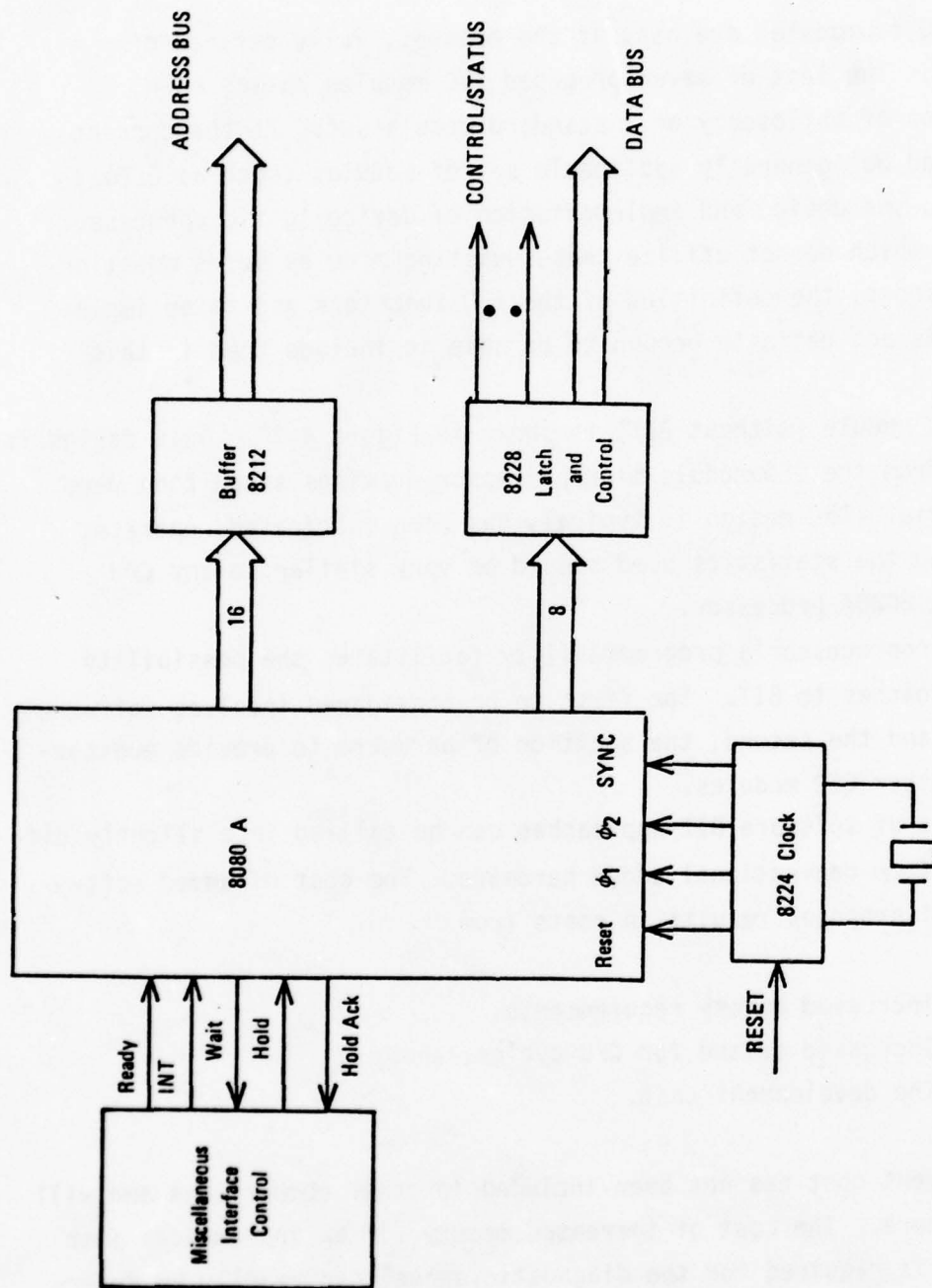


Figure 4.36 8080A CPU Card without BIT

In order to have some point from which a decision can be made, estimates of a CPU diagnostic are made in Table 4.12. It is important to note that this estimate was made using the processor exerciser routine. The routine was expanded to allow for self-test (rather than post exercising), the memory requirement was increased to provide for some data comparison storage, and the instruction size/execution time were estimated using a typical mix.

Memory Requirement * (Test & Text Storage)	600 bytes
CPU Cycle Requirement *	8000
Estimates made: Typical instruction 1.5 bytes Typical execution 7 cycles Overall iteration *2 total code.	

Table 4.12 Self Diagnosis Parameter

The estimates in Table 4.12 are for the CPU self-test only and do not include memory or I/O diagnostics. The software tests have no ability to directly control and observe the support hardware on the CPU module and thus it is only some portion of the processor which is tested. The effectiveness of the test in detecting processor or faults can not be defined accurately. This is due in part to a lack of statistics and in part to an incomplete understanding of the failure mechanisms in current microprocessor technology.

Another approach which employs a mix of hardware and software to implement is a "watch dog" time. An external timer is used to establish some countdown period. The application's program is required to reset the time at a rate faster than this period or the failure signal will result. In this way, any failure which renders the processor inoperative will cause the fail signal to be sent. The applications program may make the reset of the alarm conditional upon the passing of some diagnostic software. In this way, as much available time as desired may be used to perform self testing. The hardware block diagram for this approach is shown in Figure 4.37.

The control hardware is on the CPU module which is used in support of the microprocessor and cannot be effectively monitored using any of the standard BIT circuits proposed. A SIIB function can be used to check the two 8212 buffers and portions of the 8228. This also provides the capability to generate parity for use on the intermodule connection bus.

No further evaluation of software approaches will be given for the reasons noted in the previous section. Specific recommendations for the BIT hardware are given in Section 4.1.5 and the results are summarized in Table 4.13.

4.2.9 Application of the Analytic Measures to the Recommended BIT Approaches

As for the memory class module, to quantify the gains and costs of the BIT techniques, it is necessary to apply the analytic measures described

in Section 3.0. These measures provide a good indication of the additional costs involved, and they also furnish a guide to the effectiveness of the BIT techniques.

A summary of the BIT evaluation for the process class modules is shown in Table 4.13. As one can observe, the recommended BIT furnishes an effective error detection capability for the parallel multiplier module and the typical microprocessor module. The BIT effectiveness for the other two modules is only slightly less. The costs involved with the microprocessor module are quite reasonable which results in an effective and inexpensive BIT. The costs for the other modules are somewhat higher than for the other modules. This results from the functional complexities of the module that must be monitored by BIT. This BIT technique using residue coding does provide an effective, concurrent approach to BIT at a reasonable cost.

Included within Appendix B is a detailed package description of the particular modules with the recommended BIT. The data necessary to compute the analytic measures are also shown on each of the module's sheets.

4.2.10 Process Class BIT by Partial Duplication

By duplicating the functional elements of the module and comparing outputs, it is possible to provide a higher level of BIT effectiveness than is provided by residue coding. However, this increase in fault detection capability can be achieved only at a cost higher than with residue coding. Figure 4.38 shows at a block diagram level the Arithmetic Logic Module with the functional elements duplicated and the other circuitry needed to implement this BIT technique. Figure 4.39 gives a similar description of the Eight-Bit Index Counter Module with partial duplication BIT.

Parameter	Parallel Multiplier	Arithmetic Logic Unit	Eight-Bit Index Counter	Typical Microprocessor	Units
Percent of Gates Monitored	72	57	52	86	%
Percent of Packages Monitored	76	44	54	53	%
Number of Cycles for Test	0	0	0	10 - 8000	-
Ratio of BIT Packages to Total Module Packages	52	39	38	33	%
Failure Rate Without BIT	1.4	1.4	2.3	2.0	/10 ⁶ Hr
Failure Rate With BIT	3.0	2.5	4.4	2.5	/10 ⁶ Hr
Ratio of BIT Failure Rate to Total Module Failure Rate	55	44	48	22	%
Power Consumption of BIT	1.3	1.1	1.5	0.4	watts
Ratio of BIT Power Consumption to Total Modules Power Consumption	19	19	27	11	%

Table 4.13 Process Class BIT Evaluation

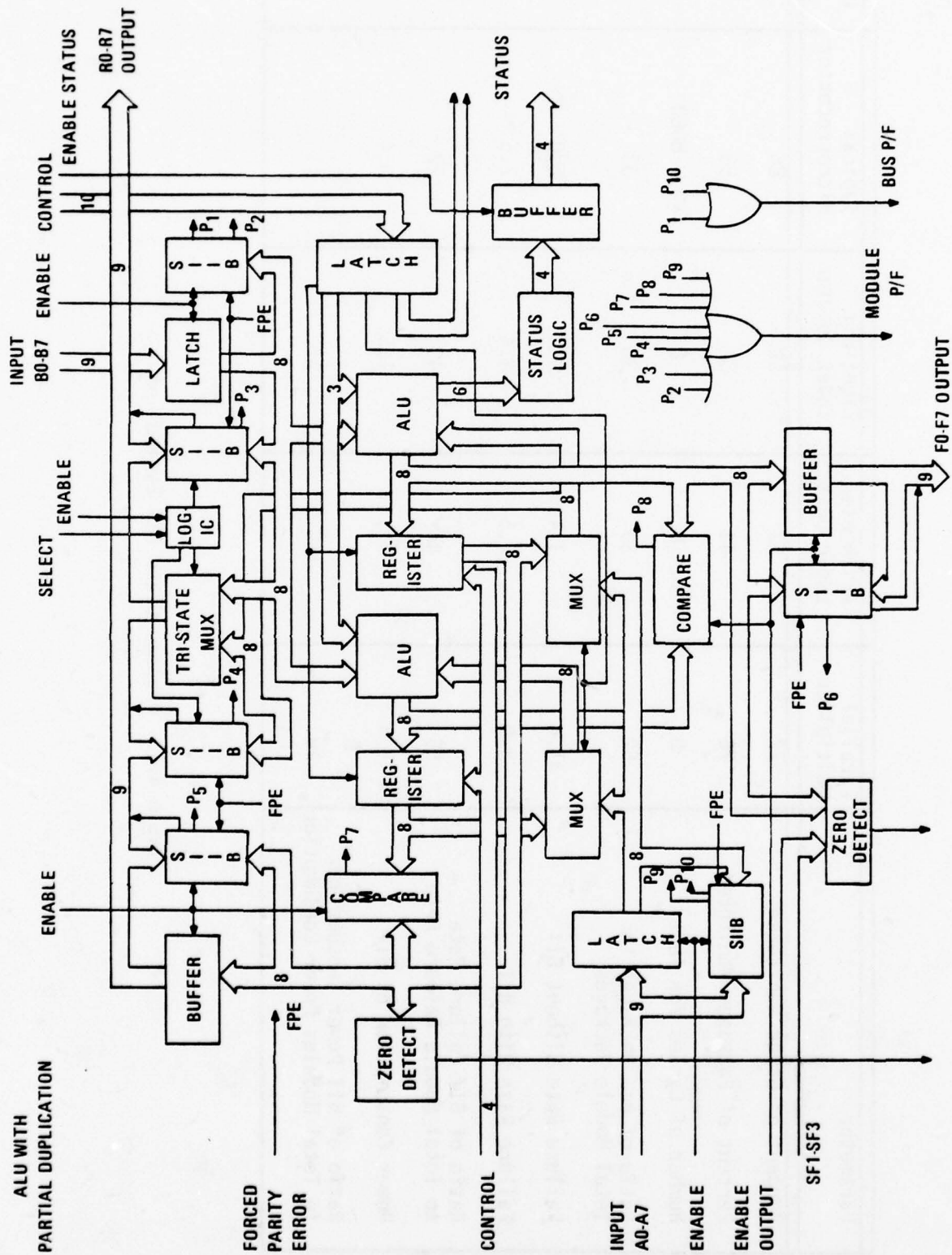


Figure 4.38 Arithmetic Logic Unit Partial Duplication BIT

Figure 4.39 Index Counter With Partial Duplication

Table 4.14 summarizes the BIT analytic measures for the ALU and the Eight-Bit Index Counter modules with partial duplication as the BIT. For easy comparison the analytic measures for these two modules with residue coding BIT is tabulated alongside. As mentioned earlier, the BIT by partial duplication does provide a higher level of fault detection. However, the costs, in terms of number of packages and to an even greater extent the power consumption, are also increased.

The desirability of this BIT approach must be carefully evaluated in terms of added costs in relation to added benefits. Also the partial duplication approach does not conform to the standard approaches to BIT for the QED modules.

Appendix B also gives a detailed package description of these two modules with BIT by partial duplication. The necessary data to compute the analytic measures are also shown.

Parameter	Arithmetic Logic Unit		Eight-Bit Index Counter		Units
	Residue	Partial Duplication	Residue	Partial Duplication	
Percent of Packages Monitored	44	61	54	82	%
Percent of Gates Monitored	57	90	52	88	%
Number of Cycles for Test	0	0	0	0	-
Ratio of BIT Packages to Total Module Packages	39	44	38	48	%
Failure Rate Without BIT	1.4	1.4	2.3	2.3	/10 ⁶ Hr
Failure Rate With BIT	2.5	2.5	4.4	4.4	/10 ⁶ Hr
Ratio of BIT Failure Rate to Total Module Failure Rate	44	44	48	47	%
Power Consumption of BIT	1.1	2.8	1.5	2.4	Watts
Ratio of BIT Power Consumption to Total Module Power Consumption	19	40	27	40	%

Table 4.14 Comparison of Residue Arithmetic and Partial Duplication

4.3 Built-In-Test of Control Class Modules

This section discusses the QED modules which perform the broad range of digital control functions required in many digital systems. The need which the control class modules fill in a system composed of QED modules contrasts sharply with that of the memory and process modules. Whereas the process and memory modules operate directly on data, the control modules generate signals which cause data, in some sense, to become dynamic or static. The result is that test methods which are generally applicable to process modules are not, for the most part, directly applicable to testing control modules. However, there are built-in-test approaches which are applicable to each control module. The following discussion considers each of the control class modules separately with regard to the types of built-in-tests which are applicable.

4.3.1 Programmable Timing Generator

The programmable timing generator provides synchronous timing signals whose characteristics are determined by data stored in programmable read-only-memories (PROM's) so that complex timing signals can be generated using straightforward sequential addressing. The module contains an on-board clock with the option of using an external fundamental timing source. Division of the fundamental clock frequency is programmable as well as the PROM address counter. Thus, the timing generator has flexibility at 3 different levels as follows:

- (1) Clock Frequency,
- (2) Address Sequence,
- (3) Programmable ROM Data.

While this flexibility adds considerably to the universality of the module, it also contributes to the difficulty of providing a built-in-test for the module. In fact, the difficulty of checking digital timing circuits is well known (c.f., McCluskey, et.al., [20] and Wakerly [21]).

In addition to the testing problems attributal to the module's basic design flexibility, the large number of indirectly related inputs and outputs

found on the programmable timing generator leads to further testing difficulties. This diversity is explainable by the nature of timing signal requirements in general since they must service quite varied functional modules in most digital system applications. However, herein also lies a clue to a suitable built-in-test approach for the programmable timing generator.

The recommended BIT approach for concurrent fault detection for the programmable timing generator is replication coupled with the parity approach recommended for checking programmable read-only-memories. The standard approach for checking input-output parity is not applicable to the timing signals generated by the programmable timing generator because the multiple destinations don't lend themselves to checking bus parity. Instead, it is recommended that parity be stored in PROM's and checked after the output buffers on the module.

This BIT using parity, which is a concurrent error detection technique, by itself provides sufficient BIT capability for many applications. However, provisions are also made for the comparison of the timing signals by replication of a portion of the module. Thus at the option of the system designer, for more critical applications, a higher level of fault detection can be easily added. Figure 4.40 illustrates the programmable timing generator module with the recommended built-in-test.

4.3.2 Priority Encoder

Unfortunately the priority encoder does not lend itself to monitoring using any of the standard BIT techniques previously described. The output buffer is the only functional block of the module that can be checked with a standard BIT circuit, namely the SIIB. This provides a low error detection capability, but there is a method available that will furnish a high error detection capability. Partial duplication of the module in conjunction with the SIIB does provide a high error detection capability. In fact this concurrent BIT technique monitors over 95% of the module's gates. The costs of this method are

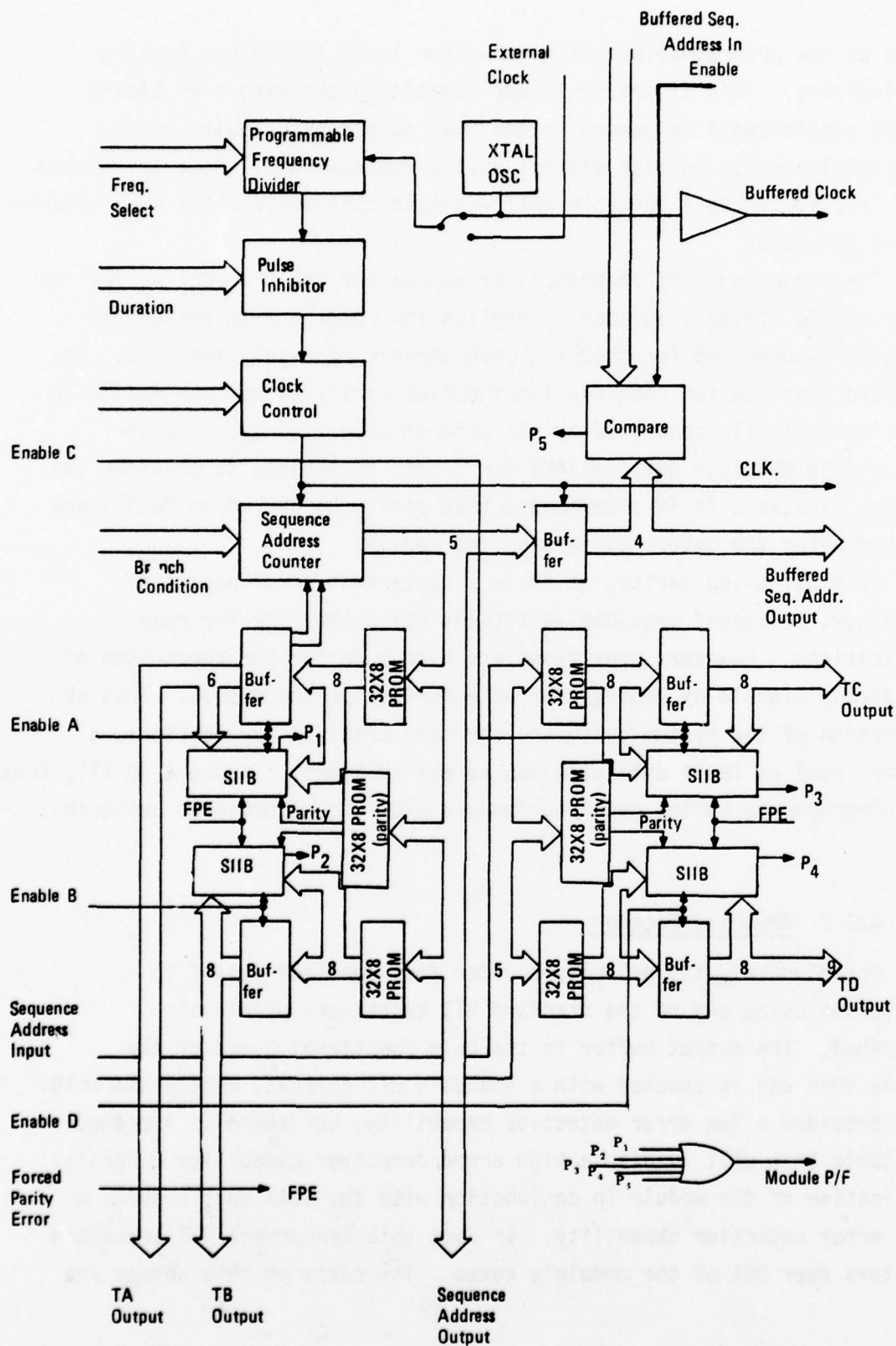


Figure 4.40 Programmable Timing Generator with BIT

higher than most of the other BIT approaches, but this module without BIT has one of the lowest failure rates. Therefore with the BIT, which accounts for less than half of the module, the module's failure rate is not excessive. Figure 4.41 depicts the resulting Priority Encoder module with the recommended BIT.

4.3.3 Application of the Analytic Measures to the Recommended BIT Approaches

To more accurately quantify the gains and costs of the BIT techniques, it is necessary to apply the analytic measures described in Section 3.0. These measures provide a good indication of the additional costs involved, and they also provide a guide to the effectiveness of the BIT techniques.

A summary of the BIT evaluation for the control class modules is shown by Table 4.15. One can see that BIT provides an effective error detection capability for each of the modules in the class. The BIT by partial duplication for the priority encoder has a higher apparent cost than most of the modules. This is because the module without BIT is relatively uncomplicated, which makes a BIT approach with average complexity appear costly relative to it. Also the BIT technique for the priority encoder does provide fault detection capability to a functional module that does not lend itself to clever, inexpensive BIT methods.

Appendix B gives a detailed package description of the particular modules with the recommended BIT. The data necessary to compute the analytic measures are also shown.

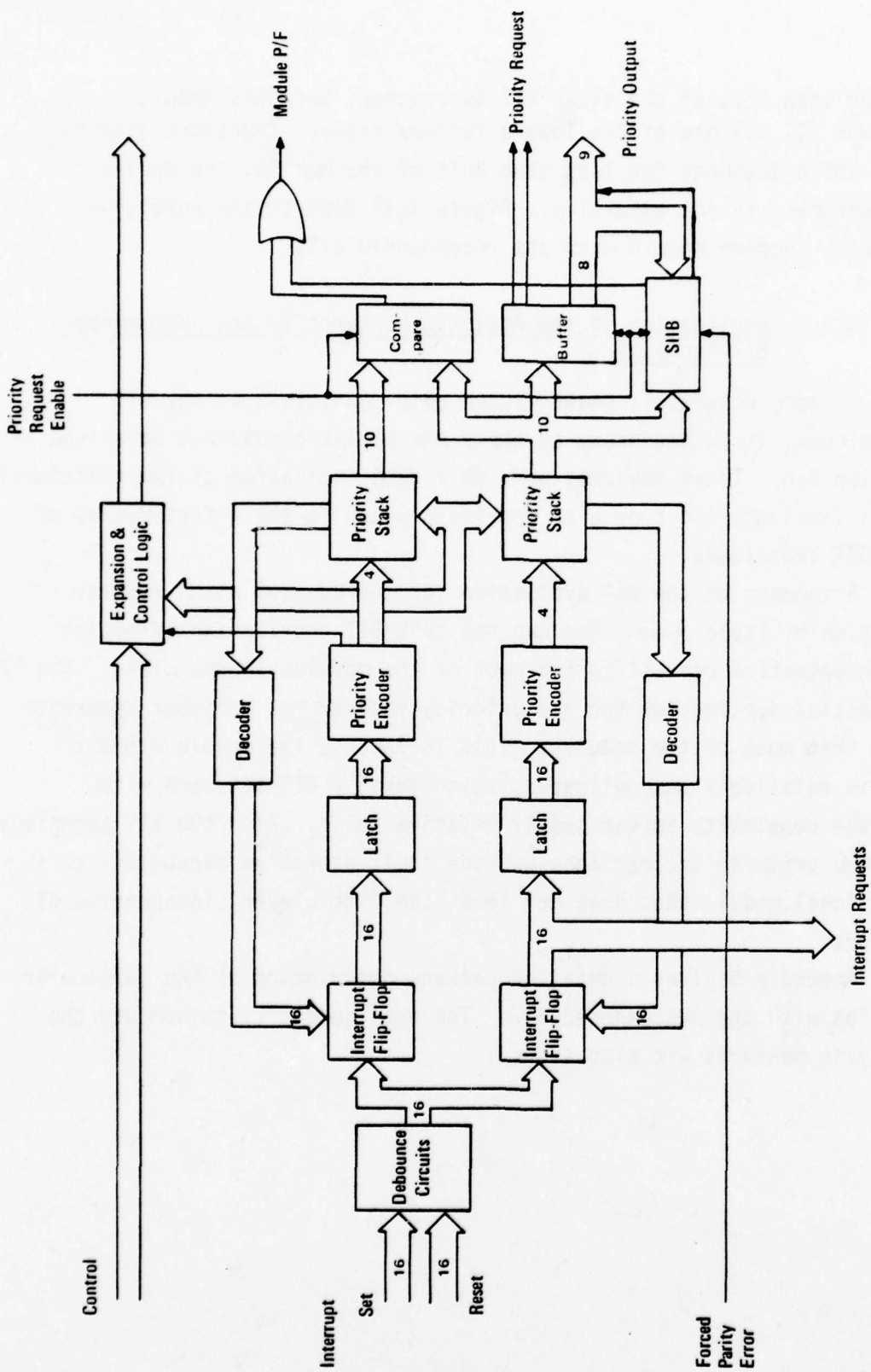


Figure 4.41 Priority Encoder With Recommended BIT

Parameter	Programmable Timing Generator	Priority Encoder	Units
Percent of Gates Monitored	86	96	%
Percent of Packages Monitored	50	89	%
Number of Cycles for Test	0	0	-
Ratio of BIT Packages to Total Module Packages	25	48	%
Failure Rate without BIT	2.9	1.7	/10 ⁶ Hr
Failure Rate with BIT	3.5	2.9	/10 ⁶ Hr
Ratio of BIT FR to Total Module FR	15	41	%
Power Consumption of BIT	1.1	2.5	Watts
Ratio of BIT Power Consumption to Total Module Power Consumption	18	44	%

Table 4.15 Analytic Measures Tabulation for Control Class

4.4 Built-In-Test for Interface Class Modules

The interface class contains five modules. Each will be discussed individually. The modules are:

- (1) Dual 8-bit Switch,
- (2) Dual Parallel 8-bit Interface,
- (3) Asynchronous Serial Interface,
- (4) NTDS Input Buffer,
- (5) NTDS Output Buffer.

The interface class of modules are characterized by their ability to transfer data from and to other members of the QED module family and also between other electronic data systems. The modules in this class accept data in eight-bit wide word (or multiples of eight bits).

In general this class of module performs very little processing. Therefore, a functional check of the processing portion of module checks a small percentage of it. However, a check of the input/output operations verifies a large portion of total circuitry of the module. To provide built-in-test for the interface class modules, the concept of checking the inputs latch, output buffer, and the bus of each module with the SIIB circuit provides a significant portion of the built-in-test needed.

4.4.1 Dual 8-Bit Switch

Since the switch module is basically a selective input/output buffer, the application of the SIIB circuit provides a high level of fault detection capability. This straightforward BIT approach is possible since all of the data buses that the module will switch, are nine bits wide. (A parity bit has been added to the data bus by the SIIB of the switched module.) Because each SIIB will check and regenerate the parity bit for each bus, it is not necessary to add a multiplexor circuit to the module to switch the parity bit. Figure 4.42 shows the 8-Bit Switch module with BIT as described.

As with all modules using SIIB circuits, it is possible to distinguish between bus failures and module failures which greatly facilitates fault localization. Another benefit of this BIT approach is the fact

that it will correct a bus parity error. Thus the error is not propagated through the system which further aids the fault localization problem.

While the BIT for this module accounts for a significant portion of the total hardware and failure rate costs, it must be noted that this module without BIT has few components. Thus, if one is limited to a certain percentage of the module for BIT, one is severely limited in this module by the amount of hardware that can be added. Even though 40 percent of the module's failure rate is from BIT, the module with BIT has the lowest failure of any of the modules in this report. Thus, the recommended BIT approach provides a concurrent monitoring technique with a high degree of fault detection capability at a moderate cost.

4.4.2 Dual 8-Bit Parallel Interface

The data path of this module can easily be monitored with the SIIB circuits. The latches and buffers as well as the small amount of processing can be checked with parity. This makes it possible to detect all single bit errors in any of these module elements. The SIIB also provides a check on the interconnections. No attempt is made to monitor the control function. A functional block diagram of this module with the recommended BIT is shown in Figure 4.43.

4.4.3 Asynchronous Serial Interface

The main functional unit in this module is the Universal Asynchronous-Receiver-Transmitter (UART). Its internal circuitry checks parity, framing and overrun errors. The non-TTL interfaces can be checked if the transmitting and/or receiving devices generate and check the parity signals. Using the SIIB circuit to implement this parity generation and checking it is possible to check the latches and buffers to the QED family. The pass/fail line from the UART can be Or-ed with parity bus checkers to provide a composite signal. The BIT is then used to detect errors in an odd number of bits in the latches as well as parity, framing, and overrun errors in the UART. A functional block diagram of the Asynchronous/Serial Interface module with BIT is shown in Figure 4.44.

The self-checking UART performs most of the BIT with no additional

Only One of Two Identical Halves of the Module is Shown

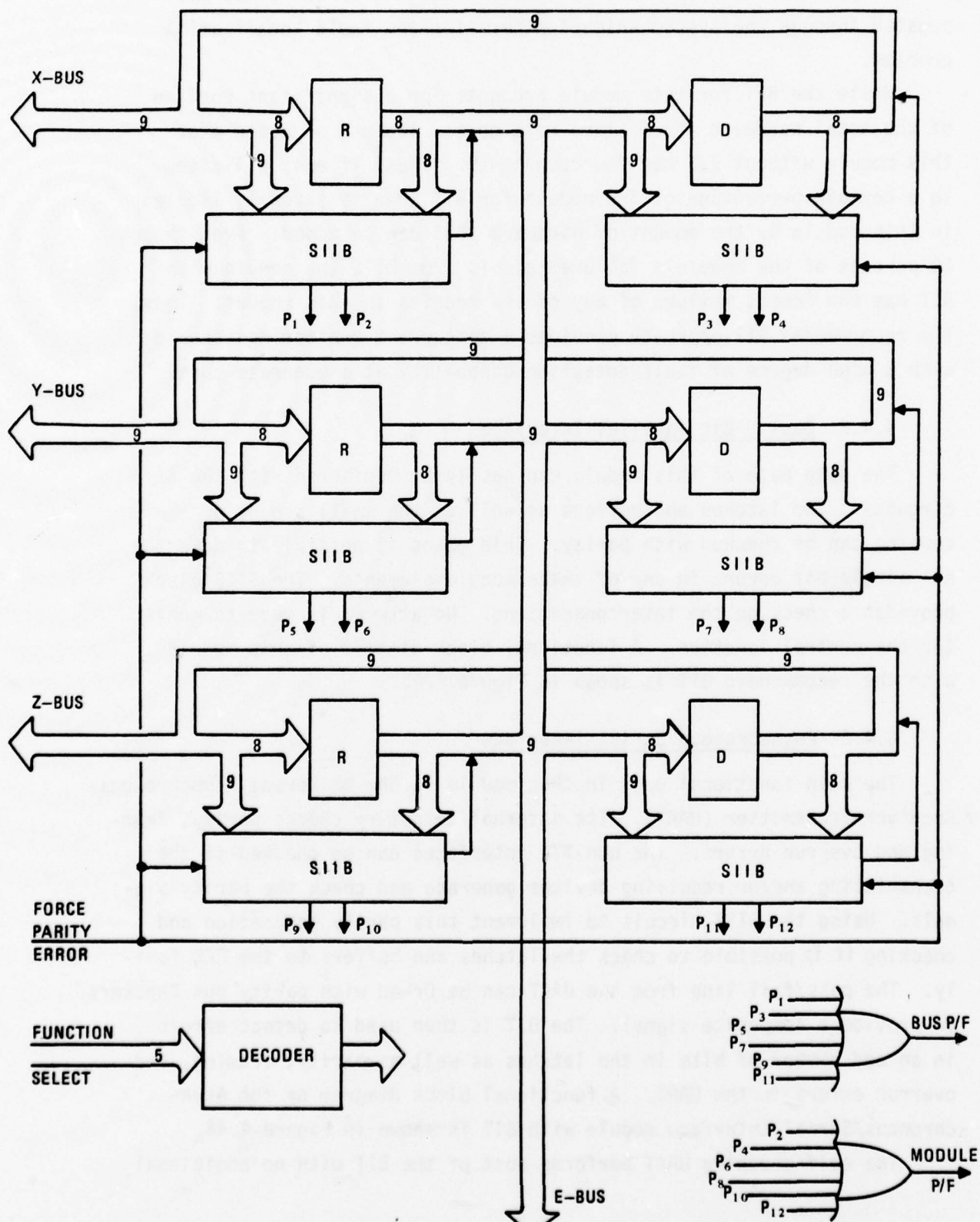


Figure 4.42 Dual 8-Bit Switch With Recommended BIT

One Only of Identical Module Halves is Shown

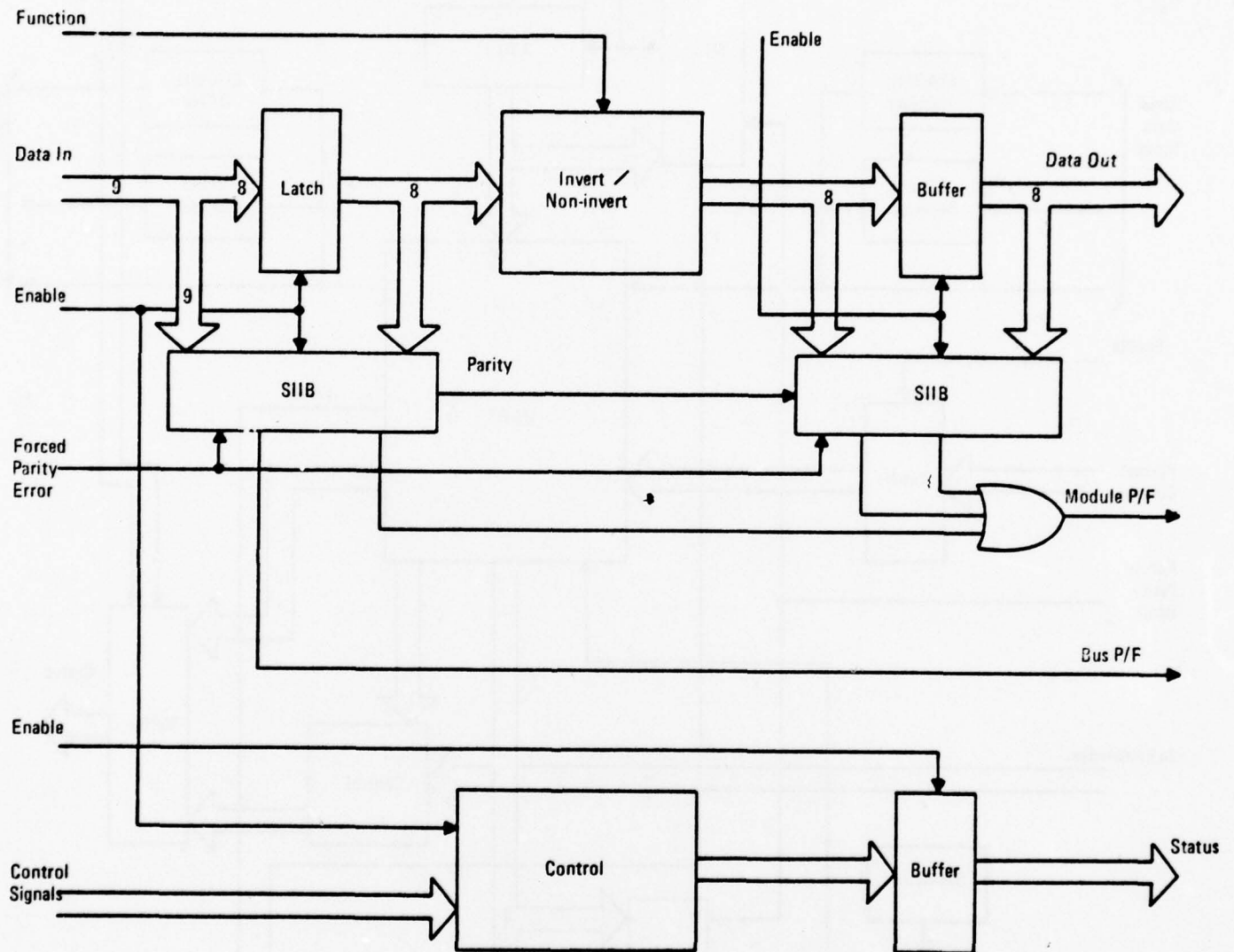


Figure 4.43 Dual Parallel 8-Bit Interface With BIT

packages. The parity bus concept provides a check of the input and output buffers. The combination of these gives a fairly complete check of the data path with only a small increase in hardware. A complete module test can be done off-line at the system level if and when necessary.

4.4.4 NTDS Input Buffer

The NTDS (Navy Tactical Data System) input buffer provides a way to convert data from an NTDS computer or peripheral to TTL levels for use by the QED family. The bus parity concept provides a check of the output buffers on the TTL interface.

In the current NTDS Input Buffer design a built-in-test circuit is employed to check some of the control circuitry. The method uses a pair of one-shots (monostable multivibrators) of the same duration, triggered from two parts of the circuit to be checked. The delay of the circuit is taken into consideration so as to make the rising and falling edges of the two one-shot pulses coincide. These outputs are fed into a parity checker which detects when the two lines are different. Since it is impossible to make the pulses coincide exactly, a filter is added to the output of the parity checker. This filter allows an error pulse (full width one-shot pulse) to pass, but does not allow the narrow pulses that occur from one shot pulse coincidence inaccuracies to pass.

Unfortunately this circuit, while providing a check on part of the control circuitry, increases the failure rate considerably. The two BIT circuits each utilize a dual one-shot which require external capacitors and resistors. These discrete components have a much higher failure rate than the integrated circuits they are checking. Table 4.16 illustrates this fact.

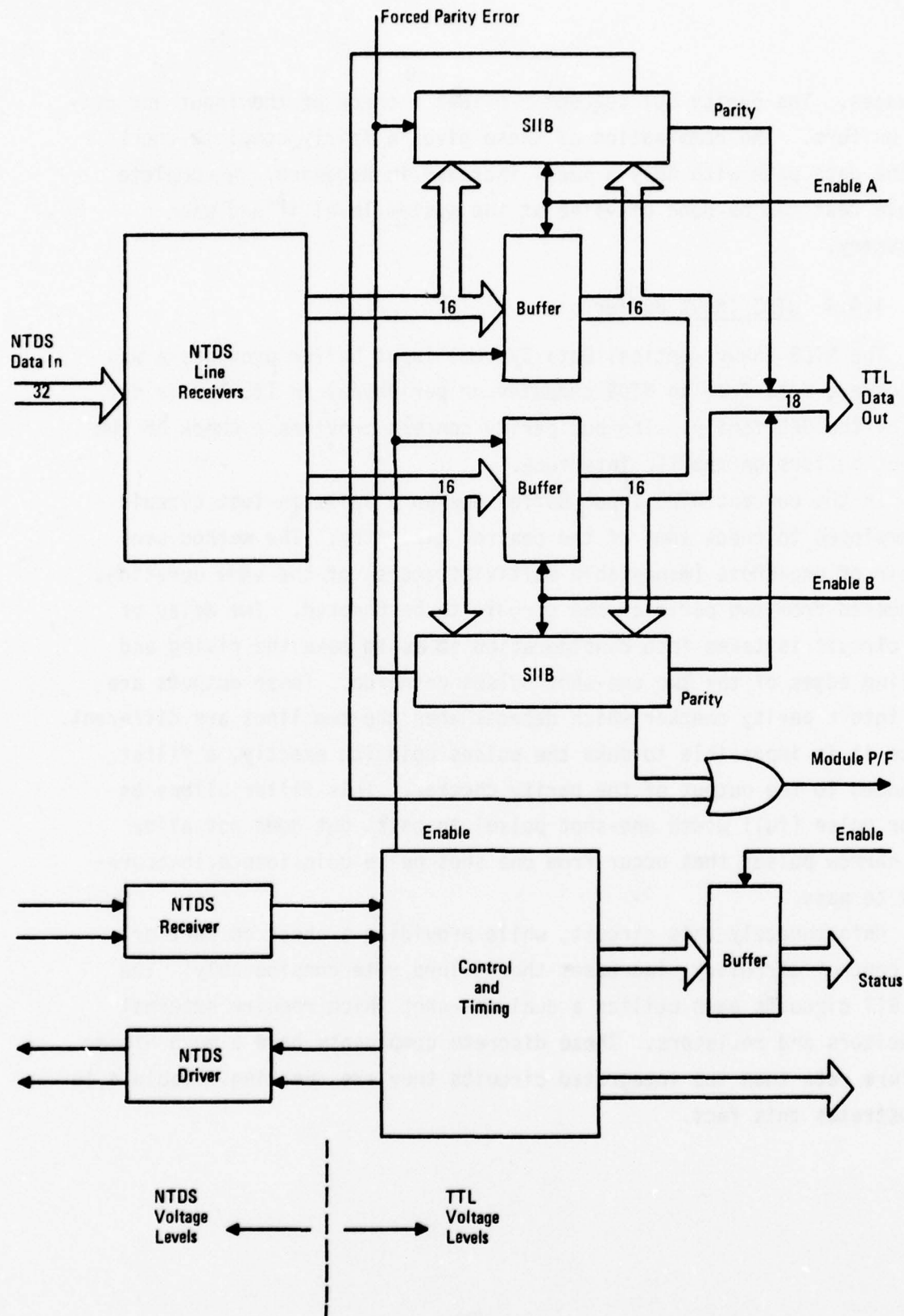


Figure 4.45 NTDS Input Buffer With BIT

<u>Parts in BIT</u>	<u>Failure Rate/Device</u>	<u>Failure Rate</u>
2 54123 dual one-shot	.03915	.0783
5 resistors	.08	.40
5 capacitors	.2	1.0
1 93S62 parity checker	.06	<u>.06</u>
	Total	1.538
<u>Parts that BIT checks</u>		
2 5474 Dual D F/F	.04365	.0873
1 54121 one shot	.03365	.03365
1 resistor	.08	.08
1 capacitor	.2	.2
1 5400 Nand	.03915	<u>.03915</u>
	Total	.440

Table 4.16 Failure Rate Data on NTDS Input Buffer BIT

One can see that the BIT circuit has a failure rate of over three times that of the circuit it is checking. When a failure is detected, it is probably in the BIT circuitry and not in the monitored circuitry. If BIT is absolutely necessary on some of the control circuitry, it would be better to duplicate the circuit and compare outputs. It is therefore recommended that the control BIT circuitry be removed from the NTDS Input Buffer. Figure 4.45 shows the NTDS Input Buffer minus the control circuitry BIT and with the recommended parity BIT.

The parity bus concept provides a check on a part of this module. While the increase in hardware is small, the amount of circuitry checked is not great. The NTDS buffers are difficult to check economically using commercially available ICs because the voltage levels used are not compatible with economical, reliable TTL circuits.

The control portion of the NTDS Input Buffer is difficult to check because it is a collection of independent asynchronous lines. The control

circuit monitor method using two one-shots and a parity checker is not desirable because its failure rate is much higher than the circuit it is checking. If a complete check of this module is desired, it should be done off-line at the system level.

4.4.5 NTDS Output Buffer

The NTDS (Naval Tactical Data System) output buffer provides an interface from the QED family which uses TTL logic levels to an NTDS computer or peripheral which uses negative voltage levels. The bus parity approach provides a way to check the TTL latches but because of the voltage levels involved, it is impractical to check the NTDS output lines in a similar manner.

Included in the current NTDS Output Buffer is a BIT circuit that checks a portion of the control circuitry. This method uses an up/down counter in conjunction with a flip-flop controlling the up/down function to make a decision on the correct operation of the module. When the module receives a ready signal from the NTDS device, it causes the counter to count up one count. The responding output from the interface module causes the control flip-flop to toggle. The next ready signal then causes the counter to count down one count which is then followed by a response which toggles the control flip-flop again. In this manner the counter alternates between the two counts. When an error occurs, the counter has a count over or under the two acceptable numbers and this activates the pass/fail line.

This circuitry has a fairly high failure rate and does not totally check the control circuitry. Table 4.17 is a list of the BIT circuitry and the control circuitry that is being partially checked.

From Table 4.17 one can see that the failure rate of the BIT circuit is almost as high as the circuit it is checking. (In fact, it is 81% of it.) While this failure rate increase is better than with duplication, it must be noted that some errors will not be detected by the proposed checking method. There are places where a single stuck-at-fault will not cause the fail line to be activated, but will cause the module to operate improperly. While this test method has some merit, it should not be included in the final module specification unless it is absolutely

<u>Parts In BIT</u>	<u>Failure Rate/Device</u>	<u>Failure Rate</u>
1 5432 Quad OR	.03365	.034
4 capacitors	.2	.8
4 resistors	.08	.32
1 diode	.04	.04
$\frac{1}{2}$ 74123 dual one-shot	.03915	.02
1 54190 Updown counter	.0835	.084
1/6 5404 Hex inverter	.03915	.006
$\frac{1}{2}$ 5474 Dual D F/F	.04365	.0168
$\frac{1}{2}$ 5400 Quad Nand	.03365	<u>.0168</u>
		1.34

Parts that BIT at least partially checks

$\frac{1}{2}$ 5417 Hex Inverter	.03915	.02
1 $\frac{1}{2}$ 54123 dual one-shot	.03915	.059
3/4 5408 Quad-And	.03365	.025
1 5474 Dual D F/F	.04365	.044
$\frac{1}{2}$ 5400 Quad Nand	.03365	.017
1 7097 Buffer	.0414	.0414
9 resistors	.08	.72
3 diodes	.04	.12
3 capacitors	.2	<u>.6</u>
		1.646

Table 4.17 Failure Rate Data on NTDS Output Buffer BIT

necessary to provide some type of check on the control function. Figure 4.46 depicts the NTDS Output Buffer without the control BIT and with the recommended BIT using the SIIB.

Like the NTDS input buffer the QED parity bus concept checks a small portion of the circuitry with a small increase in hardware. Typically, the control circuitry is hard to economically check. It is possible to duplicate the circuitry and compare outputs, but generally this is not economical as cost more than doubles. However, this approach provides the most complete built-in-test. The control circuit monitor method using the up/down counter, while providing some check of the circuitry, increases the failure rate only slightly less than duplication. Duplication would provide a complete check of the control circuitry rather than only a partial check. If a complete check of the module is necessary, it can be done off-line at the system level.

4.4.6 Application of the Analytic Measures to the Recommended BIT Approaches

As for the other module classes, to quantify the gains and cost of the BIT techniques, it is necessary to apply the analytic measures described in Section 3.0. These measures provide a good indication of the additional costs involved, and they also provide a guide to the effectiveness of the BIT techniques.

A summary of the BIT evaluation for the interface class modules is shown in Table 4.18. One can see that the BIT provides an effective error detection capability for the switch module modules is modest. It appears that the failure rate and package count increases for the 8-bit switch are high, but actually this arises from the fact that the BIT circuitry is of modest complexity and the non-BIT circuitry is of much lower complexity. Thus, the ratio appears unfavorable.

The BIT for the other modules in the class provide much less error detection capability. The major reason for this arises from the fact that these modules interface equipments that use non-TTL interface voltage levels. This prohibits the use of off-the shelf TTL packages

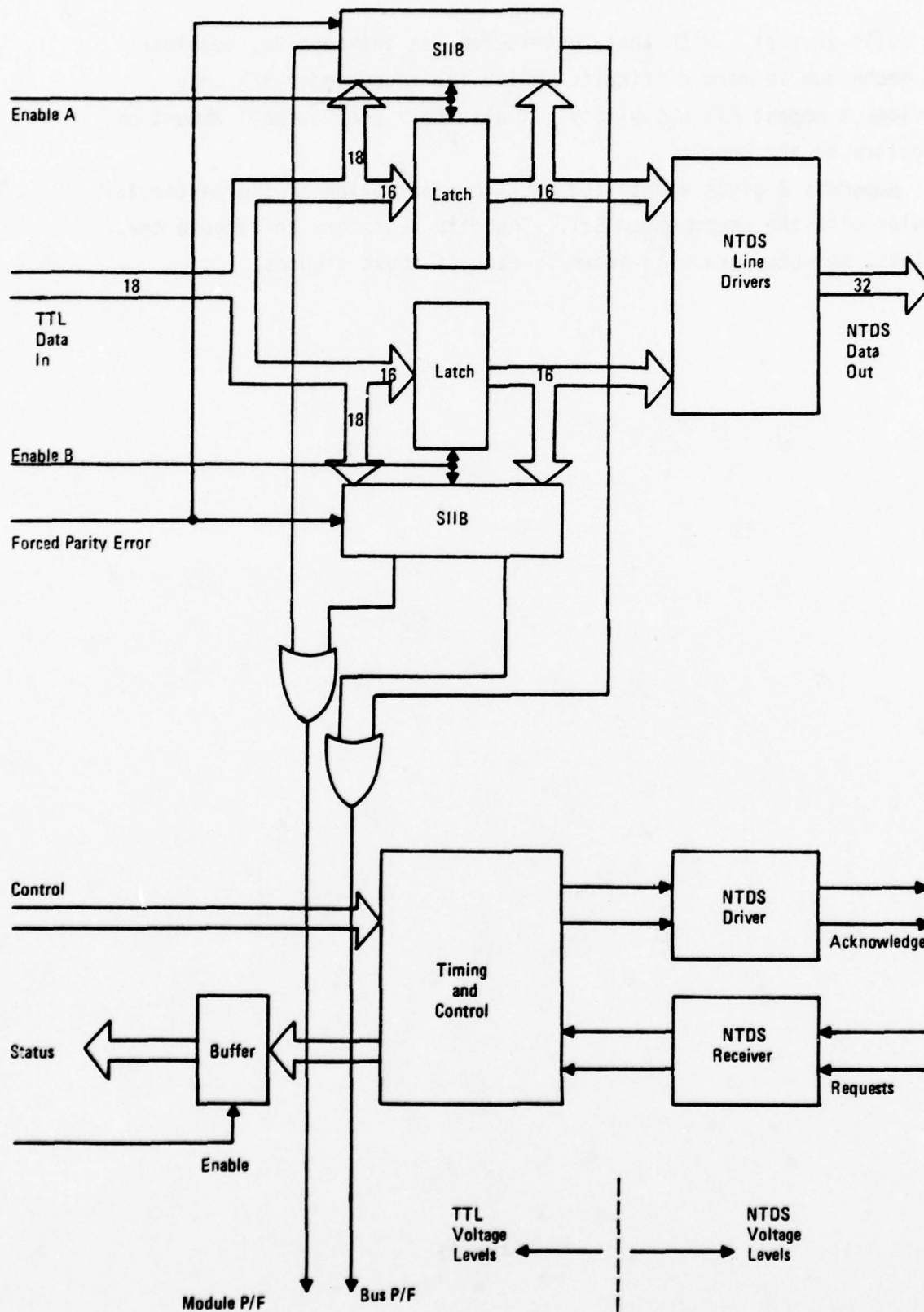


Figure 4.46 NTDS Output Buffer With Recommended BIT

for built-in-test. With that restriction, an inexpensive, complete BIT technique is more difficult. While the recommended BIT only provides a modest BIT capability, it also only adds a small amount of circuitry to the module.

Appendix B gives a detailed package description of the particular modules with the recommended BIT. The data necessary to compute the analytic measures are also shown in each of these figures.

Parameter	Dual 8-Bit Switch	Dual Parallel 8-Bit Interface	Asynchronous Serial Inter- face	NTDS Input Buffer	NTDS Output Buffer	Units
Percent of Gates Monitored	83	88	33	34	45	%
Percent of Packages Monitored	85	74	36	38	43	%
Number of Cycles for Test	0	0	0	0	0	-
Ratio of BIT Packages to Total Module Packages	35	19	11	16	18	%
Failure Rate Without BIT	0.7	1.5	6.6	5.4	4.8	/10 ⁶ Hr
Failure Rate With BIT	1.2	1.8	6.8	5.7	5.1	/10 ⁶ Hr
Ratio of BIT Failure Rate to Total Module Failure Rate	40	16	2	5	6	%
Power Consumption of BIT	0.5	0.3	0.2	0.3	0.3	Watts
Ratio of BIT Power Consumption to Total Power Consumption	9	7	5	7	6	%

Table 4.18 Analytic Measures Tabulation for Interface Class

5.0 NET GAIN EXAMPLE USING RECOMMENDED BIT

The overall objective of this section is to demonstrate how module level BIT can lead to reduced mean-time to repair (MTTR) by making it easier to localize faults and thereby facilitate repair of defective systems. It is important to note that the possibility of rapid repair exists in the initial instant because of the basic functional modularity concept. At the same time it should be apparent that in order to effect system repair by module replacement, faults must

- 1) be detected,
- 2) brought to the attention of the system user,
- 3) be localized to the replaceable module level and
- 4) replacement effected.

The timeliness and efficiency with which these steps are carried out ultimately determines the system MTTR and hence the system availability.

In order to quantitatively evaluate the potential effectiveness of the recommended BIT approach, an example digital subsystem has been designed using QED modules and the recommended module level BIT circuits. This digital subsystem is used as a vehicle to demonstrate the potential net gain resulting from the recommended module level BIT approach. In addition the example system serves as a logical entrée into the problems and opportunities associated with subsystem level built-in-test equipment (BITE). Representative of the potential BITE problems are the pass/fail (P/F) interface timing questions which must be answered in any practical design incorporating module level BIT circuits. Demonstrative of the subsystem level BITE opportunities is the chance to distribute in a cost effective way, the total built-in-test facilities throughout digital system hierarchies (including the software structures common to most present day programmable digital systems). Further consideration is given to these issues in the following sections.

Representative of those subsystems which exist for the purpose of augmenting the capabilities of a host digital system are special purpose hardware units designed to perform specific tasks. Among such subsystems are peripheral memory devices commonly used for bulk data storage and arithmetic devices designed for maximum computational through-put. In addition there are subsystems which utilize both extensive computational facilities and large amounts of data memory. Representative of this latter class of subsystems are array data processors. Since array processors represent attributes of both memory intensive and arithmetic intensive processing, this is a reasonable class from which to choose an example subsystem.

A particularly representative array processor is the fast Fourier transform (FFT) processor which uses the FFT algorithm to efficiently compute the discrete Fourier transform (DFT). While there are various organizations of FFT processors [22] [23] [24], the one most suited to high data rate applications is the cascade or "pipeline" FFT [24]. Since many of the QED modules are designed for high speed signal processing application, it is reasonable to consider machine architectures which make the most of such attributes. In addition there currently exists a sequential FFT processor designed with QED modules [25] and therefore a cascade design offers the chance to see how QED modules can be used in a different array processor embodiment.

5.1 Cascade Fast Fourier Transform (FFT) Processor Design

The efficient organization of computations in the form of algorithms can be realized in hardware especially designed to match the computational and data storage requirements of a given application. The cascade FFT is an example of a digital signal processing architecture which not only performs the basic computations required by the FFT algorithm but also computes and stores intermediate results in a way that achieves maximum data through-put with minimum speed arithmetic and storage elements. This is accomplished by first computing those partial

results which are needed to compute succeeding partial results and so on until the first input word pair from the data set has completely progressed through the algorithm.

The cascade FFT processor is best explained by a flow chart and a block diagram. The flow chart depicted in Figure 5.1 illustrates the Cooley-Tukey, base-2, decimation-in-time, pre-scrambled FFT algorithm for 8 input data points. This flow chart may be readily extended to larger input data sets which are powers of 2. The basis of the cascade approach is illustrated by the heavy lines. It can be seen that those elemental computations (or "butterflies" as they are referred to in the literature) in stage 1 whose outputs are necessary to compute the first butterfly in stage 2 are initially computed. When these partial results are completed, the first butterfly in stage 2 can be computed. For larger input data sets these results are simply extended necessitating computation of more butterflies like that shown expanded in Figure 5.2. This is the essence of the cascade FFT.

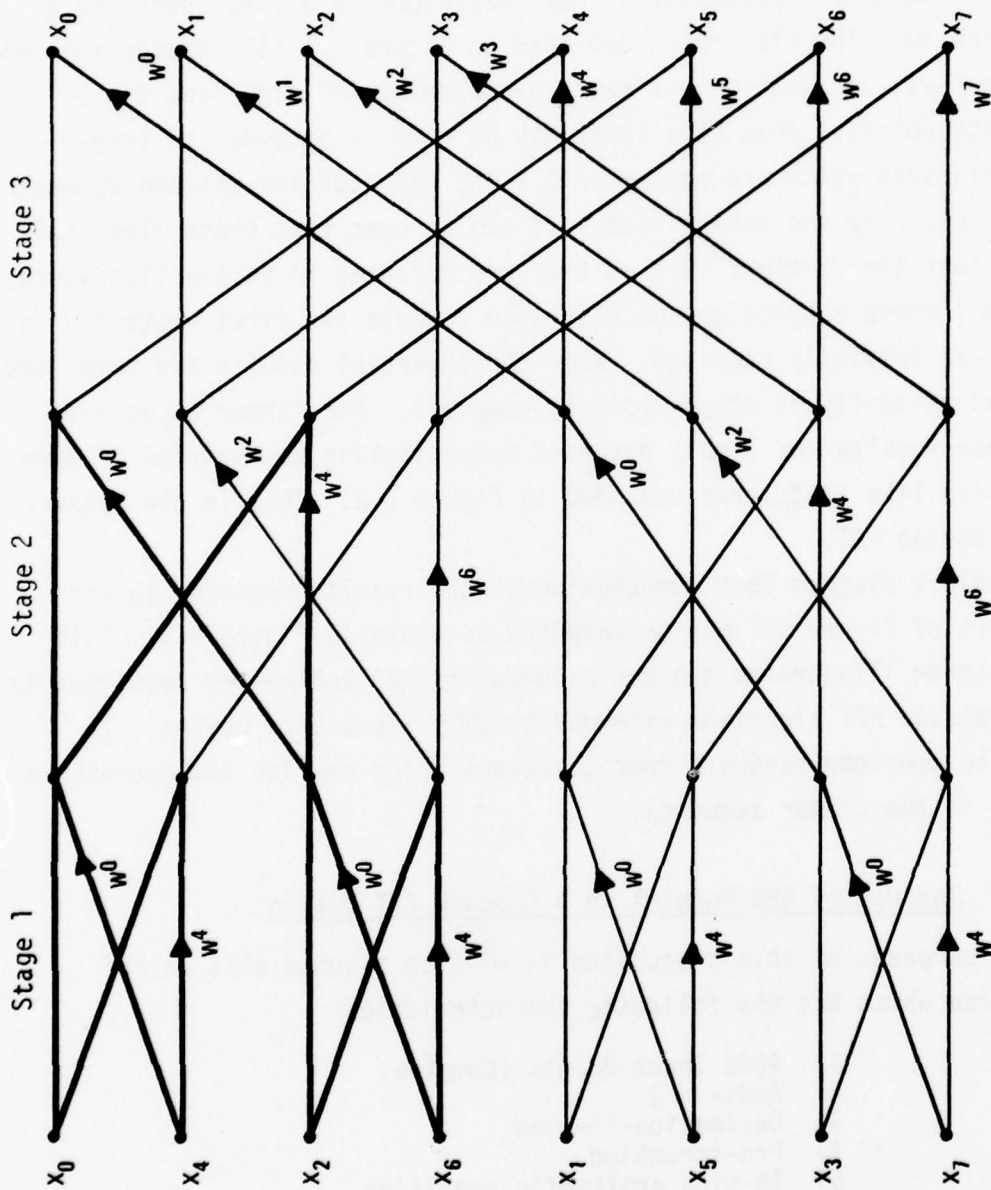
In block diagram form computation of the results depicted in the flow-chart of Figure 5.1 may be computed as shown in Figure 5.3. This block diagram illustrates the basic computational and memory requirements of the cascade FFT algorithm extended to 4096 input data points. It remains to consider hardware configurations which realize the operations depicted in the proper sequence.

5.2 The Use of QED Modules in a Cascade FFT Design

For purposes of this discussion it will be assumed that an FFT is required which has the following characteristics:

- 1) 4096 Input Points (Complex)
- 2) Radix - 2
- 3) Decimation-in-time
- 4) Pre-scrambled
- 5) 16-bits arithmetic precision
- 6) 16-bits coefficient precision (Real & Imaginary)

As a design objective it is desirable to maximize the total types of QED modules. This may be done at the expense of using a greater total number of logic cards



$$X_K = \sum_{n=0}^{N-1} x_n W^{nK}, \text{ where } W = e^{-j(2\pi/N)}$$

Figure 5.1 FFT Algorithm. 8-Point Transform Pre-scrambled, Radix 2, Decimation in Time

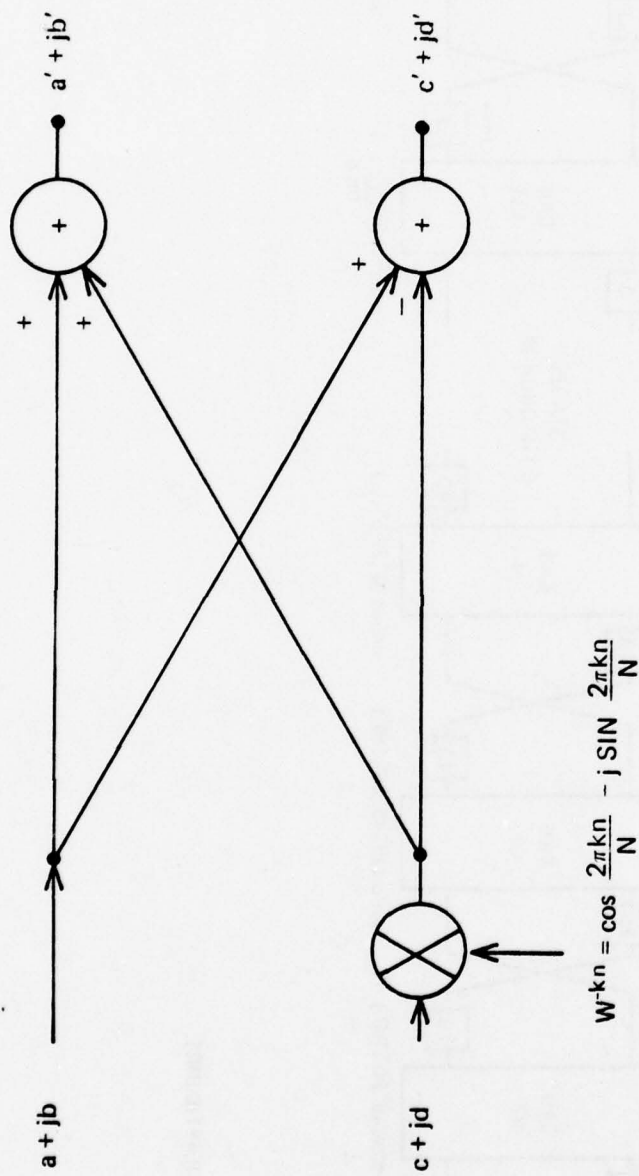
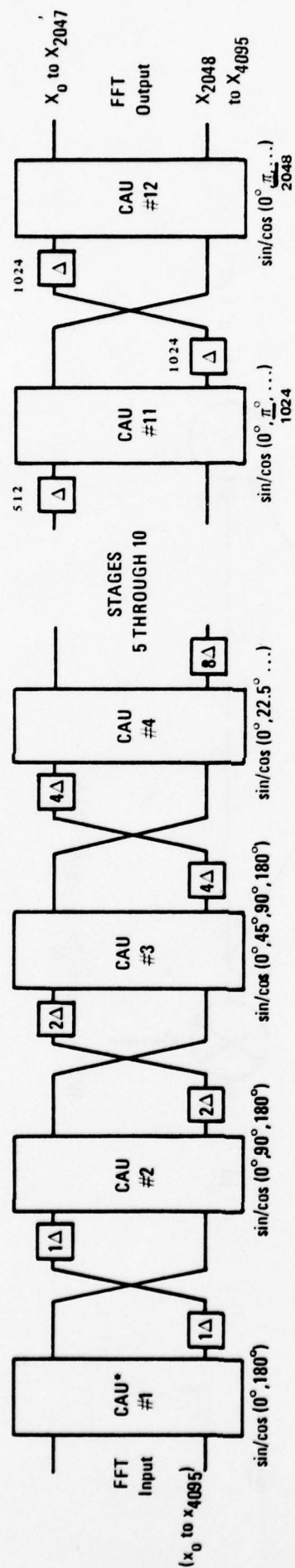


Figure 5.2 FFT Elementary Computation Unit (Butterfly)



*CAU - COMPLEX ARITHMETIC UNIT.

Figure 5.3 Cascade FFT (Radix-2, Decimation in Time)

than would be necessary if non-QED modules were used for some functions. However, since the purpose of this example is to illustrate the effectiveness of the recommended QED module BIT circuits, reducing the total number of QED modules is secondary.

To determine the number of each QED module which must be used in the cascade FFT, a more detailed block diagram is necessary. To this end, the block diagram shown in Figure 5.4 is relevant. This block diagram illustrates the overall organization of an FFT processor which uses a single arithmetic unit time-shared between 12 computational stages. A detailed diagram of the basic FFT computational element and the equations computed using this arithmetic unit is given in Figure 5.5 [24]. The QED modules required to implement the indicated computations are:

<u>QED Module</u>	<u>Application</u>	<u>Quantity</u>
Arithmetic Logic Unit (ALU)	Butterfly Add/Sub	6
Parallel Multiplier	Butterfly Multiplication	8

Table 5.1 FFT Arithmetic Unit Modules.

The numbers given in Table 5.1 assume that two real, 16X16 - BIT multipliers are time-shared using the input data delay arrangement shown. The arithmetic unit input/output interfaces the Partial Result RAM and the coefficients are supplied from read-only-memory (ROM). The QED modules required to implement these memories and the addressing circuitry are given in Table 5.2.

<u>QED Module</u>	<u>Application</u>	<u>Quantity</u>
Random Access Memory (RAM)	Partial Result Storage	32
8-Bit Index Counter	RAM Addressing	2
Read Only Memory (ROM)	Coefficient Storage	4
8-Bit Index Counter	ROM Addressing	2

Table 5.2 FFT Memory/Address Modules

The remaining operational portion of the cascade FFT is the control section. The function required is basic timing information for data transfer and arithmetic control. The modules listed in Table 5.3 are required to provide

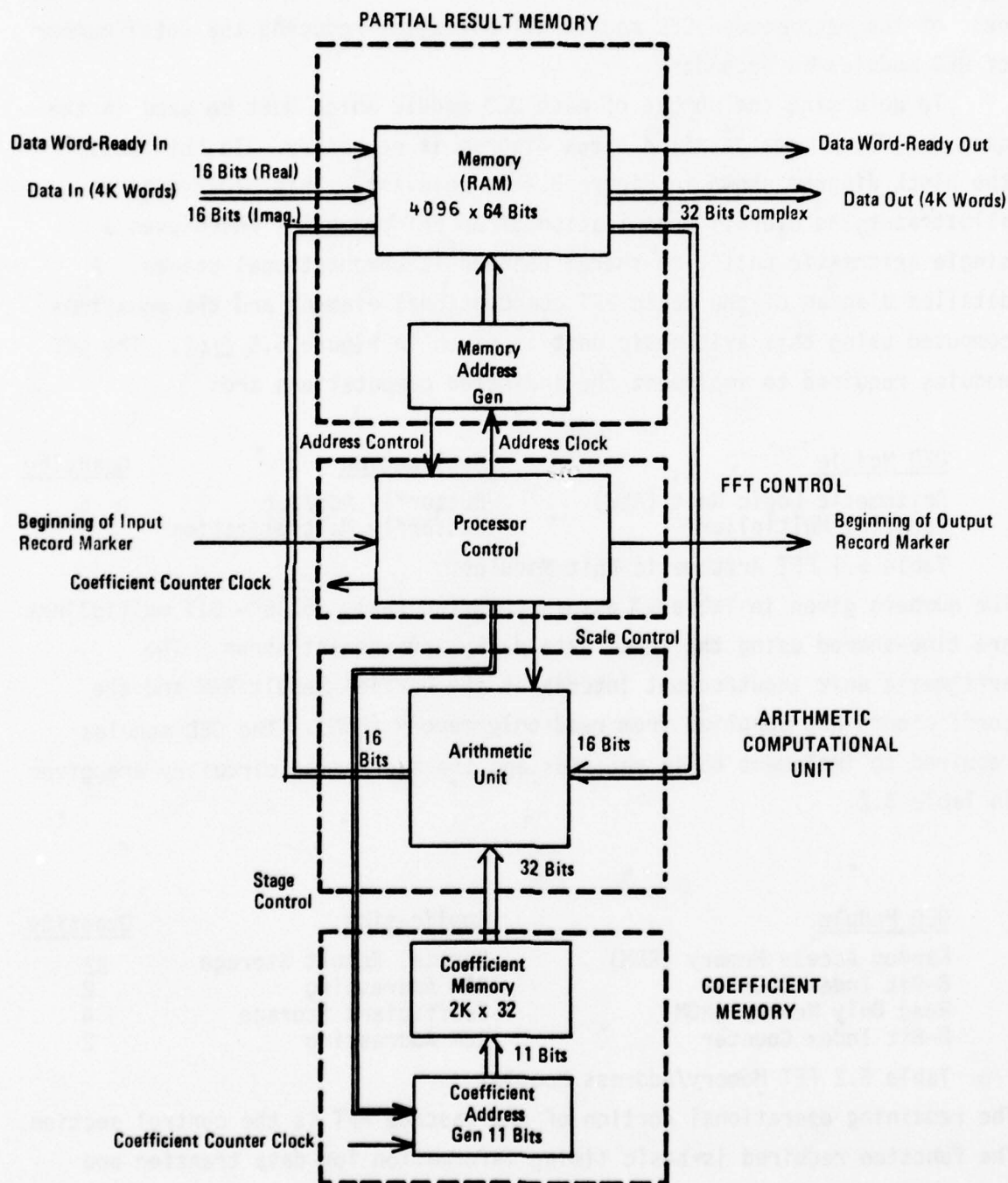


Figure 5.4 Time-shared Cascade FFT Processor

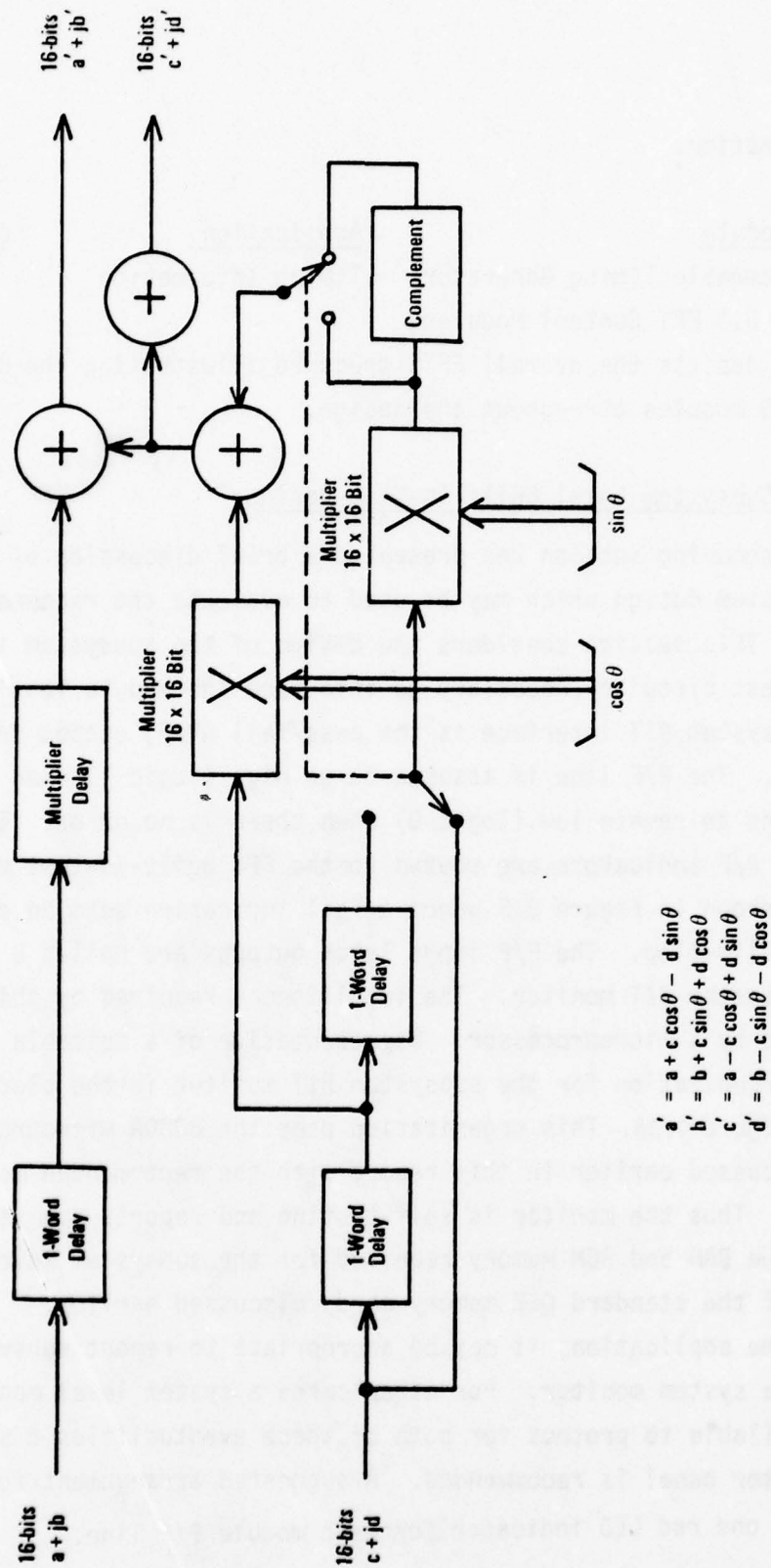


Figure 5.5 Basic FFT Computational Unit Organization

this information.

<u>QED Module</u>	<u>Application</u>	<u>Quantity</u>
Programmable Timing Generator	Timing Information	2

Table 5.3 FFT Control Modules

Figure 5.4 depicts the overall FFT structure illustrating the distribution of QED modules throughout the design.

5.3 Subsystem Level Built-In-Test Design

The preceding section has presented a brief discussion of a candidate subsystem design which may be used to evaluate the recommended BIT circuits. This section considers the design of the subsystem level built-in-test circuitry necessary to interface the module level BIT. The module/subsystem BIT interface is the pass/fail (P/F) output from each QED module. The P/F line is assumed to go high (logic 1) when there is an error and to remain low (logic 0) when there is no error. These individual P/F indicators are routed to the FFT built-in-test monitor interface shown in Figure 5.6 where a fail indication sets an edge triggered flip-flop. The P/F input latch outputs are polled 8 at a time by the subsystem BIT monitor. The intelligence required by this monitor is supplied by a microprocessor. Representative of a suitable microcomputer organization for the subsystem BIT monitor is the block diagram shown in Figure 4.36. This organization uses the 8080A microprocessor module discussed earlier in this report with the recommended module level BIT. Thus the monitor is self-testing and reports faults to itself. The RAM and ROM memory required for the subsystem level monitor consists of the standard QED memory cards discussed earlier.

In some applications it may be appropriate to report subsystem faults to a system monitor. For other cases a system level monitor may not be available to protect for both of these eventualities a subsystem fault monitor panel is recommended. A suggested arrangement for this display is one red LED indicator for each module P/F line.

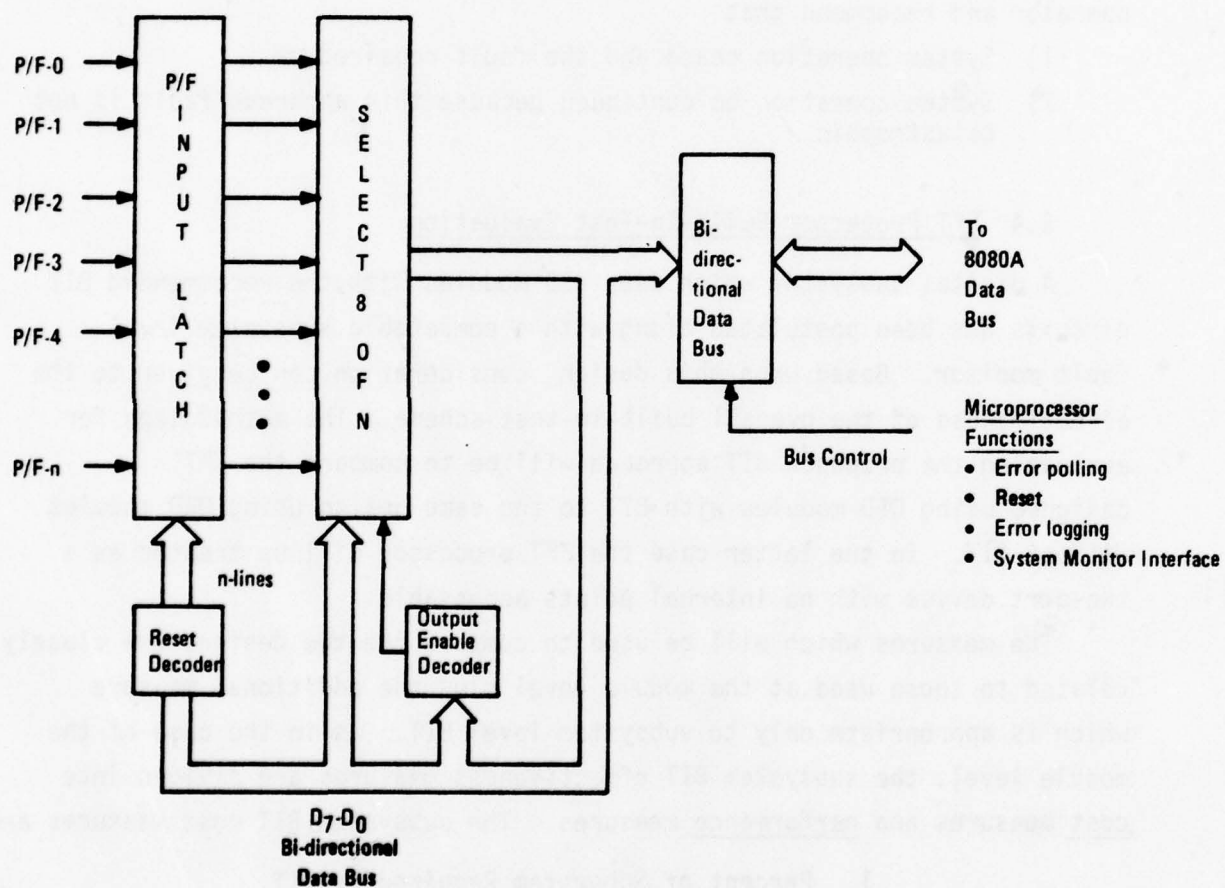


Figure 5.6 FFT Built-In-Test Monitor Interface

Where the subsystem built-in-test monitor must report to a systems level fault monitor, a suitable interface must be established. As a minimum the following information should be exchanged at this interface:

- 1) The name of the faulty module(s),
- 2) The frequency of error occurrence.

Based upon this information, the system monitor can flag the system operator and recommend that

- 1) System operation cease and the fault repaired or
- 2) System operation be continued because this apparent fault is not catastrophic.

5.4 FFT Processor Built-In-Test Evaluation

A digital subsystem which uses QED modules with the recommended BIT circuits has been postulated along with a compatible subsystem level fault monitor. Based upon this design, consideration can be given to the effectiveness of the overall built-in-test scheme. The methodology for evaluating the proposed BIT approach will be to compare the FFT designed using QED modules with BIT to the same design using QED modules without BIT. In the latter case the FFT processor will be treated as a two-port device with no internal points accessible.

The measures which will be used to compare the two designs are closely related to those used at the module level plus one additional measure which is appropriate only to subsystem level BIT. As in the case of the module level, the subsystem BIT effectiveness measures are divided into cost measures and performance measures. The subsystem BIT cost measures are

1. Percent of Subsystem Required by BIT.

$$PS = \frac{NB}{NB + NQ} \times 100, \quad (5.1)$$

where PS = Percent Subsystem in BIT,
NB = Number of ICs in BIT,
NQ = Number of ICs in Subsystem

2. Power Consumption of Subsystem BIT

$$P_B = \sum_i PB_i + PM \quad (5.2)$$

where P_B = Total Power required by BIT,
 PB_i = Power required by Module BIT,
 PM = Power required by subsystem monitor.

3. Subsystem Failure Rate Increase Due to BIT

$$BFR = \frac{\sum_i FRB_i + FRS}{\sum_i FRB_i + FRS + \sum_j FRM_j} \quad (5.3)$$

where FRB_i = Failure rate of module BIT circuitry
 FRS = Failure rate of subsystem monitor
 FRM_j = Failure rate of module without BIT

4. Number of Cycles Required for Testing

NCT = Number of clock cycles required to test subsystem (5.4)

The subsystem BIT performance measures are

1. Percent of subsystem gates monitored

$$PIM = \frac{NMQ}{NMQ + NNQ} \times 100 \quad (5.5)$$

where NMQ = Number of Gates Monitored in Subsystem Design

NNQ = Number of Gates not Monitored in Subsystem

2. Percent of Cycles Monitored

$$PCM = \frac{\text{No. of Cycles Monitored}}{\text{Total Number of Cycles}} \times 100 \quad (5.6)$$

3. Subsystem Mean-Time to Repair

$$SMTTR = \frac{\sum_i FD_i + \sum_i FL_i + \sum_i FR_i}{N} \quad (5.7)$$

where FD_i = Time to detect i-th fault

FL_i = Time to fault localize i-th fault

FR_i = Time to repair i-th fault

N = Total number of failures

All of these performance measures may be applied in a straightforward way to the FFT processor subsystem described in Sections 5.1 and 5.2 except the last one. In determining the subsystem MTTR, the time to detect a fault, FD , and the time required to determine which module is faulty (i.e., fault localization), FL , are easily determined. However, the time to repair subsystem faults, FR , must be determined, in part, by the system level fault monitor, the standard operating procedures in effect at the system point of application, system operator/maintenance manpower, etc.

One way to quantitatively compare the FFT's with and without BIT is to assume the same repair time once the fault is detected and localized. Since this time interval is, for the most part, determined by human interaction (e.g. an operator reading a teletype message indicating which module is faulty, a repair person retrieving a replacement module from stock, etc.) this time will be assumed to be the same for both systems. Thus when comparing the MTTR of the subsystem with BIT with that of the example subsystem without BIT, only the fault detection and fault localization components of SMTTR in equation 5.7 will be considered.

The results of the computations indicated in equations 5.1 through 5.7 based on the QED modules indicated in Tables 5.1 through 5.3 are given in Table 5.4 below.

Module	Quantity	Failure Rate (/10 ⁶ hrs)			
		Module		Composite	
		Without BIT	With BIT	Without BIT	With BIT
Arithmetic Logic Unit	6	1.32	2.43	7.92	14.54
Parallel Multiplier	8	1.36	3.00	10.88	24.00
Random Access Memory	32	3.55	4.16	113.60	133.12
8-Bit Index Counter	4	2.30	4.37	9.20	17.48
Read Only Memory	4	2.21	2.45	8.84	9.80
Programmable Timing Generator	2	2.93	3.46	5.86	6.92
		Total FR		156.30	205.90

Percent of Subsystem Required by BIT = 32.6%

Power Consumption of Subsystem BIT = 54 watts

Subsystem Failure Rate Increase Due to BIT = 25.2%

Number of Cycles Required for Testing = 0%

Percent of Subsystem Gates Monitored = 87.2%

Percent of Cycles Monitored = 100%

Table 5.4 Example FFT Processor Subsystem

It is at this point in the analysis that the FFT processor with integral built-in-test begins to be attractive. Specifically the amount of time necessary to detect a fault in the system with BIT is at most a few seconds. This is not true of the FFT without BIT since the initial indication of faulty operation may be when an operator notes that erroneous results are being produced by the system. Quite often in digital signal processing systems, this is only after minutes and perhaps even hours have elapsed.

The other component of concern in determining BIT effectiveness is the fault localization time. Fault localization time in systems without

BIT is often quite variable since it is dependent directly upon the skills of particular maintenance personnel. On the other hand, where effective BIT techniques do exist, the skill required by maintenance personnel is minimized.

In order to determine the overall effectiveness of the built-in-test in the example subsystem, the system availability, A, may be computed where

$$A = \frac{MTBF}{MTBF + MTTR} \quad (5.8)$$

Since MTBF is defined as the inverse of system failure rate, the system MTBF can be determined by summing the failure rates of the QED modules required by the system. The system availability can then be computed using equation 5.8. In order to determine the recommended BIT effectiveness in increasing system availability in even more complex systems, it can be assumed that multiple FFT subsystem comprise the total example system. This has been done and the results are given in Table 5.5. The MTTR's assumed in this table are intended to be representative of the range of repair times which might be required to detect and localize faults. Recall that the repair time (i.e., the time to actually replace a faulty module) should be added to these numbers in order to arrive at the total system MTTR.

The system availability results given given in Table 5.5 have been plotted as a function of system complexity with MTTR as a parameter. Figure 5.7 illustrates quantitatively the net gain in system availability where BIT is used for fault detection and localization. It should be noted that the availability of the system with BIT is insensitive to system complexity. This illustrates the important fact that extremely complex systems with BIT are just as easy to repair as uncomplicated systems without BIT. It is apparent from Figure 5.7 that the net gain in system availability increases as the system complexity increases.

There are various aspects of increased system availability which result from effective BIT that are not so easy to evaluate quantitatively. For example, in many military systems, the time at which a failure occurs can be critical. The MTBF measure has little meaning if a shipboard fire control computer operates perfectly for a year and fails during a combat

<u>Number of Equivalent FFT's</u>	<u>MTTR (hrs)</u>	<u>MTBF without BIT (hrs)</u>	<u>MTBF with BIT (hrs)</u>	<u>Availability without BIT</u>	<u>Availability with BIT</u>
1	0.25 hr	6398	4857	0.99996	0.99996
2	0.25 hr	3199	2428	0.99992	0.99990
4	0.25 hr	1599	1214	0.99984	0.99980
8	0.25 hr	800	607	0.99969	0.99959
16	0.25 hr	400	303	0.99938	0.99918
1	1.00 hr	6398	4857	0.99984	Same as Above*
2	1.00 hr	3199	2428	0.99969	
4	1.00 hr	1599	1214	0.99938	
8	1.00 hr	800	607	0.99875	
16	1.00 hr	400	303	0.99750	
1	10.00 hr	6398	4857	0.99844	Same as Above*
2	10.00 hr	3199	2428	0.99688	
4	10.00 hr	1599	1214	0.99379	
8	10.00 hr	800	607	0.98765	
16	10.00 hr	400	303	0.97560	

*Assumes Constant MTTR = 0.25 hours

Table 5.5 System Availability With and Without BIT

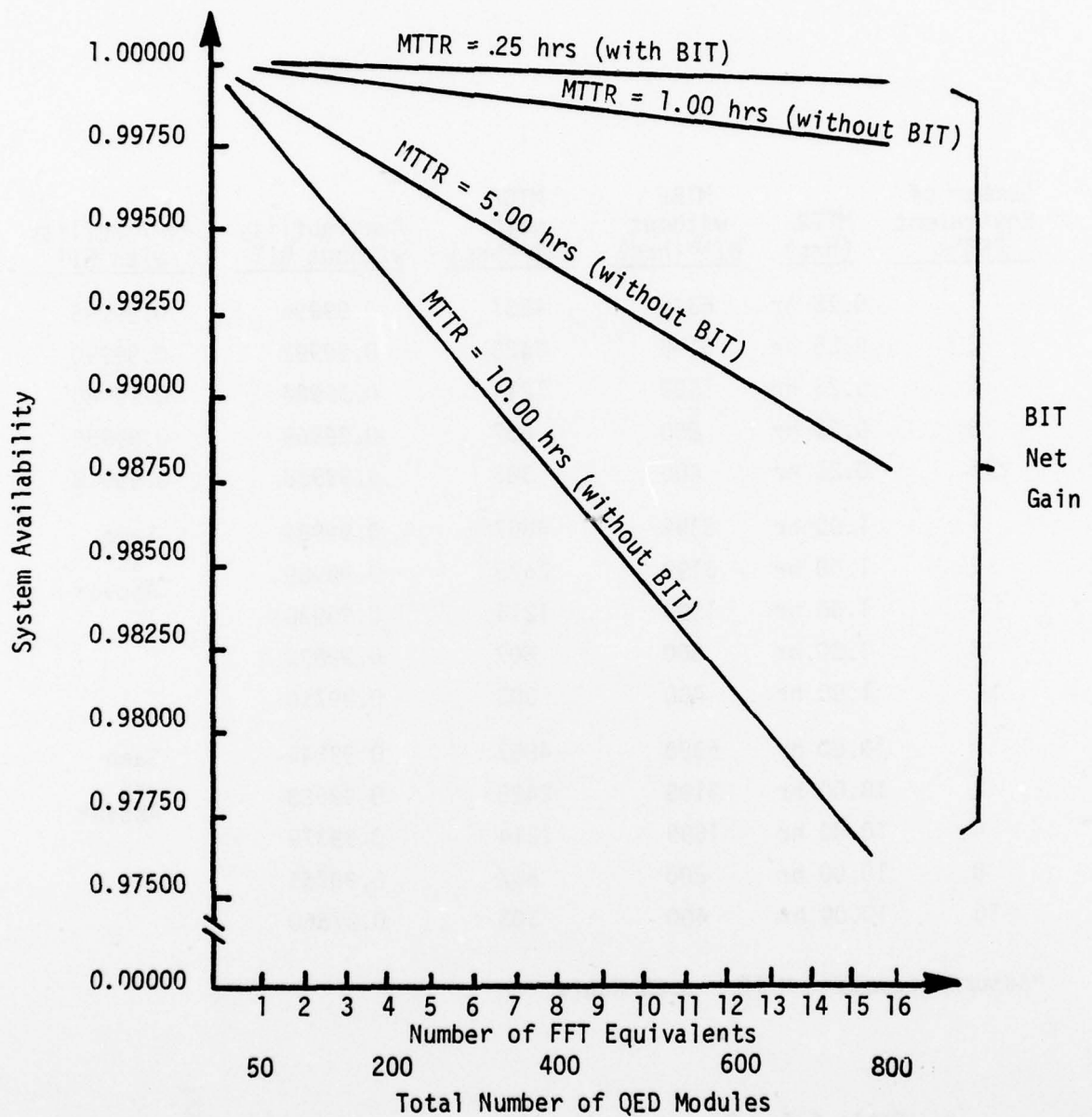


Figure 5.7 System Availability Versus System Complexity With and Without BIT

support mission. In such an instance in a system with effective BIT, a fault could be detected, localized, and repaired in time to allow the system to be used during the mission. And finally it should be apparent that the lower skill level of maintenance personnel required to effectively repair a system with effective BIT can result directly in reduced system life cycle cost; whereas in a system without BIT, more highly trained personnel would be required, thus resulting in increased life cycle cost.

6.0 SUMMARY AND RECOMMENDATIONS

This study has addressed the problems of continuous, on-line testing of functional digital modules using test techniques which can reside on the modules and be applied to a variety of module classes. The issues and results relevant to these techniques are summarized in the following discussion.

6.1 Summary of BIT Concepts and Approaches

There are three fundamental ideas which impact this work and have a general applicability to the entire topic of functional module built-in-testing. These ideas are:

- (1) a classification of possible approaches to module level BIT,
- (2) an analytic approach to evaluating and selecting these approaches, and
- (3) a concept of standardizing these BIT approaches as part of system design.

Within the taxonomy of BIT approaches (Section 2.1), the focus is on concurrent fault monitoring techniques. Within this category there are two techniques which are emphasized, arithmetic codes and sampled monitoring. Arithmetic coding techniques apply to a substantial number of module types and provide concurrent on-line diagnosis. Sampling applies to a broader class of modules but does not provide total concurrency.

As discussed in Section 3.0, the selection of a particular BIT hardware approach requires the evaluation of certain cost and performance tradeoffs. The basis for decision must be made as analytic as possible. This requires that a set of meaningful measures be derived to quantify the answers to the questions, "How much does the addition of the BIT hardware Cost?" and "How effective is the BIT hardware in diagnosing faults?". A set of measures designated as the monitoring capability index (MCI) are defined which are suited to the particular class of BIT of interest (on-line monitoring).

There are a range of interpretations which may be applied to the idea of standardizing approaches to BIT hardware (Section 2.5). There are two interpretations applied here. The first states that a single standard BIT circuit suitable for use on most modules can be defined if the performance tradeoff incurred by sampled monitoring is acceptable. The second point of view defines a family of BIT circuits which are applicable to monitoring

subfunctions typically found in module implementations. The most important impact of standardization is to facilitate the use of BIT circuits as a part of the normal design procedure.

The recommended built-in-test technique for all of the QED modules in the memory class is word parity (Section 4.1). This approach alone gives single bit per word error detection capability on over 98 percent of the gates on the module. The cost involved is approximately a 25% increase in package count and a failure rate increase of less than 18%. The other candidate approaches (memory duplication, single bit error correction, and off-line checking) cost, in terms of added packages and failure rate, significantly more than word parity and offer only a slight improvement in error detection capability.

The proposed Standard Interconnection and Interface BIT (SIIB) can be added to the modules in the memory class. This would further increase the percent of packages (and gates) in which the BIT can detect error. The additional cost in package count is zero because the SIIB replaces the parity generator/checker already specified for BIT. The failure rate increase is limited to the small number of additional gates necessary to implement the SIIB over a conventional parity generator. Thus, the combination of storing word parity in the memory and checking parity with the SIIB gives nearly total single bit per word error detection capability with only a modest increase in failure rate and required number of packages.

A standard residue code approach to built-in-test has been defined for members of the process class module family in Section 4.2.3. Examples are given which illustrate the effectiveness of these codes when used for checking weighted number systems such as that used by the QED process class modules. It has been shown that all single bit errors may be detected by a low cost integer residue code.

A standard circuit approach has been identified which may be used to implement an odd integer residue code generation and checking approach for all members of the process class module family including the microprocessor. This circuit coupled with the standard I/O parity circuit described in Section 4.1.5. provides an effective means of testing the QED process class modules.

Control class modules are characterized by having many input and output

interfaces which may be only indirectly related (Section 4.3). Because of this, standard testing techniques are not generally applicable. In fact replication is the approach often described in the literature as a means of checking control circuits.

In the case of the Programmable Timing Generator provisions are made for replicating the timing (address) generation portion of the module. The remainder of the circuit is checked by storing parity in the read only memories and checking parity at the module outputs using the recommended ROM built-in-test techniques described in Section 4.1.2. Thus, the system designer has the option of using replication if the reliability requirements of his system warrant. In either case, concurrent error detection is performed automatically on the ROM portion of the module.

The Priority Encoder has the interesting property that a portion of its circuitry may be self-checked at the system level. The system designer may take advantage of this fact through careful construction of the system software. Standard I/O parity is used to check the major part of the remainder of the Priority Encoder module. This module serves as an excellent example of where system level software and module level BIT can and should be mutually supportive. In addition, partial duplication is an alternative BIT approach (Section 4.3.2) which may be used.

The built-in-test technique for the interface class modules provides a parity check on the data path (See Section 4.4). This is a low cost BIT approach that generally only requires a parity generator and a parity checker. This small amount of additional hardware is capable of detecting single bit faults in the data. The Standard Interconnection and Interface BIT (SIIB) can be added in place of the parity generator/checker to provide a check of the TTL input buffers and output buffers as well as the internal data handling logic. This approach provides a check of a major portion of the logic on the module at a modest cost.

The control section of the modules however does not benefit from this BIT approach. The nature of the control function makes coding techniques unusable. With the efficiencies of coding removed from possible BIT methods, the most likely built-in-test approach is duplication. While this method in general is costly in terms of added hardware, it provides a complete check on the control function.

In summary, the proposed standard BIT circuits recommended for the QED modules are:

- (1) Standard Interconnection and Interface BIT (SIIB)
- (2) Standard Residue (Modulo-3) Generator
- (3) Standard Residue (Modulo-3) Checker
- (4) Standard Timer Circuit.

These few circuits along with limited partial duplication on selected modules provide the basic capability for testing a wide variety of functional digital modules. This document has presented logic gate level descriptions of the proposed standard BIT circuits. The application of the recommended BIT circuits to each QED module has been discussed and the cost and effectiveness evaluated based upon specified measures.

The proposed BIT circuits share the common attributes that the number of logic gates and pins required are such that each may be realized with reliable monolithic circuits capable of operating at the high data throughput rates necessary to provide continuous, on-line testing for QED modules. The circuit level documentation presented in this report provides the basis on which to proceed with development of computer engineering evaluation circuits.

Finally, it should be emphasized that the built-in-test circuits presented in this preliminary specification may be applied to the testing problems of a wide range of functional modules of which the QED family is a subset.

6.2 Recommended Further Work

The present study has shown that it is feasible to do concurrent fault monitoring with module-resident hardware which is a small percentage of the total module circuitry. In performing this study, a number of issues and possible BIT techniques were encountered which were either beyond the scope of this study or which could not be fully explored in the allotted time. Because of the promising nature of the results of the present study and the potential benefits to the Navy and the other services which could result from implementation of the recommended module level BIT approaches, the following suggestions for further study are briefly discussed.

Digital Module Partitioning for Testability - This study has shown the viability of testing digital circuit modules which have resulted from a functional partitioning of the operations common to many digital subsystems. The study was conducted and the approaches evaluated using the present QED module partitioning. During the course of the study it became apparent that while the present module set is representative of the types of partitioning which can be done, it does not represent the optimum partitioning from a testability standpoint. As one example, it is desirable to include memory error correction in future RAM and ROM designs (see Section 4.1.4) in order to improve reliability and to aid in fault detection and isolation.

Rules for Built-In-Test - It is recommended that some of the results of this study be incorporated into a set of guidelines to be used by digital system designers. For example, the idea of using word parity to verify not only each input and output circuitry but also the logic card connectors and the inter-module (backplane) wiring should result in the definition of standard interfaces with built-in-test.

The BIT net gain study presented in this report served as an introduction to the problems and opportunities of subsystem and system level BIT. It is apparent from this study that the opportunity exists in many digital system architectures to effectively distribute built-in-test resources (both hardware and software) throughout all hierarchical levels. In particular, on modules where it is sufficient to monitor only a limited number of input and outputs and no internal data test points, it might be appropriate to use the subsystem monitor resources to verify proper module operation. By the same token, if only normal module I/O data are available, a very sophisticated subsystem or system monitor might be required, whereas access to module internal data could result in a greatly simplified monitor.

Extension of BIT Effectiveness Measures - The current module level BIT cost and effectiveness measures were discussed in Section 3.0. It was pointed out that these measures have their roots in conventional off-line test measures. Integral, concurrent fault monitoring at the module level requires additional consideration. As a limited first step,

a linearized model for on-line BIT was supported in Section 3.4.3. This model should be expanded and evaluated as the basis for a more appropriate on-line BIT effectiveness measure.

Simulation of Functional Modules With BIT - A gate level logic design of the recommended BIT circuits has been completed and the results are given in this report. It is recommended that these test circuits along with the circuits which are being tested be simulated in order to experimentally verify the performance of the BIT design.

Error Correction Techniques for Process Class Modules - In Section 4.1.4 it was shown that the reliability of the memory modules could be improved significantly through the use of error correction codes. It is recommended that error correction techniques for the non-linear Process Class modules be investigated and their ability to improve module Mean Time Between Failure (MTBF) be evaluated.

In conclusion, while these are but a few of the issues which should be investigated further, they are representative of the tasks which, if successfully completed, could lead to reduced modular digital systems life cycle cost and mean time to repair.

REFERENCES

1. N. L. Tinklepaugh and D. C. Eddington, "2175 Program: Quick and Easy Design (QED) of Systems Through High-Level Functional Modularity," NELC/TR 1904, Navy Electronics Laboratory Center, San Diego, California, January, 1974.
2. R. D. Alberts, J. B. Clary, J. W. Gault, S. J. Weidel, R. A. Whisnant, "A Study of a Standard BIT Circuit," Interim Technical Report, prepared for the Naval Avionics Facility, Indianapolis (NAFI) under contract N00163 76-C-0231, September, 1976.
3. D. C. Eddington, "Maintenance Concept for QED Modules", Navy Electronics Laboratory Center, San Diego, California, January 3, 1975.
4. D. Harrison, "Built-In-Test for Digital LSI Circuitry and Complex Digital Modules (Interim Report)" Navy Electronics Laboratory, San Diego, California, Division, Code 5200, 1976
5. W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection", Proceedings of the IRE, January, 1961.
6. W. W. Peterson, Jr. "Error Correcting Codes", MIT Press, Cambridge Massachusetts, 1961
7. N. S. Szabo and R. I. Tanaka, Residue Arithmetic and Its Applications to Computer Technology, McGraw-Hill, New York, NY, 1967.
8. C. V. Ramamoorthy and Y. W. Han, "Reliability Analysis of Systems with Concurrent Error Detection," IEEE Transactions on Computers, Vol. C-24, No. 9, September, 1975.
9. H. L. Garner, "The Residue Number System", IRE Transactions on Electronic Computers, June, 1959.
10. R. W. Watson and C. W. Hastings, "Self-Checked Computation Using Residue Arithmetic", Proceedings of the IEEE, Vol. 54, No. 12, December, 1966.
11. F. Barsi and P. Maestrini, "Error Detection and Correction by Product Codes in Residue Number Systems", IEEE Transactions on Computers, Vol. C-23, No. 9, September, 1974.
12. J. P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method", IBM Journal, July, 1966.
13. J. P. Roth, W. G. Bouricius and P. R. Schneider (1967), "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits", IEEE Transactions on Electronic Computers, Vol. EC-16, No. 5, October, 1967.

14. G. P. Putzolu and J. P. Routh, "A Heuristic Algorithm for the Testing of Asynchronous Circuits", IEEE Transactions on Computers, Vol. C-20, No. 6, June, 1971.
15. N. Benowitz, R. H. Bork, D. F. Calhoun, G. W. K. Lee and J. B. Shen, "Advanced Avionics Fault Isolation System (AAFIS) Application, Vol. I: AAFIS Application", Hughes Aircraft Company, Culver City, California, May, 1974.
16. R. Dandapani and S. M. Reddy, "On the Design of Logic Networks with Redundancy and Testability Considerations", IEEE Transactions on Computers, Vol. C-23, No. 11, November, 1974.
17. M. J. Y. Williams and J. B. Angell, "Enhancing Testability of Sarge-Logic", IEEE Transactions on Computers, Vol. C-22, No. 1, January, 1973.
18. A. Avizienis, "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design", IEEE Transactions on Computers, Vol. C-20, No. 11, November, 1971.
19. E. A. Toschi and T. Watanabe, "An All-Semiconductor Memory with Fault Detection, Correction, and Logging", Hewlett-Packard Journal, August, 1976.
20. McCluskey, E. J. Wakerly, J. F., Ogus, R. C., Current Research, Technical Report No. 100, Center for Reliable Computing, Stanford University, Stanford, California, October, 1975.
21. Wakerly, J. F., "Low Cost Error Detection Techniques for Small Computers", Technical Report No. 51, Digital Systems Laboratory, Stanford University, Stanford, California, December 1975.
22. W. D. Stanley, Digital Signal Processing, Reston Publishing Company, Reston, Virginia, 1975.
23. A. V. Oppenheim and R. W. Schaffer, Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.
24. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.
25. J. F. Poulson, Private Communication, NELC, San Diego, California, 1976.

APPENDIX A

RELIABILITY DATA FOR INTEGRATED CIRCUIT PACKAGES USED ON QED MODULES

This appendix contains the failure rate, equivalent number of gates, and typical power consumption data for the integrated circuits used in the QED Modules. The failure rate information is taken from "Built-In-Test for Digital LSI Circuitry and Complex Digital Modules (Interim Report)" by David Harrison, Jr., NELC code 5600. The number of gates was derived by solving the equations from MIL-HDBK-217B for the number of gates, given the failure rate. The equations are listed below:

For Monolithic Bipolar and MOS Digital SSI/MSI Devices

G is number of gates

$$C_1 = .00129 (G)^{0.67} \text{ or } G = \exp [(\ln C_1 - \ln .00129)/.67]$$

For Monolithic Bipolar and MOS Linear Devices

T is number of transistors

$$C_1 = .00056 (T)^{0.76} \text{ or } T = \exp [(\ln C_1 - \ln .00056)/.76]$$

For Monolithic Bipolar and MOS Digital LSI Devices

G is number of gates

$$C_1 = .0187 \exp (0.005G) \text{ or } G = (\ln C_1 - \ln .0187)/.005$$

For Monolithic Bipolar and MOS Memories

B is number of bits

For RAMs

$$C_1 = .00199B^{.603} \text{ or } B = \exp [\ln C_1 - \ln .00199]/.603]$$

For ROMs

$$C_1 = .00114B^{.603} \text{ or } B = \exp [\ln C_1 - \ln .00114]/.603]$$

For an integrated circuit that was not included in the above reference, the number of gates was determined from the data sheet and the failure rate calculated in a manner identical to the other circuits. Explicitly the equation is

$$\lambda_p = \pi_L \pi_Q (C_1 \pi_T + C_2 \pi_E) \quad (A.1)$$

AD-A056 147

RESEARCH TRIANGLE INST; RESEARCH TRIANGLE PARK NC SYST--ETC F/6 14/2
A STUDY OF A STANDARD BIT CIRCUIT.(U)

FEB 77 J B CLARY, J W GAULT, S J WEIKEL

N00163-76-C-0231

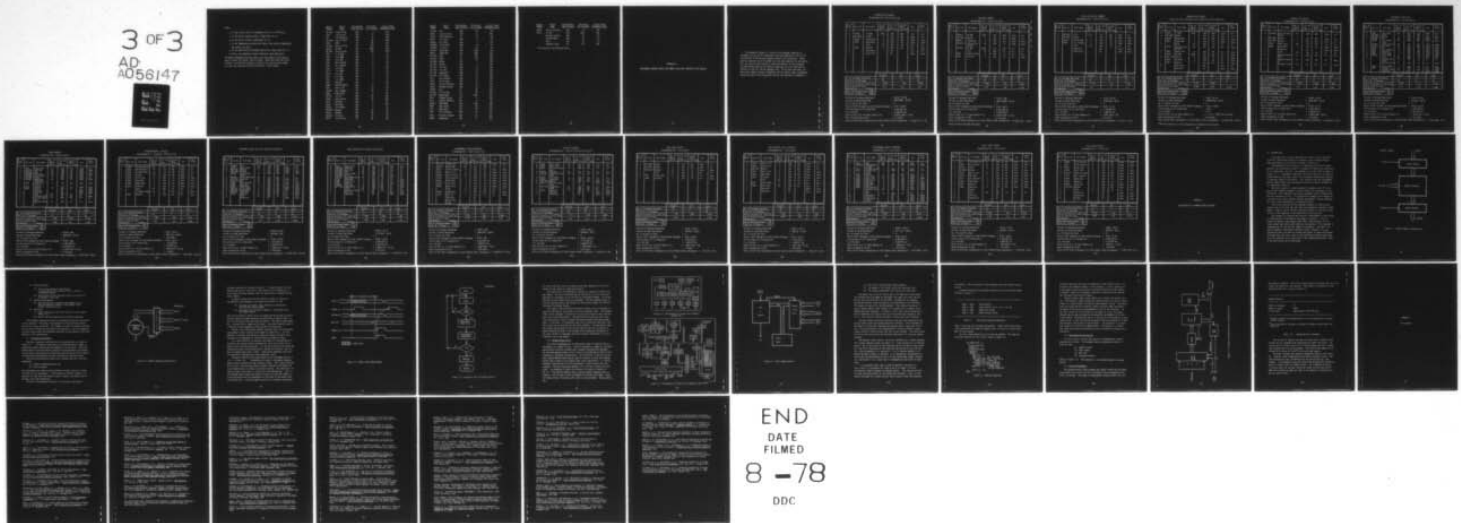
UNCLASSIFIED

RTI-43U-1269

NL

3 OF 3

AD
A056147



END

DATE

FILMED

8 -78

DDC

where:

λ_p is the failure rate of integrated circuit in F./10⁶ hours

π_L is the device learning factor, taken equal to 1.0

π_Q is the quality factor, taken equal to 5.0

π_T is the temperature acceleration factor, the junction temperature was taken to be 25°C

π_E is the application environment multiplier, taken equal to 1.0

C_1 and C_2 are complexity factors defined as described above.

The power consumption data was taken to be the product of the typical supply current and typical supply voltage. Where more than one supply voltage is involved the products of each of the supplies were summed.

This data was obtained from the manufacturer's data sheets.

<u>Device Number</u>	<u>Device Name</u>	<u>Failure Rate (per 10⁶ hrs)</u>	<u>Equivalent No. of Gates</u>	<u>Typical Power Consumption (mw)</u>
MH 0026	Clock Driver	.026	2	150
268A	Line Driver	.256	51	360*
272	Line Receiver	.256	51	105*
AM 2505	4x2 bit Mult.	.108	83	390*
2536	UART	.30	330	200
AM 2813	32x9 bit FIFO	.115	512	320
3604-6	4K ROM	.435	4096	460*
MM 5307	Prog. Divider	.30	330	300*
5400	2-in NAND	.034	4	60
5401	2-in NAND	.034	4	60
5402	2-in NOR	.034	4	70
5404	Hex Inverter	.039	6	90
5405	Hex Inverter	.039	6	90
5408	2-in AND	.034	4	100
5410	3-in NAND	.030	3	45
54LS11	3-in AND	.034	4	20
5427	3-in NOR	.030	3	55
5430	8-in NAND	.021	1	15
5432	Quad OR	.034	4	115
54S64	AND-OR-INVERT	.036	5	45
5474	Dual D F/F	.051	12	45
54LS85	Mag. Compare	.077	35	55
5486	Quad EX-OR	.034	4	150
54S113	J-K F/F	.066	25	150
54120	Pulse Driver	.060	20	255
54121	One-Shot	.030	3	115
54123	One-Shot	.039	6	230
54S133	13-in NAND	.021	1	30
54LS139	Decoder	.078	39	35
54154	Decoder	.071	28	170
54LS157	2-in Mux	.054	15	50
54175	D Flip-Flop	.071	28	150

<u>Device Number</u>	<u>Device Name</u>	<u>Failure Rate (per 10⁶ hrs)</u>	<u>Equivalent No. of Gates</u>	<u>Typical Power Consumption (mw)</u>
54LS181	ALU	.096	64	105
54182	Look Ah. Carry	.060	19	225
54198	Shift Register	.114	99	360
54279	R-S Latch	.044	8	90
54S260	5-in NOR	.026	2	100
54LS283	4-bit Adder	.082	36	110
54LS298	2-in Mux.	.077	35	65
54S370	512x4 ROM	.227	2048	525
54S371	256x8 ROM	.227	2048	525
MM 5750	RALU	.30	330	550
MM 5751	CROM	.30	330	600
DM 7095	Buffer	.041	7	325
DM 7097	Buffer	.041	7	325
DM 7121	8-in Mux.	.059	17	155
DM 7123	2-in Mux.	.054	15	200
DM 7130	Comparator	.048	10	240
DM 7223	8-in Demux.	.054	15	140
7417	Hex Inverter	.039	6	140
74173	4-bit Latch	.059	16	250
74180	Parity Generator	.054	14	170
74190	Up/down Counter	.084	47	325
74S381	ALU	.12	83	525
DM 7551	D Flip-Flop	.07	28	250
DM 7574	256x4 PROM	.178	1024	410
8T15	EIA/MIL Driver	.03	3	200
8T16	EIA/MIL Receiver	.03	3	90
82S123	32x8 PROM	.076	256	400
9308	Dual Latch	.071	28	310
9314	Quad Latch	.072	31	175
9316	4-bit Counter	.084	47	325
9318	Priority Encoder	.066	25	250
93S46	Comparator	.045	9	225

<u>Device Number</u>	<u>Device Name</u>	<u>Failure Rate (per 10⁶ hrs)</u>	<u>Equivalent No. of Gates</u>	<u>Typical Power Consumption (mw)</u>
93S62	Parity Generator	.060	20	225
93415	1K RAM	.308	1024	550
	Residue Checker	.084	47	60
	Residue Gener.	.082	41	60
	SIIB	.065	24	50
	Standard Timer	.32	114	150

* Data derived from QED Data Sheets.

APPENDIX B

PERFORMANCE MEASURE TABLES FOR MEMORY CLASS AND INTERFACE CLASS MODULES

This appendix contains a listing of the packages necessary to implement the various BIT approaches that were examined as well as the calculations of the analytic measures used in the evaluation. These analytic measures are not claimed to be the best measures for evaluating BIT, but they provide a guide to costs and benefits of each approach. The definitions used for the analytic measures are fully described in Section 3.0 and will not be repeated here. The failure rate data, the equivalent number of gates information and the typical power consumption values for each of the integrated circuits are tabulated in Appendix A.

RANDOM ACCESS MEMORY
Recommended BIT: Word Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Monitored
9	93415	1K RAM	1024	.308	550	1/9	All
4	DM 7095	Buffer	7	.041	325	None	All
1	93S46	Comparator	9	.045	225	None	None
1	5400	Quad Nand	4	.034	60	None	None
2		SIIB	24	.065	50	All	All
1	5432	Quad Or	4	.034	115	All	None
9		Resistors		.08		1/9	All
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	12	3	15
Total No. of Packages	14	4	18
Sum of Circuits F.R. (10 ⁻⁶ hr)	3.547	.552	4.099
Sum of Circuits Power Consumption (watts)	6.0	0.8	6.8
Sum of Monitored Gates = 9292			
Total No. of Gates = 9309			

Percent of Packages Monitored = $15/18 = 83.3\%$
Percent of Gates Monitored = $9292/9309 = 99.8\%$
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = $4/18 = 22.2\%$
Failure Rate (F.R.) without BIT = $3.547/10^6$ hrs
F.R. with BIT = $4.099/10^6$ hrs
Ratio of BIT F.R. to Total Module F.R. = $0.522/4.099 = 13.5\%$
Power Consumption of BIT = 0.8 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = $.765/6.75 = 11.3\%$

READ ONLY MEMORY

Recommended BIT: Word Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
4	3604-6	4K ROM	4096	.435	460*	None	A11
4	DM 7095	Buffer	7	.041	325	None	A11
1	93S46	Comparator	9	.045	225	None	None
1	54L5139	2-to-4 Line Decoder	39	.078	35	None	None
1	54S370	2K ROM	2048	.227	525	A11	A11
2		SIIB	24	.065	50	A11	A11
1	5454	And-Or-Invert	5	.036	45	A11	A11
1	5432	Quad OR	4	.034	115	A11	None

	Non-BIT	BIT	Total
No. of Packages Monitored	8	4	12
Total No. of Packages	10	5	15
Sum of Circuits F.R. (10 ⁻⁶ hr)	2.027	.427	2.454
Sum of Circuits Power Consumption (watts)	3.4*	.785	4.185
Sum of Monitored Gates = 18513			
Total No. of Gates = 18565			

Percent of Packages Monitored = 12/15 = 80%

Percent of Gates Monitored = 18513/18565 = 99.7%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 4/15 = 26.7%

Failure Rate (F.R.) without BIT = 2.207/10⁶ hrs

F.R. with BIT = 2.454/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = 0.427/2.454 = 17.4%

Power Consumption of BIT = 0.79 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = 0.785/4.185 = 18.8%

*Value derived from QED data sheet.

FIRST-IN FIRST-OUT MEMORY
Recommended BIT: Word Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
8	AM 2813	FIFO	512	.115	320	None	All
2	DM 7097	Buffer	7	.041	325	None	None
2	54LS11	Triple AND	3	.030	20	None	None
3	5404	Hex Inverter	6	.039	90	None	None
4		SIIB	24	.065	50	All	All
2		Resistors		.08		None	None
2		Capacitors		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	8	4	12
Total No. of Packages	15	4	19
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.739	.26	1.999
Sum of Circuits Power Consumption (watts)	3.5	.2	3.7
Sum of Monitored Gates = 4192			
Total No. of Gates = 4230			

Percent of Packages Monitored = 12/19 = 63.2%

Percent of Gates Monitored = 4192/4230 = 99.1%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 4/19 = 21.1%

Failure Rate (F.R.) without BIT = 1.739/10⁶ hrs

F.R. with BIT = 1.999/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = 0.26/1.999 = 13%

Power Consumption of BIT = 0.2 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = 0.2/3.7 = 5.4%

RANDOM ACCESS MEMORY

Single Bit Error Correction with Double Bit Error Detection

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
13	93415	1K RAM	1024	.308	550	5/13	A11
4	DM 7095	Buffer	7	.041	325	None	A11
1	93S46	Comparator	9	.045	225	None	None
1	5400	Quad Nand	4	.034	60	None	None
2		SIIB	24	.065	50	A11	A11
1	54154	4-to-16 Line Decoder	28	.071	170	A11	None
4	5486	Quad Ex-Or	4	.034	150	A11	None
1	54370	2K ROM	2048	.227	525	A11	None
2	54LS 157	Switch	15	.054	50	A11	None
1	5405	Inverter	6	.039	90	A11	None
1	5432	Quad Or	4	.034	115	A11	None
13		Resistors		.08		5/13	A11
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	12	7	19
Total No. of Packages	14	17	31
Sum of Circuits F.R. (10 ⁻⁶ hr)	3.547	2.685	
Sum of Circuits Power Consumption (watts)	5.985	4.45	10.435
Sum of Monitored Gates = 13388			
Total No. of Gates = 15533			

Percent of Packages Monitored = $19/31 = 61.3\%$
 Percent of Gates Monitored = $13388/15533 = 86.2\%$
 No. of Cycles for Test = 0
 Ratio of BIT Packages to Total Module Packages = $17/31 = 54.8\%$
 Failure Rate (F.R.) without BIT = 3.547
 F.R. with BIT = 3.111
 Ratio of BIT F.R. to Total Module F.R. = $1 - 3/1 = -200\%$ for one year
 Power Consumption of BIT = 4.45 watts
 Ratio of BIT Power Consumption to Total Module Power Consumption = $4.45/10.435 = 42.6\%$

*See text Section 4.1.4 for discussion of effective failure rate.

PARALLEL MULTIPLIER
Recommended BIT: Residue Coding

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
8	AM 2505	4x2 Bit Multiplier	83	.108	390	None	A11
2	DM 7095	Buffer	7	.041	325	None	A11
1	DM 7097	Buffer	7	.041	325	None	A11
3	9308	Dual Latch	28	.071	310	1/3	2/3
1	5400	Quad Nand	4	.034	60	None	None
1		Capacitor		.20		None	None
4		SIIB	24	.065	50	A11	A11
5		Residue Generator	41	.082	60	A11	A11
3		Residue Checker	47	.084	60	A11	None
2		Standard Timer	114	.32	150	A11	None

	Non-BIT	BIT	Total
No. of Packages Monitored	13	9	22
Total No. of Packages	14	15	29
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.363	1.633	2.996
Sum of Circuits Power Consumption (watts)	5.4	1.29	6.69
Sum of Monitored Gates = 1042			
Total No. of Gates = 1443			

Percent of Packages Monitored = $22/29 = 75.9\%$
Percent of Gates Monitored = $1042/1443 = 72.2\%$
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = $15/29 = 51.7\%$
Failure Rate (F.R.) without BIT = $1.363/10^6$ hrs
F.R. with BIT = $2.996/10^6$ hrs
Ratio of BIT F.R. to Total Module F.R. = $1.633/2.996 = 54.5\%$
Power Consumption of BIT = 1.29 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = $1.29/6.69 = 19.3\%$

ARITHMETIC LOGIC UNIT
Recommended BIT: Residue Coding

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
3	9308	Dual Latch	28	.071	310	None	2/3
3	DM 7097	Buffer	7	.041	325	1/3	All
1	DM 7095	Buffer	7	.041	325	None	None
2	74S381	ALU	83	.120	525	None	All
1	54182	Look Ahead Carry	19	.056	225	None	None
1	54198	Shift Register	99	.114	360	None	None
2	DM 7123	Quad Mux.	15	.054	200	None	None
2	54157	Mux	15	.054	50	None	None
3	5404	Hex Inverter	6	.039	90	None	None
1	5408	Quad AND	4	.034	100	None	None
2	5486	Quad EX-OR	4	.034	150	None	None
2	54S133	13-in NAND	1	.021	30	None	None
5		SIIB	24	.065	50	All	All
3		Residue Generator	41	.082	60	All	All
1		Residue Checker	47	.084	60	All	None
1	5410	Triple NAND	3	.030	45	All	None
2	5432	Quad OR	4	.034	115	All	None
1		Standrd. Timer	114	.32	150	All	None
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	7	9	16
Total No. of Packages	22	14	36
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.423	1.114	2.537
Sum of Circuits Power Consumption (watts)	4.77	1.125	5.895
Sum of Monitored Gates = 514			
Total No. of Gates = 903			

Percent of Packages Monitored = $16/36 = 44.4\%$
Percent of Gates Monitored = $514/903 = 56.9\%$
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = $14/36 = 38.9\%$
Failure Rate (F.R.) without BIT = $1.423/10^6$ hrs
F.R. with BIT = $2.537/10^6$ hrs
Ratio of BIT F.R. to Total Module F.R. = $1.114/2.537 = 43.9\%$
Power Consumption of BIT = 1.125 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = $1.125/5.895 = 19.1\%$

INDEX COUNTER

Recommended BIT: Residue Coding

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C.(mw)	BIT	Moni-tored
9	9314	Quad Latch	31	.072	175	None	8/9
6	54LS157	Quad Mux.	15	.054	50	None	None
2	54LS181	ALU	64	.096	105	None	A11
2	54LS298	Quad Mux.	35	.077	65	None	None
2	54LS283	4-bit Adder	36	.082	110	None	A11
3	DM 7097	Buffer	7	.041	325	None	A11
4	54LS85	Magnitude Comparator	35	.077	55	None	None
1	5486	Quad Exclu-sive-Or	4	.034	150	None	None
1	5402	Quad Nor	4	.034	70	None	None
3	5432	Quad Or	4	.034	115	2/3	None
1		Resistor		.08		None	None
1		Capacitor		.20		None	None
6		SIIB	24	.065	50	A11	A11
3		Standard Timer	114	.32	150	A11	None
6		Residue Gener-ator	41	.082	60	A11	A11
2		Residue Check-er	47	.084	60	A11	None

	Non-BIT	BIT	Total
No. of Packages Monitored	15	12	27
Total No. of Packages	31	19	50
Sum of Circuits F.R. (10 ⁻⁶ hr)	2.295	2.078	4.373
Sum of Circuits Power Consumption (watts)	3.965	1.46	5.425
Sum of Monitored Gates = 859			
Total No. of Gates = 1646			

Percent of Packages Monitored = 27/50 = 54%
 Percent of Gates Monitored = 859/1646 = 52.2%
 No. of Cycles for Test = 0
 Ratio of BIT Packages to Total Module Packages = 19/50 = 38%
 Failure Rate (F.R.) without BIT = 2.295/10⁶ hrs
 F.R. with BIT = 4.373/10⁶ hrs
 Ratio of BIT F.R. to Total Module F.R. = 2.078/4.363 = 47.5%
 Power Consumption of BIT = 1.46 watts
 Ratio of BIT Power Consumption to Total Module Power Consumption = 1.46/5.425 = 26.9%

MICROPROCESSOR (TYPICAL)

Recommended BIT: "Watchdog" Timer and SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
1	8080A	Microprocessor	650	1.32	780	None	All
2	8212	Buffer/Latch	70	.115	450	None	All
1	8228	Latch/Control	90	.125	700	None	All
1	8224	Clock	45	.084	720	None	All
2	5474	Flip-Flop	12	.051	45	None	None
1	5402	Quad NOR	4	.034	70	None	None
1	5400	Quad Nand	4	.034	60	None	None
1	5404	Inverter	6	.039	90	None	None
3		SIIB	24	.065	50	All	All
1		Standard Timer	114	.32	150	All	None
1	5432	Quad Or	4	.034	115	All	None

	Non-BIT	BIT	Total
No. of Packages Monitored	5	3	8
Total No. of Packages	10	5	15
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.968	.549	2.517
Sum of Circuits Power Consumption (watts)	3.41	.415	3.825
Sum of Monitored Gates = 997			
Total No. of Gates = 1153			

Percent of Packages Monitored

$$= 8/15 = 53.3\%$$

Percent of Gates Monitored

$$= 997/1153 = 86.5\%$$

No. of Cycles for Test

$$= 10 \text{ to } 8000$$

Ratio of BIT Packages to Total Module Packages

$$= 5/15 = 33.3\%$$

Failure Rate (F.R.) without BIT

$$= 1.97/10^6 \text{ hrs}$$

F.R. with BIT

$$= 2.52/10^6 \text{ hrs}$$

Ratio of BIT F.R. to Total Module F.R.

$$= .549/2.547 = 21.8\%$$

Power Consumption of BIT

$$= .4 \text{ watts}$$

Ratio of BIT Power Consumption to Total Module Power Consumption = .415/3.825 = 10.8%

ARITHMETIC LOGIC UNIT WITH PARTIAL DUPLICATION

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
4	74S381	ALU	83	.120	525	1/2	A11
3	9308	Dual Latch	28	.071	310	None	2/3
2	DM 7097	Buffer	7	.041	325	None	A11
1	DM 7095	Buffer	7	.041	325	None	None
2	54182	Look Ahead Carry	19	.056	225	1/2	A11
2	54198	Shift Register	99	.114	360	1/2	A11
2	DM 7123	Quad Mux	15	.054	200	None	A11
4	54LS157	Mux	15	.054	50	1/2	A11
3	5404	Hex Inverter	6	.039	90	None	None
2	5408	Quad And	4	.034	100	1/2	None
2	5486	Quad Ex-Or	4	.034	150	None	None
2	54S133	13-in NAND	1	.021	30	None	None
2	DM 7130	Comparators	10	.048	240	A11	None
1	54S260	Dual NOR	2	.026	110	A11	None
1	5402	Quad OR	4	.034	70	A11	None
6		SIIB	24	.065	50	A11	A11
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	12	12	24
Total No. of Packages	22	17	39
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.423	1.098	2.521
Sum of Circuits Power Consumption (watts)	4.77	2.785	7.555
Sum of Monitored Gates = 872			
Total No. of Gates = 969			

Percent of Packages Monitored = 24/39 = 61.5%
 Percent of Gates Monitored = 872/969 = 90%
 No. of Cycles for Test = 0
 Ratio of BIT Packages to Total Module Packages = 17/39 = 43.6%
 Failure Rate (F.R.) without BIT = 1.423/10⁶ hrs
 F.R. with BIT = 2.521/10⁶ hrs
 Ratio of BIT F.R. to Total Module F.R. = 1.098/2.521 = 43.6%
 Power Consumption of BIT = 2.785 watts
 Ratio of BIT Power Consumption to Total Module Power Consumption = 2.785/7.555 = 36.97%

INDEX COUNTER WITH PARTIAL DUPLICATION

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
9	9314	Quad Latch	31	.072	175	None	8/9
12	54LS157	Quad Mux	15	.054	50	1/2	A11
4	54LS298	Quad Max	35	.077	65	1/2	A11
4	54LS181	ALU	64	.096	105	1/2	A11
4	54LS283	4-bit Adder	36	.082	110	1/2	A11
3	DM 7097	Buffer	7	.041	325	None	A11
8	54LS85	Mag. Comparator	35	.077	55	1/2	A11
2	5402	Quad NOR	4	.034	70	1/2	None
2	5486	Quad Ex-Or	4	.034	150	1/2	None
1	5432	Quad Or	4	.034	115	None	None
2	54S260	Dual NOR	2	.026	100	A11	None
2	DM 7130	Comparators	10	.048	240	A11	None
6		SIIB	24	.065	50	A11	A11
1		Standard Timer	114	.32	150	A11	None
1		Resistor		.08		None	None
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	27	22	49
Total No. of Packages	31	29	60
Sum of Circuits F.R. (10 ⁻⁶ hr)	2.295	2.068	4.363
Sum of Circuits Power Consumption (watts)	3.965	2.430	6.395
Sum of Monitored Gates = 1413			
Total No. of Gates = 1602			

Percent of Packages Monitored = 49/60 = 81.7%
Percent of Gates Monitored = 1413/1602 = 88.2%
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = 29/60 = 48.3%
Failure Rate (F.R.) without BIT = 2.295/10⁶ hrs
F.R. with BIT = 4.363/10⁶ hrs
Ratio of BIT F.R. to Total Module F.R. = 2.068/4.363 = 47.4%
Power Consumption of BIT = 2.43 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = 2.43/6.395 = 40%

PROGRAMMABLE TIMING GENERATOR
Recommended BIT: Work Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
9	DM 7551	Quad D F/F	28	.07	250	None	8/9
6	82S123	32x8 PROM	256	.076	400	1/3	All
3	9316	4-bit Counter	47	.084	325	None	None
5	5474	Dual D F/F	12	.051	45	None	None
1	54120	Pulse Driver	20	.060	255	None	None
2	5404	Hex Inverter	6	.039	90	None	None
1	5400	Quad Nand	4	.034	60	None	None
1	54279	R-S Latch	8	.044	90	None	None
1	5486	Quad Ex-Or	4	.034	150	None	None
4		SIIB	24	.065	50	All	All
1	93S46	Comparator	9	.045	225	All	None
2	5432	Quad OR	4	.034	115	All	None
5		Resistors		.08			
4		Capacitors		.20			
1		Diode		.04			

	Non-BIT	BIT	Total
No. of Packages Monitored	12	6	18
Total No. of Packages	27	9	36
Sum of Circuits F.R. (10 ⁻⁶ hr)	2.931	.525	3.456
Sum of Circuits Power Consumption (watts)	5.785	1.055	6.84
Sum of Monitored Gates = 1856			
Total No. of Gates = 2150			

Percent of Packages Monitored = $18/36 = 50\%$
Percent of Gates Monitored = $1856/2150 = 86.3\%$
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = $9/36 = 25\%$
Failure Rate (F.R.) without BIT = $2.931/10^6$ hrs
F.R. with BIT = $3.456/10^6$ hrs
Ratio of BIT F.R. to Total Module F.R. = $.525/3.456 = 15.2\%$
Power Consumption of BIT = 1.055 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = $1.055/6.84 = 15.4\%$

PRIORITY ENCODER

Recommended BIT: Partial Duplication and SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
4	9308	Dual Latch	28	.071	310	1/2	A11
16	5474	Dual D F/F	12	.051	45	1/2	A11
4	9318	Priority Enco-der	25	.066	250	1/2	A11
2	54154	Decoder	28	.071	170	1/2	A11
2	54LS85	Magnitude Com- pare	35	.077	55	1/2	A11
8	54279	R-S Latch	8	.044	90	1/2	A11
4	5400	Quad Nand	4	.034	60	1/4	1/2
3	DM 7097	Buffer	7	.041	325	None	2/3
2	89B-1-1K	Resistor Net- work		.02		None	None
1	DM 7130	Comparator	10	.048	240	A11	None
1		SIIB	24	.065	50	A11	A11
1	5432	Quad OR	4	.034	115	A11	None
3		Resistor		.08		None	None
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	21	20	41
Total No. of Packages	24	22	46
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.711	1.187	2.898
Sum of Circuits Power Consumption (watts)	3.22	2.53	5.75
Sum of Monitored Gates = 640			
Total No. of Gates = 669			

Percent of Packages Monitored = 41/46 = 89.1%

Percent of Gates Monitored = 640/669 = 95.7%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 22/46 = 47.8%

Failure Rate (F.R.) without BIT = 1.711/10⁶ hrs

F.R. with BIT = 2.898/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = 1.187/2.898 = 41%

Power Consumption of BIT = 2.53 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = 2.53/5.75 = 44%

DUAL 8-BIT SWITCH
Recommended BIT: Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
8	DM 7095	Buffer	7	.041	325	None	All
8	DM 7097	Buffer	7	.041	325	None	All
1	54LS139	Decoder	39	.078	35	None	None
6		SIIB	24	.065	50	All	All
2	5427	Triple NOR	3	.030	55	All	None
1	5404	Inverter	6	.039	90	All	None

	Non-BIT	BIT	Total
No. of Packages Monitored	16	6	22
Total No. of Packages	17	9	26
Sum of Circuits F.R. (10 ⁻⁶ hr)	.734	.489	1.223
Sum of Circuits Power Consumption (watts)	5.234	.5	5.734
Sum of Monitored Gates = 256			
Total No. of Gates = 307			

Percent of Packages Monitored = 22/26 = 84.6%

Percent of Gates Monitored = 256/307 = 83.4%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 9/26 = 34.6%

Failure Rate (F.R.) without BIT = .734/10⁶ hrs

F.R. with BIT = 1.223/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = .489/1.223 = 40%

Power Consumption of BIT = .5 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = .5/5.734 = 8.7%

DUAL PARALLEL 8-BIT INTERFACE

Recommended BIT: Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
8	54S113	Dual JK F/F	25	.066	150	None	All
4	5486	Quad Ex-Or	4	.034	150	None	All
2	DM 7097	Buffer	7	.041	325	None	All
3	5474	Dual D F/F	12	.051	45	None	None
2	5408	Quad Nand	4	.034	100	None	None
1	5404	Inverter	6	.039	90	None	None
4		SIIB	24	.065	50	All	All
1	5427	Triple NOR	3	.030	55	All	None
2	9308	Dual Latch	28	.096	310	None	All
2		Resistors		.08		None	None
1		Capacitor		.20		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	16	4	20
Total No. of Packages	22	5	27
Sum of Circuits F.R. (10 ⁻⁶ hr)	1.478	.29	1.768
Sum of Circuits Power Consumption (watts)	3.495	.255	3.75
Sum of Monitored Gates = 382			
Total No. of Gates = 435			

Percent of Packages Monitored = 20/27 = 74.1%

Percent of Gates Monitored = 382/435 = 87.8%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 5/27 = 18.5%

Failure Rate (F.R.) without BIT = 1.478/10⁶ hrs

F.R. with BIT = 1.768/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = .29/1.768 = 16.4%

Power Consumption of BIT = .255 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = .255/3.75 = 6.8%

ASYNCHRONOUS SERIAL INTERFACE

Recommended BIT: Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
1	2536	UART	330	.30	200	None	All
1	MM 5307	Programmable Divider	330	.30	300	None	None
2	8T15	EIA/MIL Driver	3	.030	200	None	1/2
2	8T16	EIA/MIL Receiver	3	.030	90	None	1/2
4	7097	Buffer	7	.041	325	None	1/2
5	9314	Quad Latch	31	.072	175	None	2/5
1	5474	Dual D F/F	12	.051	45	None	None
1	5486	Quad Ex-Or	4	.034	150	None	All
1	5408	Quad And	4	.034	100	None	None
1	5400	Quad Nand	4	.034	60	None	None
1	5432	Quad Or	4	.034	115	All	None
1	5404	Hex Inverter	6	.039	90	None	None
5		Standard Tim-ers	114	.32	150	None	None
2		SIIB	24	.065	50	All	All
6		Capacitors		.20		None	None
12		Resistors		.08		None	None
3		Transistors		.41		None	None
5		Diodes		.04		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	8	2	10
Total No. of Packages	25	3	28
Sum of Circuits F.R. (10 ⁻⁶ hr)	6.626	.164	6.79
Sum of Circuits Power Consumption (watts)	4.45	.215	4.665
Sum of Monitored Gates = 464			
Total No. of Gates = 1507			

Percent of Packages Monitored

$$= 10/28 = 35.7\%$$

Percent of Gates Monitored

$$= 464/1507 = 30.8\%$$

No. of Cycles for Test

$$= 0$$

Ratio of BIT Packages to Total Module Packages

$$= 3/28 = 10.7\%$$

Failure Rate (F.R.) without BIT

$$= 6.626/10^6 \text{ hrs}$$

F.R. with BIT

$$= 6.79/10^6 \text{ hrs}$$

Ratio of BIT F.R. to Total Module F.R.

$$= .164/6.79 = 2.4\%$$

Power Consumption of BIT

$$= .215 \text{ watts}$$

Ratio of BIT Power Consumption to Total Module Power Consumption = .215/4.665 = 4.6%

NTDS INPUT BUFFER
Recommended BIT: Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
6	272	Line Receiver	51	.256	105	None	None
1	268A	Line Driver	51	.256	360	None	None
8	74173	Latch	16	.059	250	None	All
1	DM 7097	Buffer	7	.041	325	None	None
3	7474	Dual D F/F	12	.051	45	None	None
5	74121	One Shot	3	.030	115	None	None
1	7417	Hex Inverter	6	.039	140	None	None
1	7400	Quad Nand	4	.034	60	None	None
1	7432	Quad Or	4	.034	115	All	None
1	7408	Quad And	4	.034	100	None	None
3		Diodes		.04			
4		SIIB	24	.065	50	All	All
12		Resistors		.08			
8		Capacitors		.20			

	Non-BIT	BIT	Total
No. of Packages Monitored	8	4	12
Total No. of Packages	27	5	32
Sum of Circuits F.R. (10 ⁻⁶ hr)	5.395	.294	5.689
Sum of Circuits Power Consumption (watts)	4.325	.315	4.64
Sum of Monitored Gates = 224			
Total No. of Gates = 657			

Percent of Packages Monitored = $12/32 = 37.5\%$
Percent of Gates Monitored = $224/657 = 34.1\%$
No. of Cycles for Test = 0
Ratio of BIT Packages to Total Module Packages = $5/32 = 15.6\%$
Failure Rate (F.R.) without BIT = $5.395/10^6$ hrs
F.R. with BIT = $5.689/10^6$ hrs
Ratio of BIT F.R. to Total Module F.R. = $.294/5.689 = 5.2\%$
Power Consumption of BIT = .315 watts
Ratio of BIT Power Consumption to Total Module Power Consumption = $.315/4.64 = 6.8\%$

NTDS OUTPUT BUFFER
Recommended BIT: Parity/SIIB

No. of Parts	Part No.	Part Name	No. of Gates per IC	F.R. per I.C. (/10 ⁶ hr)	Typical Power /I.C. (mw)	BIT	Moni-tored
6	268A	Line Drivers	51	.256	360	None	None
1	272	Line Receiver	51	.256	105	None	None
8	9314	Quad Latch	31	.072	175	None	All
2	7474	Dual D F/F	12	.051	45	None	None
2	74123	Dual One Shots	6	.039	230	None	None
1	7417	Hex Inverter	6	.039	140	None	None
1	7400	Quad Nand	4	.034	60	None	None
1	7432	Quad Or	4	.034	115	All	None
1	7408	Quad And	4	.034	100	None	None
1	DM 7097	Buffer	7	.041	325	None	None
4		SIIB	24	.065	50	All	All
9		Resistors		.08		None	None
5		Capacitors		.20		None	None
10		Diodes		.04		None	None

	Non-BIT	BIT	Total
No. of Packages Monitored	8	4	12
Total No. of Packages	23	5	28
Sum of Circuits F.R. (10 ⁻⁶ hr)	4.816	.294	5.11
Sum of Circuits Power Consumption (watts)	4.84	.315	5.155
Sum of Monitored Gates = 344			
Total No. of Gates = 762			

Percent of Packages Monitored = 12/28 = 42.9%

Percent of Gates Monitored = 344/762 = 45.1%

No. of Cycles for Test = 0

Ratio of BIT Packages to Total Module Packages = 5/28 = 17.9%

Failure Rate (F.R.) without BIT = 4.816/10⁶ hrs

F.R. with BIT = 5.110/10⁶ hrs

Ratio of BIT F.R. to Total Module F.R. = .294/5.11 = 5.8%

Power Consumption of BIT = .315 watts

Ratio of BIT Power Consumption to Total Module Power Consumption = .315/5.155 = 6.1%

Appendix C

DISCUSSION OF A STANDARD SAMPLE MONITOR

C.0 INTRODUCTION

The standard BIT circuits described in Section 4.0 were designed to monitor subfunctions which are frequently used in module designs. These approaches are to the left of center in Figure 2.7.

The objective of this section is to describe a BIT circuit whose behavior can be modified so that it is capable of monitoring a wide variety of operational circuits. This approach is to the right of center in Figure 2.7. The general applicability of this device is provided by program mobility and hence, there is an attendant overhead and reduction in operating speed of the monitor. This requires that the programmable monitor sample rather than continuously verify the results produced by the operational circuit.

The basic intent of a sampled approach to module level BIT is to balance the tradeoffs between a single standard and continuous on-line monitoring. This identifies the primary parameter which differentiates the two intentions as time. If an operational module can be sampled at a suitable rate, then a general purpose, programmatic (single standard) BIT circuit can be defined for a functional family.

The initial reaction to this approach probably is instinctively negative due to the fear that the resulting diagnostic coverage is inadequate. Clearly, one of the most important aspects of a serious consideration of this approach is the definition of cost and gain.

The concept of a sampled verification of performance is best suited to those modules of the general form shown in Figure C.1. This includes the process class modules (ALU, MULT, Index counter) and to a lesser extent the control and interface class modules. This approach is inappropriate for use with the memory class modules. The idea is to take a snapshot of the module interface values, (Figure C.1) at an appropriate time, and then to validate the output to input relationship in slower than real time. Most of the significant problems associated with this approach can be identified in the statement made above. Some of the most obvious are listed below.

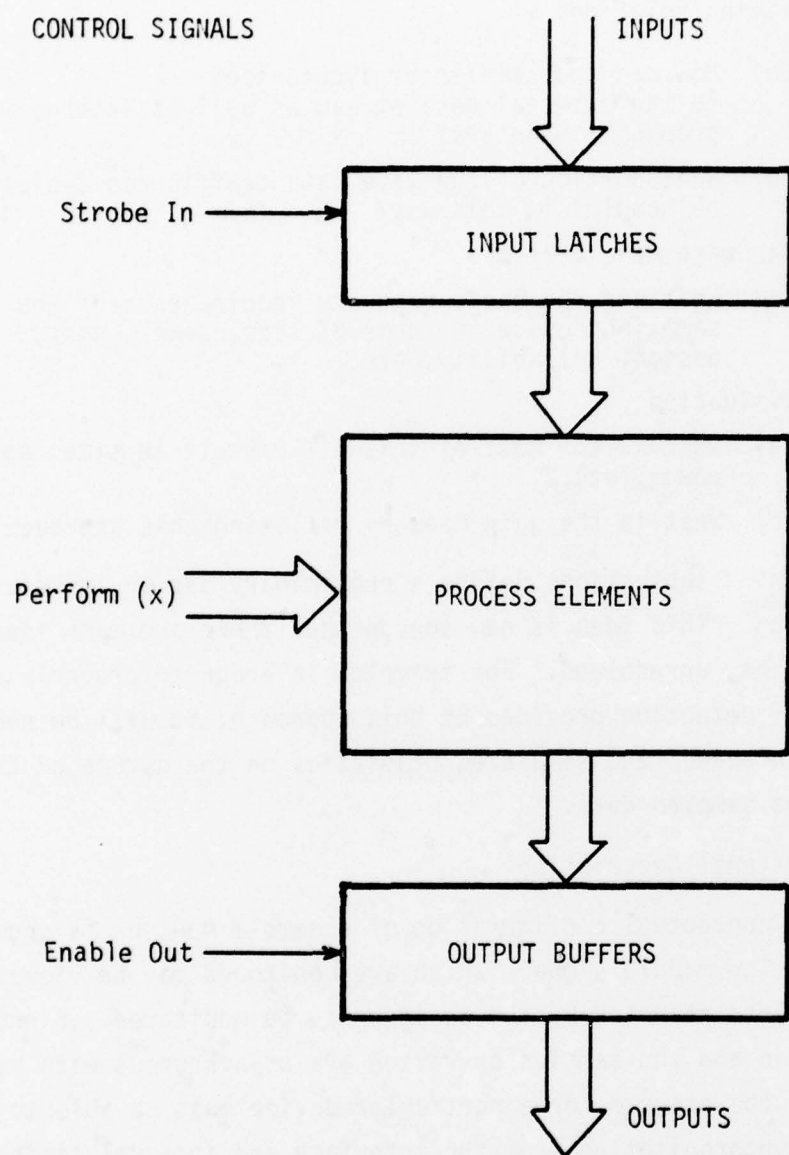


Figure C.1 General Module Configuration

- (1) Timing relations
 - (a) How can the samples be synchronized to the external data stream as well as internal propagation delays?
 - (b) What portion of the live data traffic can typically be handled in this way?
- (2) Hardware Requirements
 - (a) What are the basic hardware requirements of the sampling device in terms of size, power, cost, design, reliability, etc.
- (3) Evaluation
 - (a) What is the cost of this BIT circuit in size, space, power, etc.?
 - (b) What is the gain made by utilizing this approach?

The following subsections define a preliminary design and evaluation of this approach. This idea is new enough that there are many issues which are, as yet, unresolved. For example, in order to properly evaluate the per cent of detection provided by this approach, it will be necessary to collect both actual and simulated statistics on the nature of the fault behavior in the sampled data.

C.1 Functional Description

The basic conceptual configuration of a sample monitor is shown in Figure C.2. The module signals which are monitored may be viewed as system test points selected by the designer to be monitored. Since the module operation and the samples operation are asynchronous with respect to each other, the program for a particular device must be able to derive the required synchronization from the interface and internal timing.

The timing considerations which are of concern here fall into two categories:

- (1) Sampling synchronization, and
- (2) Cycle coverage.

The requirements for sample synchronization are quite similar to those for other on-line approaches. In the process of taking a sample, it is essential to know that the outputs, control signals, and inputs are all related to the same computation.

The most straightforward solution is to constrain the system

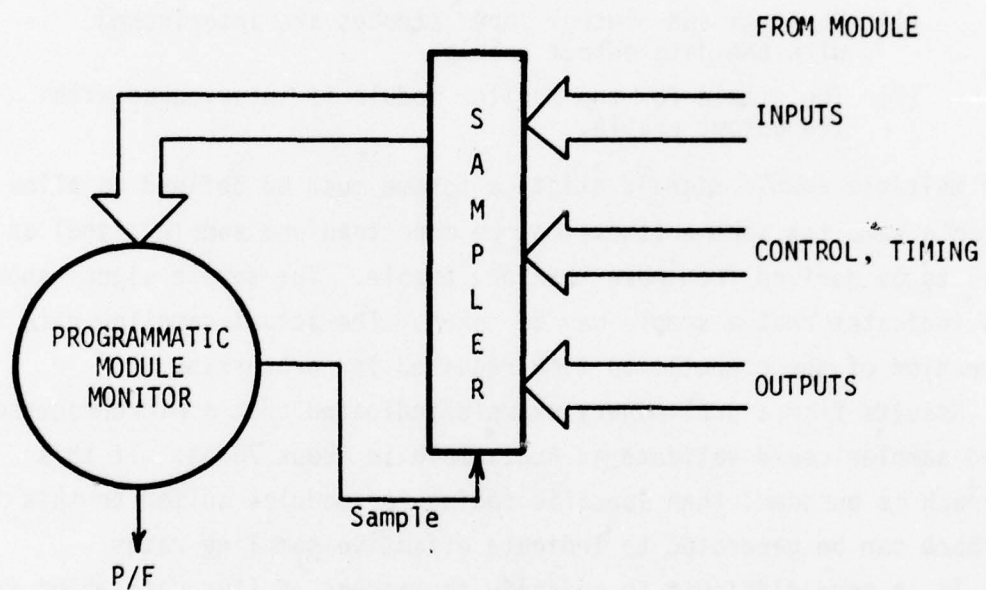


Figure C.2 General Sampling Configuration

designers approach to timing and control. If output buffers are only enabled when data is valid and if the inputs are not changed until a valid output has been enabled, then the sample may be triggered by the output enable.

A typical timing chain for the signals is shown in Figure C.3. The important relationships to be noted from Figure C.3 are:

- (1) The data and control input strobes are interlocked with the data output enable,
- (2) The sample for the monitor module is interlocked with the output enable.

When multiple enable signals exist, a scheme must be defined to allow a suitable sampling scheme (there may be more than one sample signal as well) to be derived from more than one enable. The sample signal shown only indicates that a sample may be taken. The actual sampling rate is a function of the computation time required for processing.

Results from a preliminary example indicated that a microprocessor based sampler could validate an ALU sample in about 70 μ s. If this approach is pursued, then specific coding for modules suited to this approach can be generated to indicate effective sampling rates.

It is more difficult to quantify the amount of live data which is monitored (the second timing consideration) since this figure varies greatly with time and application. For the ALU estimate given above and assuming the worst case (minimum delay, maximum utilization), the live data to sampled data ratio is 600:1. Application dependent statistics are required to substantiate a more meaningful ratio.

The basic functional flow description of the programmed monitor is shown in Figure C.4. The monitor will begin a new sample sequence based on the occurrence of some interface signal. This signal typically will be one of the existing control signals received by the module to latch incoming data or select a particular process. The sampling of interface signals must occur in an explicit, well-defined way. That is, the designer must program the sample routine to collect the samples in a way consistent with the presentation of the data to the module from the outside world. Once the sampling activity is complete the program

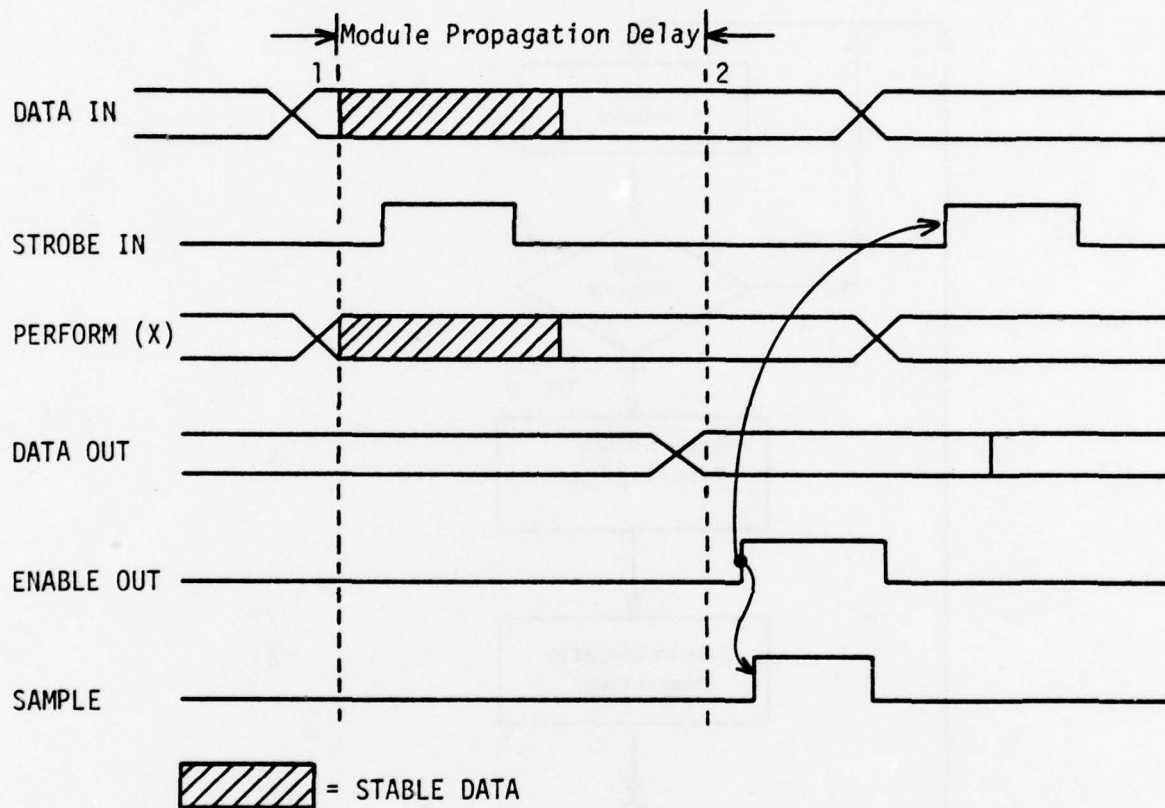


Figure C.3 Sample Timing Requirements

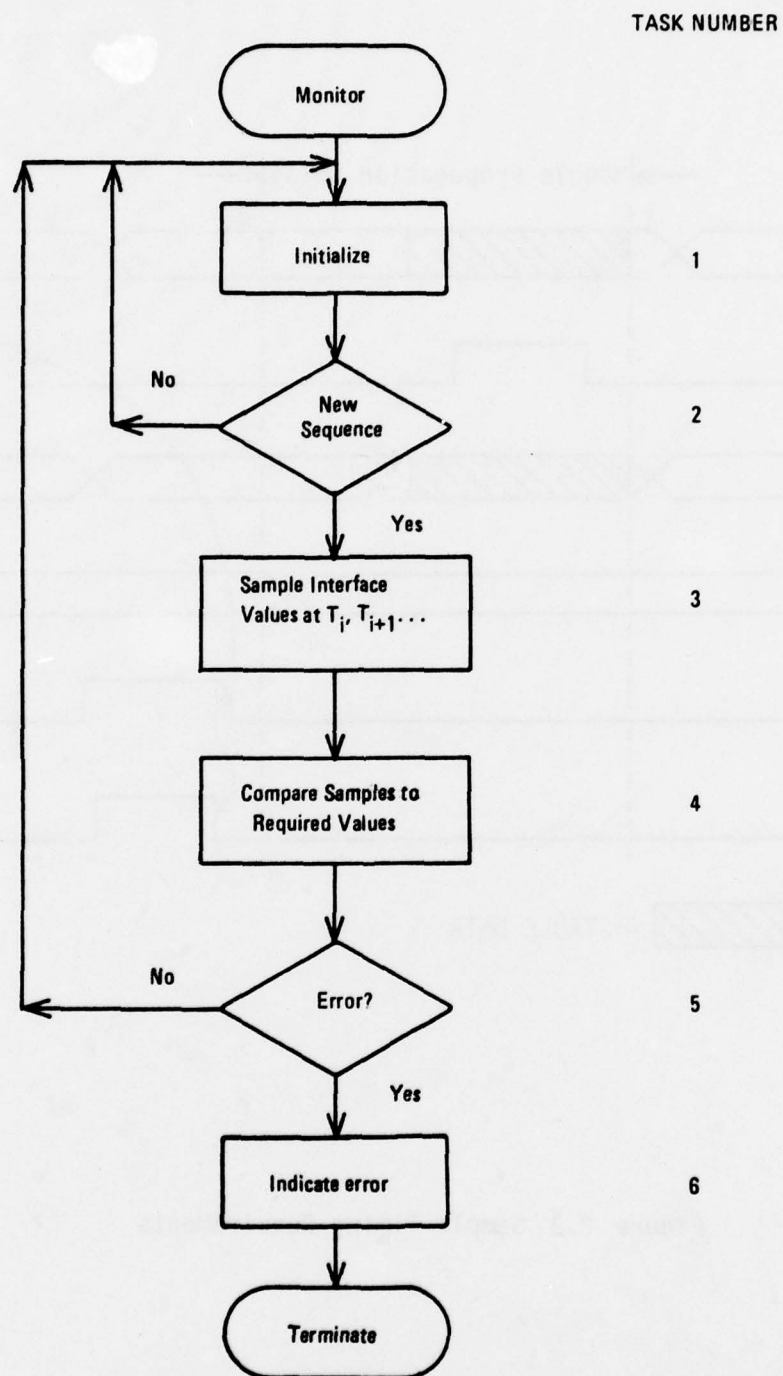


Figure C.4 Functional Flow of Program Monitor

can verify that the values of outputs which were sampled are valid with respect to the values of inputs and controls.

The results will be indicated by the use of a pass/fail output from the sampler. An error indication may be the result of (1) an incorrect operation in the module or (2) an error by the sample element. In either case the failure status should be presented. The most damaging types of failures possible within the sampler are those which prevent the failure indication from being issued.

This type of failure within the monitor module can be eliminated by requiring the monitor to actively utilize an external timer. The standard timer circuit proposed in Section 4.2.4 can be utilized for this function with very little modification. The monitor must reload the timer count value sufficiently often to prevent the count complete from being generated. In this way, if the monitor even fails to activate the count, then the failure can be indicated by the count complete. The possibility that the sampler could properly utilize the timer and yet fail to detect or report bonafide failures in the system can be reasonably discounted if the timer control is dependent upon the successful completion of a self-diagnostic run by the sampler.

C.2 Hardware Description

The actual implementation of the monitor shown in Figure C.2 may be approached in a number of ways. There are presently a number of available devices which could be selected and a standard configuration designed. The microprocesssing module, which is part of the QED family might be considered as a candidate configuration. The difficulty is that one of the most complex modules would now be used as a monitor on much less complex modules which create a cost greater than simple duplication. If a microprocessor configuration is to be considered, it must require very few packages. The most attractive approach is to utilize a single chip computer. Two examples of single chip devices are shown in Figures C.5 and C.6. The required memory, ports, CPU and support elements shown in Figure C.7 can be seen in both the MOSTEK F-8 and the Intel 8048. There are three primary limitations which need to be considered. These concerns are:

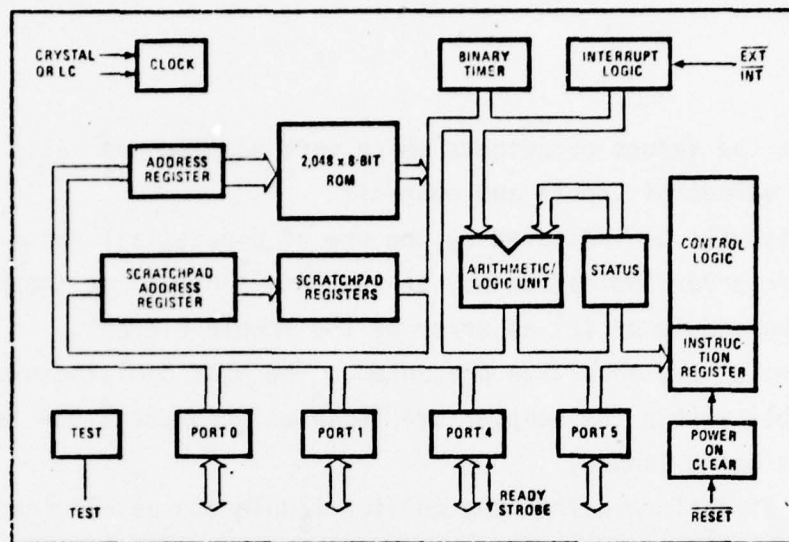


Figure C.5 Architecture of Single Chip Processor (Mostek F8)

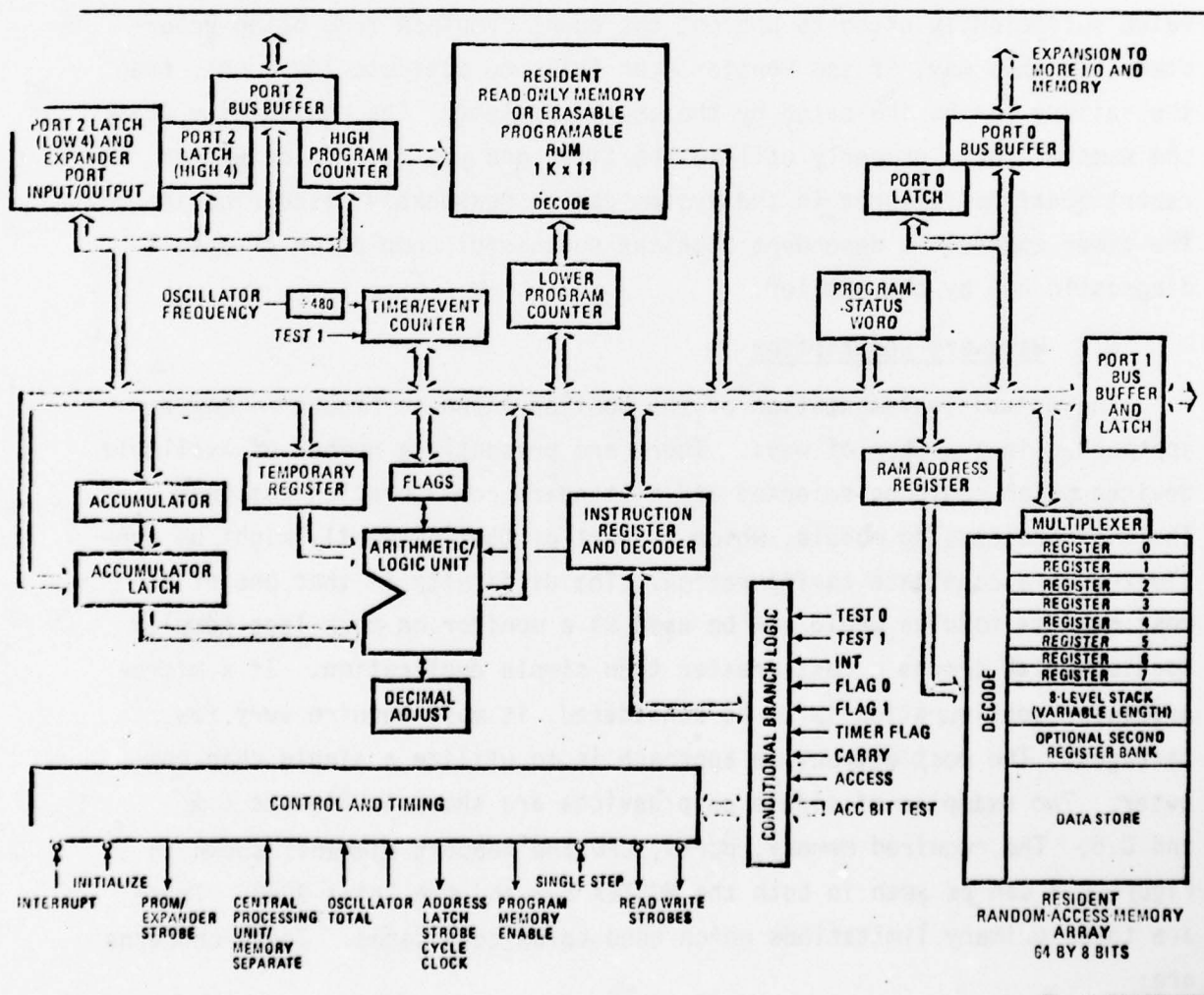


Figure C.6 Architecture of Single Chip Processor (Intel 8048)

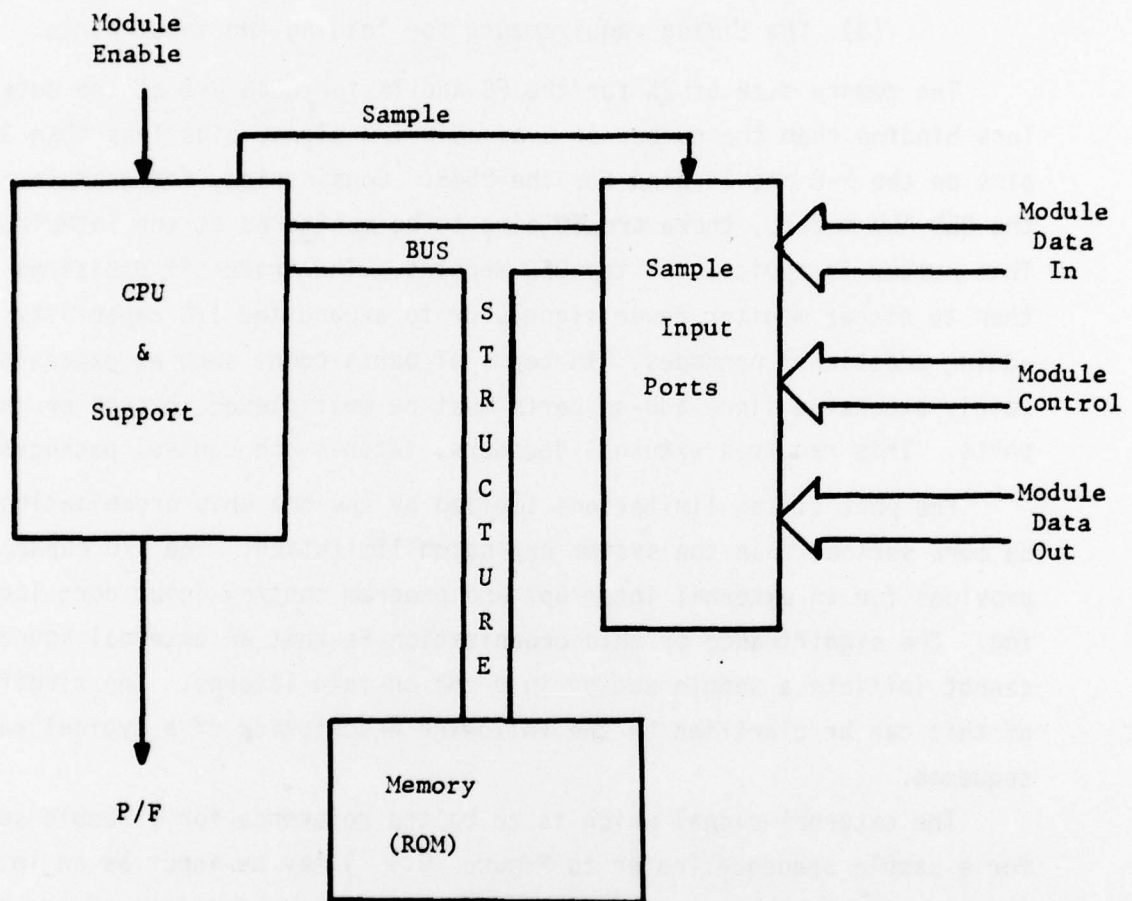


Figure C.7 Basic Sample Monitor

- (1) The size of the available program memory,
- (2) The number of I/O signals which are available, and
- (3) The timing requirements for loading the input ports.

The memory size of 2K for the F8 and 1K for 8048 are at the outset less binding than the number of available I/O signal pins less than 32 pins on the F-8 and 24 pins for the 8048. Considering, for example, the QED ALU module, there are 80 pins to be monitored at the interface. This number is typical for the QED modules. The trade-off decision is then to either monitor fewer signals or to expand the I/O capability by adding additional packages. In terms of parts count such as expansion is fairly expensive since add-on parts must be multiplexed through on chip ports. This requires external decoders, latches and control packages.

The port timing limitations implied by the one chip organization may be more serious than the system expansion limitation. The I/O capability provides for an external interrupt and program control input port latching. The significance of this organization is that an external source cannot initiate a sample action into the on chip latches. The significance of this can be clarified by the following description of a typical sample sequence.

The external signal which is to be the reference for a sample sequence for a sample sequence (refer to Figure C.4) may be input as an interrupt source. Due to the nature of the interrupt service function of the processor, the first instruction of the interrupt service may not be initiated for a few microseconds (1-3 typically). It will then be another 10 μ s before the entire sample is complete. In an operational system which requires that the QED modules be run at or near operational speed (100-200 μ s), the monitor's 10 μ s input rate is inadequate by two orders of magnitude.

It is essential then, that in order to monitor live data at a 100 μ s rate it is necessary for tasks 2 and 3 of Figure C.4 to be initiated by signals external to program control. This implies the need for data latching external to the single chip processor. Thus, in turn implies the need for a sample control which properly times and sequences

the samples. This controller is also separate from the single chip processor.

Consider monitoring the straight forward activity of the ALU module defined in Table C.1.

Step 1	/BLE	latch input B
Step 2	/CLE	latch control for B - Acc → Acc
Step 3	/ROE	enable Acc output
Step 4	/SOE	enable status output

Table C.1. ALU Module Operating Reference

Steps 1 and 2 may be initiated concurrently. Steps 3 and 4 may be executed concurrently and must be delayed from 1, 2 by 20 μ s if only valid data is to be bussed.

For this simple operation 23 bits must be sampled. The sampling algorithm required for this task is shown in Table C.2.

```

Do Sample ALU: = (
  Do while BLE END,
    Sample B,
  Do while CLE END,
    Sample C,
  Decode C (
    Case 0:=( ),
    Case 1:=(      ! B - Acc case
      (Do while ROE, END, ! The 120  $\mu$ s
        Sample R,); ! delay is
      (Do while SOE END, ! a design
        Sample S,)), ! responsibility
    Case 2:=( )      ! other cases
                ! of ALU operation
    Case 7:=( )),
END.
```

Table C.2. Monitor Algorithm

The table indicates the level of complexity of the control which is required of the hardware external to the processor. In this example all of the sample events are directly initiated by external system signals. In general, there will be samples required at times which are some delay time away from explicit signals.

Figure C.8 show a basic block diagram of a single chip monitor system. The necessary sample latches are strobed either by external control signals or a delay signal initiated by some external control. The delay timer may accept multiple external signals and delay them individually. The amount of delay may be programmatically set by the processor. In this way, within the restrictions on the number of lines, the sample software may be used to establish the sampling requirements of an individual module. Multiplexing control is provided by the processor so that samples taken in real time may be accessed by the test program at a lower rate through a narrow port on the processor. The processor is a single chip process or of the F-8 or 8048 variety. The monitor is a "watch dog" timer which must be periodically reset to prevent the process fail indication.

C.3 Implementation Description

This BIT circuit involves two aspects of implementation, namely hardware and software. The hardware design implementation detail is given for the:

- (1) Sample latches
- (2) Delay timer,
- (3) Mux, and
- (4) Processor monitor

shown in Figure C.8. The processor is a purchased element discussed earlier.

C.4 Critical Parameters

The synchronization timing between the sample latches and the operating signals are critical but can be handled by the programmable delay which is provided. The number of programmably delayed signals may vary

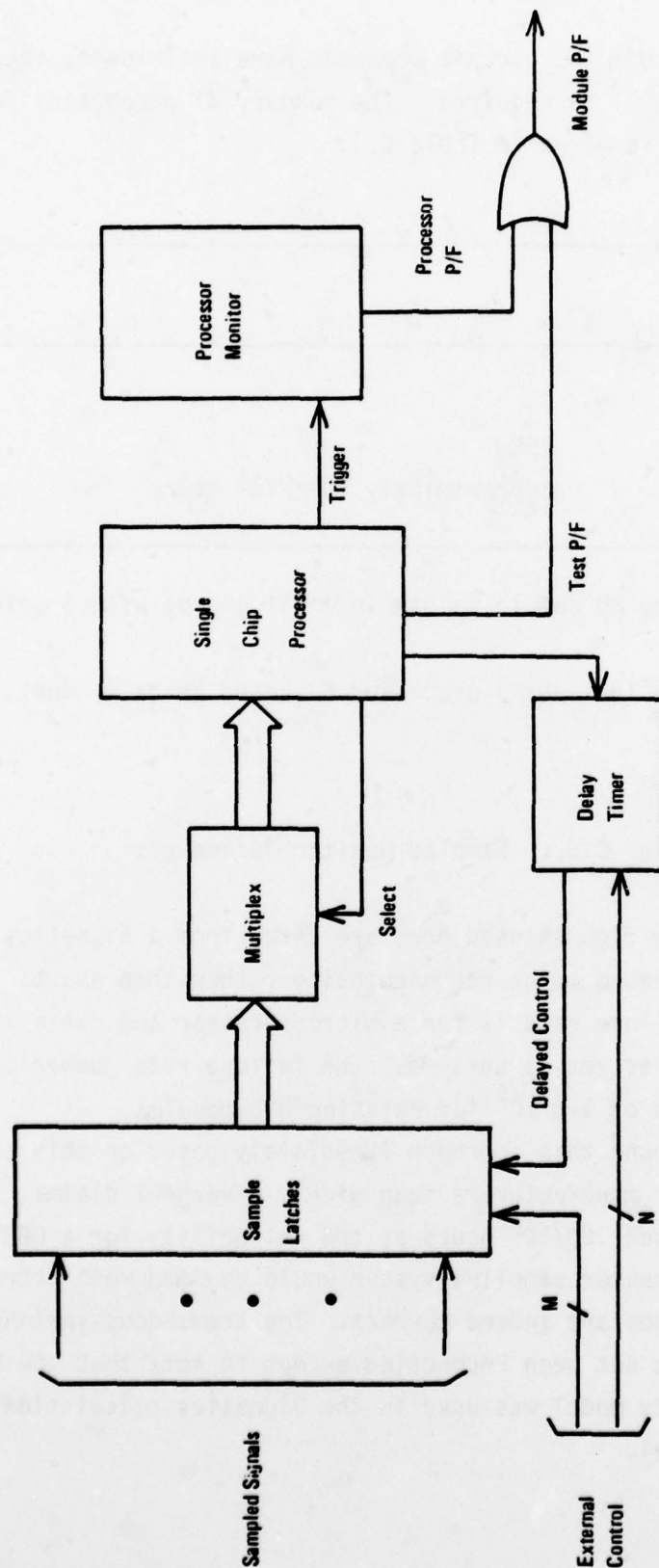


Figure C.8 Hardware Implementation of Microprocessor BIT

from module to module. The circuit proposed here anticipates that many such signals per module will be required. The summary of parameters for the sampled monitor BIT is given in Table C.3.

Sampled Monitor*

Number of packages	74
Number of gates	2500
Failure Rate	approximately $10^{**}/10^6$ hours

* Capable of handling 80 sample points in 4 bit groups with 5 delayed control signals.

** This FR estimation includes a processor FR based on gate count, MIL-HDBK-217B.

Table C.3. Sampled Monitor Parameters

The reliability figures used here are taken from a Signetics data catalog and are intended to be representative rather than exact. The catalog value of failure rate is for a microprocessor and not a single chip computer which as yet is unrated. The failure rate number can be related to FR values of $1-8/10^6$ for existing QED modules.

One might discount this approach immediately based on this statistic. However, other manufacturers make widely divergent claims. Another manufacturer uses $.22/10^6$ hours as the reliability for a CPU element. A microcomputer sampling system would be a much more attractive option if these values are indeed correct. The tremendous variance in Failure Rates has not been reconciled except to note that the MIL-HDBK-217B reliability model was used in the Signetics calculation and was not used by Intel.

APPENDIX D

Bibliography

Abraham, J. A., "An Algorithm for the Accurate Reliability Evaluation of TMR Networks," Digest of the 1973 International Symposium on Fault-Tolerant Computing, Palo Alto, California, IEEE Computer Society, 1973.

Alberts, R. D., Clary, J. B., Gault, J. W., Weikel, S. J., Whisnant, R. A., "A Study of a Standard BIT Circuit," Interim Technical Report, Research Triangle Institute, Naval Avionics Facility, Indianapolis, Contract No. N00163-76-C-0231, September 1976.

Anderson, D. A. and Metze, G., "Design of Totally Self-Checking Check Circuits for m-out-of-n Codes," IEEE Transactions on Computers, Vol. C-22, No. 3.

Arnold, T. F., "The Concept of Coverage and Its Effect on the Reliability Model of a Repairable System," IEEE Transactions on Computers, Vol. C-22, No. 3, March 1973.

Avizienis, A., "Architecture of Fault-Tolerant Computing Systems," Fault-Tolerant Computing Conference, 1975.

Avizienis, A. and Parhami, B., "A Fault-Tolerant Parallel Computer System for Signal Processing," Digest of the Fourth International Symposium on Fault-Tolerant Computing, Urbana, Illinois, IEEE Computer Society, June 1974.

Avizienis, A., "Arithmetic Algorithms for Error-Coded Operands," IEEE Transactions on Computers, Vol. C-22, No. 6, June 1973.

Avizienis, A., "The Methodology of Fault-Tolerant Computing," Proceedings of the First USA-Japan Computer Conference, Tokyo, October 1972.

Avizienis, A., "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design," IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971.

Avizienis, A., Gilley, Mathus, F. P., Rennels, D. A., Rohr, J. S., Rubin, D. K., "The STAR (Self-Testing And Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971.

Avizienis, A., "Design of Fault-Tolerant Computers," AFIPS Conference Proceedings, Vol. 31, Thompson Books, Washington, D. C., 1967.

Barsi, F. and Maestrini, P., "Error Detection and Correction by Product Codes in Residue Number Systems," IEEE Transactions on Computers, Vol. C-23, No. 9, September 1974.

Benowitz, N., Bork, R. H., Calhoun, D. F., Lee, G. W. K., Shen, J. B., "Advanced Avionics Fault Isolation System (AAFIS) Application, Vol. I: AAFIS Application," Hughes Aircraft Company, Culver City, California, May 1974.

Bouricius, W. G., Carter, W. C., and Schneider, P. R., "Reliability Modeling Techniques for Self-Repairing Computer Systems," Proceedings of 24th National Conference of ACM, 1969.

Bricker, J. L., "A Unified Method for Analyzing Mission Reliability for Fault-Tolerant Computer Systems," IEEE Transactions on Reliability, Vol. R-22, No. 2, June 1973.

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

Carter, W. C. and Bouricius, W. G., "A Survey of Fault-Tolerant Computer Architecture and Its Evaluation," Computer, Vol. 14, No. 1, January - February 1971.

Carter, W. C. and McCarthy, C. E., "Implementation of an Experimental Fault-Tolerant Memory System," Digest of the Fifth Annual International Symposium on Fault-Tolerant Computing, IEEE Computer Society, Paris, France, June 1975.

Carter, W. C. and Schneider, P. R., "Design of Dynamically Checked Computers," Information Processing 68 (Proceedings of IFIP Congress 1968), North Holland Publishing Company, Amsterdam, 1969.

Carter, W. C., Wadia, A. B., Jessep, Jr., D. C., "Computer Error Control by Testable Morphic Boolean Functions - A Way of Removing Hardcore," Digest of the 1972 International Symposium on Fault-Tolerant Computing, Newton, Massachusetts, IEEE Computer Society, June 1972.

Chien, R. T., "Memory Error Control: Beyond Parity," IEEE Spectrum, Vol. 10, No. 7, July 1973.

Clary, J. B. and Weikel, S. J., "A Study of a Standard BIT Circuit," Preliminary Specification, Research Triangle Institute, Naval Avionics Facility, Indianapolis, Contract No. N00163-76-C-0231, December 1976.

Cook, R. W., Sisson, W. H., Storey, T. F., and Toy, W. N., "Design of a Self-Checking Microprogram Control," IEEE Transactions on Computers, Vol. C-22, No. 3, March 1973.

Cha, Charles Wei-Yuan, "Multiple Fault Diagnosis in Combinational Networks," Technical Report, Illinois University Urbana Coordinated Science Lab, June 1974, (AD/7813 25).

Chih-Chun Ko, Danny, "Self-Checking of Multi-Output Combinational Circuits Using Forced-Parity Technique," Doctoral Thesis, June 1973, (AD/7650 37/7).

Dandapani, R., Reddy, S. M., "On the Design of Logic Networks with Redundancy and Testability Considerations," IEEE Transactions on Computers, Vol. C-23, No. 11, November 1974.

Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., "No. 1. ESS Maintenance Plan," The Bell System Technical Journal, Vol. 43, No. 5, Part 1, September 1964.

Eddington, D. C., "Maintenance Concept for QED Modules," Navy Electronics Laboratory Center, San Diego, California, January 1975.

Friedman, A. D., "A New Measure of Digital System Diagnosis," Proceedings Fault Tolerant Computing Symposium, 1975.

Gaddess, T. G., "Improving the Diagnosability of Modular Combinational Logic by Test Point Insertion," Coordinated Science Lab, University of Illinois, Urbana, Illinois, Report R-409, March 1969.

Garner, H. L., "The Residue Number System," IRE Transactions on Electronic Computers, June 1959.

Goldberg, J., Levitt, K. N., Short, R. A., "Techniques for the Realization of Ultra-Reliable Spaceborne Computers," Stanford Research Institute, Final Report-Phase I, SRI Project 5580, September 1966.

Goodman, David M., "Advanced Techniques for Automatic Testing and Built-In-Test Equipment (BITE) for Test, Measurement, and Diagnostic Equipment (TMDE)," Sete Workshop Proceedings, February 1974, (AD/A005 670).

Griesmer, J. E., Miller, R. E., Roth, J. P., "The Design of Digital Circuits to Eliminate Catastrophic Failures," Redundancy Techniques for Computing Systems, Spartan Press, Inc., Washington, D. C., 1962.

Grohman, D. M. and Sergeev, B. G., "Automated System for the Control of Digital Modules of Computers," Technical Report, Foreign Technology Division Wright-Patterson Air Force Base, Ohio, October 1975, (AD/A001 80).

Harrison, D., "Built-In-Test for Digital LSI Circuitry and Complex Digital Modules," Interim Report, Navy Electronics Laboratory, San Diego, California, Division, Code 5200, 1976.

Hayes, John P., "Detection of Pattern-Sensitive Faults in Random-Access Memories," IEEE Transactions on Computers, Vol. C-24 n 2, February 1975.

Hecht, H., "Fault Tolerant Software for Spacecraft Applications," Final Report, Aerospace Corporation, El Segundo, California Division, December 1975.

Hopkins, A. L., Jr., "A Fault-Tolerant Information Processing Concept for Space Vehicles," IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971.

Ingle, A. D. and Siewiorek, D. P., "A Reliability Model for Various Switch Designs in Hybrid Redundancy," IEEE Transactions on Computers, Vol. C-25, No. 2.

Jack L. A., Heimerdinger, W. L., Johnson, M. D., "Theory of Fault Tolerance," Annual Report, Honeywell, Inc., Minneapolis, Minnesota, Systems and Research, August 1975.

Jensen, P. A., "Quadded NOR Logic," IEEE Transactions on Reliability Vol. R-12, September 1963.

Keiner, William L., "The NWL Logic Simulation Program: Tool for Maintainable Digital Design," Technical Report, Naval Weapons Lab, Dahlgren, Virginia, April 1974.

Kennedy, P. J. and Quinn, T. M., "Recovery Strategies in the No. 2 Electronic Switching System," Digest of 1972 International Symposium on Fault-Tolerant Computing, IEEE Computer Society, June 1962.

Kolupaev, S. G., "Self-Testing Residue Trees," Stanford Electronics Lab, Stanford University, Technical Report No. 49, August 1973.

Kuehn, R. E., "Computer Redundancy: Design, Performance, and Future," IEEE Transactions on Reliability, Vol. R-18, No. 1, February 1969.

Lyons, R. E. and Vanderkulk, W., "The Use of Triple Modular Redundancy to Improve Computer Reliability," IBM Journal of Research and Development, Vol. 6, No. 2, April 1962.

Marver, J. M., "Fault Tolerance in Galois Trees: An Algorithm for Detection and Location of Stuck-at Type Errors in Trees of Galois Linear Modules," Technical Report, Sperry Univac, St. Paul, Minnesota, Defense Systems Division, Office of Naval Research, Arlington, Virginia, June 1975, (AD/A012 397).

Mastranadi, J. F., "Fault-Tolerant Storage Systems Design Studies," Digest of Third International Symposium on Fault-Tolerant Computing, IEEE Computer Society, Palo Alto, California.

Mathur, F. P. and Avizienis, A., "Reliability Analysis and Architecture for Hybrid-Redundant Digital System: Generalized Triple Modular Redundancy with Self-Repair," AFIPS Conference Proceedings, Vol. 36, AFIPS Press, 1970.

McCluskey, E. K., Wakerly, J. F., Ogus, R. C., "Current Research," Technical Report No. 100, Center for Reliable Computing, Stanford University, Stanford, California, October 1975.

McKenna, John F., Jr., "Demonstration and Evaluation of a Fault-Tolerant Input/Output Network," Final Report, Lab, Inc., Cambridge, Massachusetts, Office of Naval Research, Arlington, Virginia, August 1975.

Merryman, P. M. and Avizienis, A., "Modeling Transient Faults in TMR Computer SYstems," Proceedings 1975 Annual Reliability and Maintainability Symposium, Washington, D. C., January 1975.

Mine, H. and Koga, Y., "Basic Properties and a Construction Method for Fail-Safe Logical Systems," IEEE Transactions on Electronic Computers, Vol. EC-16, 1967.

Murphy, Lee R., Avizienis, Algirdas A., Rennels, David A., McNeely, Lyle D., Fulton, Roger L., "Fault Tolerant Avionics Systems Architectures STudy," Final Report, Ultrasystems, INC., Newport Beach, California, Air Force Avionics Lab, Wright-Patterson Air Force Base, Ohio, June 1974, (AD/7848 79).

Neumann, P. G., Levitt, K. N., Goldberg, J., and Wensley, J. H., "A Study of Fault-Tolerant Computing," Final Report, Stanford Research Institute, July 1973.

Neumann, P. G. and Rao, T. R. N., "Error-Correcting Codes for Byte-Organized Arithmetic Processors," IEEE Transactions on Computers, Vol. C-24, March 1975.

Newton, R. S., "A Review of Innovative Computer Architecture," Technical Report, Royal Radar Establishment Malvern (England), Defence Research Information Centre, Orpington (England), May 1975, (AD/7848 84).

Ozguner, Fusun, "Design of Totally Self-Checking Asynchronous Sequential Machines," Technical Report, Illinois University Urbana Coordinated Science Lab, Army Electronics Command, Fort Monmouth, New Jersey, May 1975, (AD/A010 719).

Parhami, Behrooz, "Techniques for the Design and Evaluation of Self-Checking Systems," Final Report, Ultrasystems, Inc., Newport Beach, California, Information Systems Division, February 1974, (AD/7759 09).

Pascoe, B., "Reliability Report -8080/8080A," Intel Corporation, Santa Clara, California, 1976.

Patel, Mahesh, "Fault Diagnosis by Inserting the Minimum Number of Test Points in System Graphs," Technical Report, Montana State University, Bozeman Electronics Research Lab, Office of Naval Research, Arlington, Virginia, July 1974, (AD/7852 72).

Pierce, W. H., "Adaptive Vote-Takers Improve the Use of Redundancy," Redundancy Techniques for Computing Systems, Spartan Press, Inc., Washington, D. C., 1962.

Peterson, Jr., W. W., Error Correcting Codes, MIT Press, Cambridge, Massachusetts, 1961.

Peterson, Jr., W. W. and Brown, D. T., "Cyclic Codes for Error Detection," Proceedings of the IRE, January 1961.

Peterson, Jr., W. W. and Weldon, E. J., Error Correcting Codes, MIT Press, Cambridge, Massachusetts, 1972.

Pierce, W. H., "Interwoven Redundant Logic," Journal of the Franklin Institute, Vol. 277, No. 1, January 1964.

Poulson, J. (Team Leader), "QED Module Built-In-Test Evaluation," Navy Electronics Laboratory Center, San Diego, California.

Putzolu, G. P. and Routh, J. P., "A Heuristic Algorithm for the Testing of Asynchronous Circuits," IEEE Transactions on Computers, Vol. C-20, No. 6, June 1971.

Preparata, F. P., Metze, G., and Chien, R. T., "On the Connection Assignment Problem of Diagnosable Systems," IEEE Transactions on Electronic Computers, Vol. EC-16, 1967.

Pribble, Howard Warren, "Effects of Redundancy on Fault Detection and Diagnosis on Fault Detection and Diagnosis in Combinational Logic Circuits," Technical Report, Illinois University Urbana Coordinated Science Lab, Army Electronics Command, Fort Monmouth, New Jersey, September 1974, (AD/7874 63).

Ramamoorthy, C. V. and Chang, L. C., "System Modeling and Testing Procedures for Microdiagnostics," IEEE Transactions on Computers, Vol. C-21, No. 11, November 1972.

Ramamoorthy, C. V. and Han, Y. W., "Reliability Analysis of Systems with Concurrent Error Detection," IEEE Transactions on Computers, Vol. C-24, No. 9, September 1975.

Rennels, David A., "Fault Detection and Recovery in a Redundant Computer Using Standby Spares," Technical Report, California University, Los Angeles, School of Engineering and Applied Science, July 1973, (PB/2269 48/8).

Roth, J. P., "Diagnosis of Automata Failures: A Calculus and a Method," IBM Journal, July 1966.

Roth, J. P., Bouricius, and Schneider, P. R., "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," IEEE Transactions on Electronic Computers, Vol. EC-16, No. 5, October 1967.

Russell, J. D. and Kime, C. R., "System Fault Diagnosis: Closure and Diagnosability with Repair," IEEE Transactions on Computers, Vol. C-24, November 1975.

Tyson, Thomas E., "An Introduction to Fault-Tolerant Design Techniques," Technical Report, Air Force Avionics Lab, Wright-Patterson Air Force Base, Ohio, June 1974, (AD A013934).

von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Automata Studies, C. C. Shannon and J. McCarthy, eds., Annals of Math. Studies No. 34, Princeton University Press, 1956.

Wakerly, J. F., "Low Cost Error Detection Techniques for Small Computers," Technical Report No. 51, Digital Systems Laboratory, Stanford University, Stanford, California, December 1975.

Watson, R. W. and Hastings, C. W., "Self-Checked Computation Using Residue Arithmetic," Proceedings of the IEEE, Vol. 54, No. 12, December 1966.

Wensley, J. H., Levitt, K. N., and Neumann, P. G., "A Comparable Study of Architectures for Fault-Tolerance," Proceedings Fault Tolerant Computing Symposium, 1974.

White, Donnamaie E., "Fault Detection Through Parallel Processing in Boolean Algebra," Group Series Report, California University Los Angeles Department of Computer Science, Office of Naval Research, Arlington, Virginia, March 1975, (AD/A007 552).

Williams, M. J. Y. and Angell, J. B., "Enhancing Testability of Sarge-Logic," IEEE Transactions on Computers, Vol. C-22, No. 1, January 1973.

Williams, M. J. Y. and Angell, J. B., "Enhancing Testability of Large-Scale Integrated Circuits Via Test Points and Additional Logic," IEEE Transactions on Computers, Vol. C-22, January 1973.