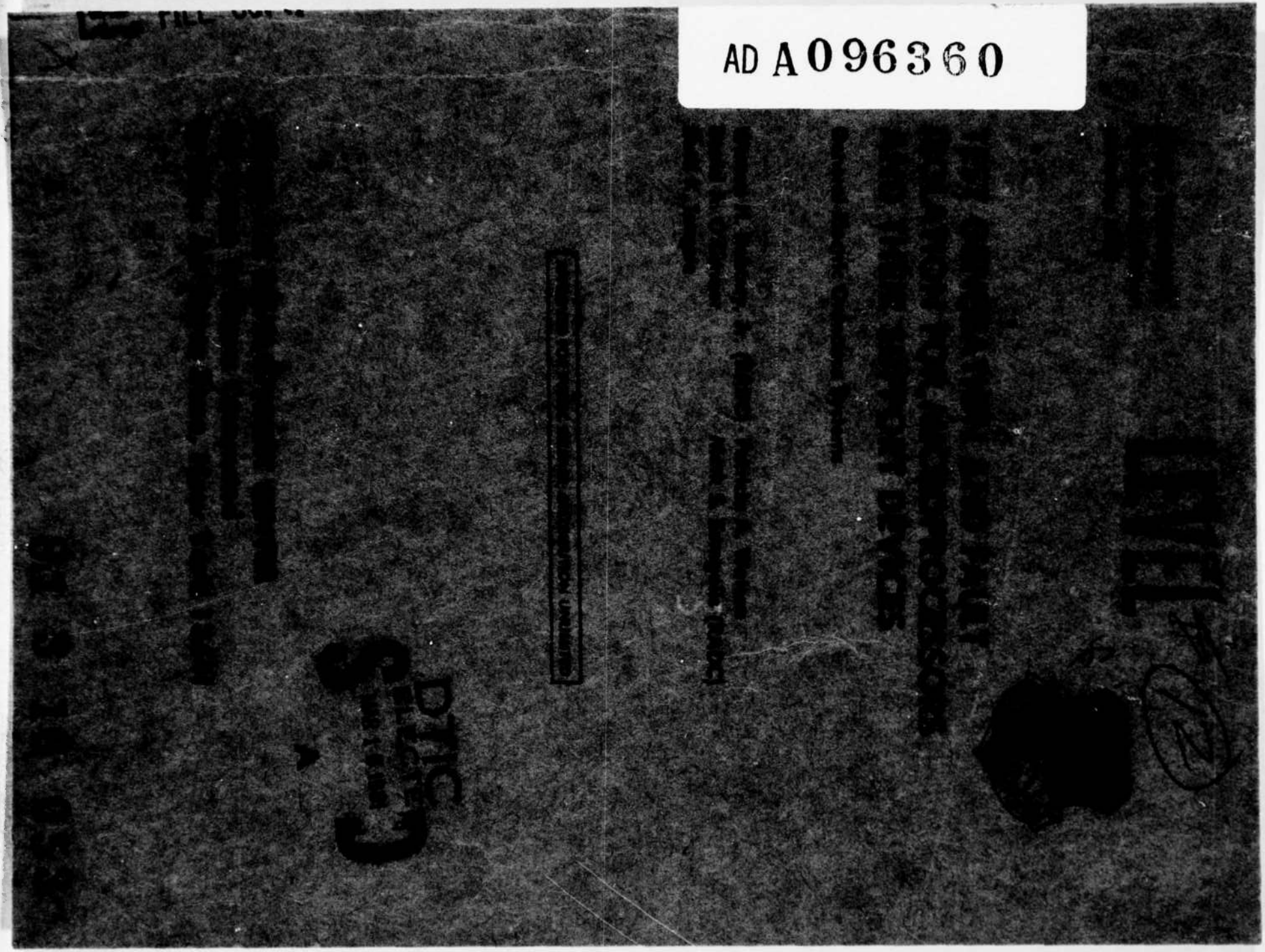
AD-A096 360 GENERAL ELECTRIC CO PITTSFIELD MA ORDNANCE SYSTEMS F/G 9/2
TEST GENERATION AND FAULT ISOLATION FOR MICROPROCESSORS AND THE--ETC(U)
NOV 80 W H DEBANY, D A O'CONNOR, B K TEAGUE F30602-78-C-0235
UNCLASSIFIED RADC-TR-80-274 NL

1 OF 2
AD A
096360


A
963



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

RADC-TR-80-274 has been reviewed and is approved for publication.

APPROVED: Warren H. DeBary, Jr.

WARREN H. DEBARY, JR.
Project Engineer

APPROVED: David C. Luke

DAVID C. LUKE, Colonel, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

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

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-80-274	2. GOVT ACCESSION NO.: AD-A096360
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): TEST GENERATION AND FAULT ISOLATION FOR MICROPROCESSORS AND THEIR SUPPORT DEVICES	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report Jul 78 — Jul 80
6. PERFORMING ORG. REPORT NUMBER: N/A	
7. AUTHOR(s): Warren H. Debany, Jr. (RADC); David A. O'Connor; Patricia A. Watson; Basil K. Teague; Mark S. Zemgulis (RADC)	8. CONTRACT OR GRANT NUMBER(s): F30602-78-C-0235
9. PERFORMING ORGANIZATION NAME AND ADDRESS: General Electric Ordnance Systems, Circuit Test Engineering, 100 Plastics Ave, Pittsfield MA 01201	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F; 23380152; 91
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (RBRA), Griffiss AFB NY 13441	12. REPORT DATE: November 1980; 13. NUMBER OF PAGES: 191
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Warren H. Debany, Jr. (RBRA)

This report was written jointly by GEOS and RADC.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Automatic Test Equipment	IC	Scanning Electron Microscope
Computer-Aided Test	LASAR	
Device Under Test	Logic Integrity Test	MIL-M-38510
FACTOR	LSI, VLSI	TEKTEST
High Level Language	Qualified Parts List	VHSIC (See Reverse)

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

This report covers the generation of tests for complex digital devices, implementation of those tests on currently available ATE, and fault isolation.

Techniques are described for the modeling of devices in order to facilitate the testing process. A new algorithm for fault dictionary searching is given. Voltage Contrast (on the Scanning Electron (Cont'd

next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

149175

Item 20 (Cont'd)

Microscope) was used as an aid for fault insertion, which in turn provided test cases for fault isolation. Computer-Aided Test techniques are described. As the demonstration vehicle for this work was the development of MIL-M-38510 Detail Specifications (Slash Sheets) a tutorial description of a recommended slash sheet format is given. Many areas were only touched on during this contract; these have been described, with suggestions for further research topics.

Item 19 (Cont'd)

Computer Programs
Software

ACKNOWLEDGEMENTS

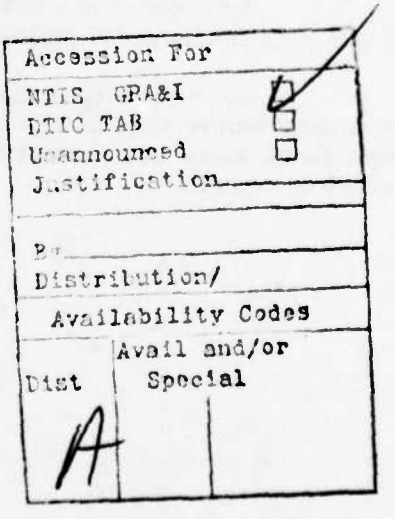
The authors wish to offer their special thanks to the following persons, without whose help this report could not have been completed:

Tom Dellecave, William Bader, and Jack Haberer, who reviewed sections for us.

Patti Schram, Lorraine Armstrong, and Marie St. Thomas, for assistance in the data entry and manuscript preparation.

Evelyn Mesagna, whose foresight and planning made the manuscript preparation possible.

Allen Converse, most sincere thanks for the many unpaid weekend and evening hours he spent formatting the final text for the word processor.



SUMMARY

This report is the result of a joint effort by General Electric Ordnance Systems and RADC, to improve RADC's in-house testing capability for complex digital microcircuits.

All of the topics covered relate to either the generation of tests, or the reduction of the data to provide fault isolation.

Section 1 of this report concerns the philosophy held by the investigators at the end of the effort. In many cases, this philosophy differs greatly from that with which the effort started.

Section 2 deals with the development of MIL-M-38510 "Slash Sheets," and the problems encountered when implementing the Slash Sheet tests on current Automatic Test Equipment.

Section 3 discusses the various software tools developed for use during this effort. Two types are shown: those running on the RADC Multics facility, and those which run on the S-3260/3270 IC Tester.

Section 4 describes the development of a test, including the items which should be considered during test generation, and the implementation of the test with the help of ALICE and LASAR.

Section 5 covers the reduction of failure data, producing a concise listing of possible bad nodes in a device (fault isolation). A new matching algorithm for the fault dictionary search was developed for this, and is described in this section.

Section 6 documents the techniques for fault insertion that were used in this effort. This was done to provide test cases for fault isolation.

Section 7 outlines the progress made with each of the devices studied under this effort. The "learning" process, test generation and implementation, fault insertion and isolation are covered.

Section 8 summarizes and provides some recommendations.

Appendix A is a tutorial on the use of Automaton Diagrams, which were one of the most fundamental tools used in this effort.

No "user's manuals" are provided here, as the techniques require further debugging before they should be declared to be complete. Once the rough spots have been fixed, more documentation will most likely follow, in the form of monographs or technical reports.

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

.ASC	-	ASCII-type file, used on the S-3260/3270
.EDT	-	S-3260/3270 source program
.PAT	-	S-3260/3270 pattern file
.PIN	-	S-3260/3270 file for defining the configuration of the test fixture
.TST	-	S-3260/3270 compiled program
ALICE	-	Automatic LASAR Input Coding Editor, the generic name for several Computer-Aided Testing utilities
ATE	-	Automatic Test Equipment
CAT	-	Computer-Aided Test
CRT	-	Cathode Ray Tube (television screen)
Decap	-	De-encapsulate
DUT	-	Device Under Test
FACTOR	-	Fairchild Sentry test language
G	-	Goodness of fit
HLL	-	High Level Language
I/O	-	Input/Output
IC	-	Integrated Circuit
LASAR	-	Hardware simulation program
LIT	-	Logic Integrity Test
LSI	-	Large Scale Integration (roughly 100 to 1000 gates)
LUN	-	Logical Unit Number, on S-3260/3270
Mode 1	-	S-3260/3270 mode where any sector can either force, or inhibit, or compare, or mask
Mode 3	-	S-3260/3270 mode where any sector can either force or inhibit, or it can compare or mask
NRZ	-	Non-Return Zero

PDA	-	Percent Defective Allowable
PE Report	-	Product Evaluation Report
QPL	-	Qualified Parts List
RAM	-	Random Access Memory .
ROM	-	Read-Only Memory
RTO	-	Return To One
RTZ	-	Return To Zero
S	-	Fault Signature
SA0	-	Stuck At Zero
SA1	-	Stuck At One
SEM	-	Scanning Electron Microscope (not to be confused with Standard Electronic Module, which is not mentioned in this report)
Slash Sheet	-	A detail specification added to MIL-M-38510, General Specification for Microcircuits
TCL	-	Testing Confidence Level
TEKTEST	-	Tektronix test language
TSCU	-	Test Station Control Unit
UID	-	User Identification (on S-3260/3270, it is a three-character name)
VHSI	-	Very High Speed Integration
VLSI	-	Very Large Scale Integration (roughly from 1000 gates, up)
Z	-	An entry in the fault dictionary
Z-entry	-	An entry in the fault dictionary

EVALUATION

The objective of this effort was to develop new techniques for the testing of complex digital microcircuits, as well as improve existing techniques.

As the preparing activity for MIL-M-38510 "Slash Sheets," RADC has attempted to stay at the leading edge of testing technology. It is felt that it is more important, in the area of electrical characterization, to make intelligent use of existing resources, and refine the techniques for using these resources, rather than constantly have to procure "fancy" new equipment. The techniques outlined in this report are either usable directly on commonly-available test systems, or compatible with them.

General Electric Ordnance Systems provided much of the know-how and practical applications described here. This, and an earlier effort, helped to improve in-house expertise in this area, at RADC. This report is the result of a joint writing effort, between GEOS and RADC, which attempts to adequately describe the in-house and out-of-house synergism which took place.

WARREN H. DEBANY, JR.
Project Engineer

1. Introduction

The age of extensive data processing on a single integrated circuit has arrived. Today's equipment designs call for small boxes that have many times the throughput of entire rooms of older equipment. This increase in capability is being accomplished by reducing the signal path length (because of limitations imposed by the speed of light) and the signal swing (because of rise and fall times, or switching speed). These tactics reenforce the trend of designing more processing into each package. In the future, multi-package designs may be practical only for parallel data paths (as presently shown in bit-slice microprocessors), with elaborate carry look-ahead schemes implemented to reduce interface considerations and off-board processing.

Avoiding intricate and fault-prone interconnections dictates that each device must have a severely limited number of connections to the outside world. If consideration is not given to Design-for-Testability, then the testing of buried sections of a Very Large Scale Integrated Circuit (VLSIC) may be expected to increase exponentially in difficulty in the near future.

The testability of presently available integrated circuits is seldom adequate and is usually beyond the control of the user. The work described in this report addresses approaches that can be taken to test these devices more thoroughly:

1) Generate tests for fault detection,

2) Perform fault isolation,

3) Provide feedback to IC manufacturers on the operation and failure modes of their devices.

The demonstration vehicle for the test generation is the MIL-M-38510 Standardization Program for Microcircuits. The "Detail Specifications" for devices on the Qualified Parts List (QPL) include comprehensive tests intended to screen out devices which do not function correctly or are likely to fail in use. It is hoped that techniques outlined in this report may help to standardize test generation methods for military parts to a consistently high Testing Confidence Level (TCL).

A detail specification (also called a "slash sheet") is judged not only on its test completeness, but also on its timeliness. Ideally, a slash sheet should be prepared quickly to encourage the use of newer, more capable parts in military systems.

The key to accuracy and fast turnaround is Computer-Aided Testing (CAT). The test writer should not be concerned with the mundane details of the manipulation of large vector sets, but should instead prepare his test plan in a human-oriented shorthand. Utility programs (ALICE, etc.) that assist in test generation are described in this report.

Test failure data reduction and fault isolation techniques are developed to both verify the completeness of the test generation methods and provide tools to aid in reliability investigations. A small number of devices were deliberately faulted during this effort to demonstrate the correct operation of the fault isolation routines.

Accurate tests and failure data represent valuable information to the IC manufacturers. This data, reduced by computer to a concise and meaningful form, can

help increase the yield and reliability of their products. This will benefit both military and industrial users by providing better and lower cost devices.

A brief discussion of testing philosophy is in order here. When possible, both functional and structural models were made for each device tested. The functional representation used was the Automaton Diagram, which expresses the operation of the device in terms of registers, data selectors, and boolean function blocks. Parallel data paths are immediately apparent to the user and test generation is facilitated. The structural representation chosen was the NAND gate equivalent network used by LASAR. This software package was selected partly because of its availability to both the contractor and to RADC, and in addition, it is the purest of structurally-based techniques and provides an extreme against which to measure other techniques. The automatic test pattern generation (STIMGN) feature was not used extensively during this effort. Instead, the input patterns were generated using manual techniques. The LASAR program then generated the expected outputs (using SIMUL), graded the pattern for TCL (using DYSOGN) and developed the fault dictionary (using REDUCE). Automaton Diagrams, shorthand utility programs, and functional block "recipe" test approaches were used to manually generate the input vectors. Then, the LASAR output was used to home in on undetected faults, thus improving the TCL. This procedure was repeated until an adequate TCL was obtained. An adequate TCL is considered to be 100% of all faults which can possibly be detected. This technique proved to be quick and accurate, as fault injection experiments showed.

As LASAR and most practical fault simulators assume a single-stuck-at failure mechanism, the approach used in this effort may be questioned as being unrealistic. However, the evidence so far indicates that such is not the case. Admittedly, only a small number of devices were deliberately faulted, but in each case the faults were isolated to the correct region, if not to the particular gate or metal line.

The difference between proposing to isolate to a region, instead of a node is significant. There is certainly not a one-to-one mapping between a gate-equivalent model and the actual implementation. However, there is by necessity such a mapping from the functional level to the silicon level. By recognizing that the gate-equivalent model is only a convenient language for describing the function, and nothing more, great progress may be made.

By analogy, note that when solving a system of equations, the engineer never confuses the significance of the symbols, and their properties, with that of the real-world problem being solved. A notation or a language, elegant though it may be, is not sacred. Failure to recognize this, and acceptance of results at no more or less than face value, is a great pitfall in the testing of complex devices. It is important to cultivate scepticism.

By diminishing its importance, the single-stuck-at model has become more powerful. The techniques based on it have led to the recipe approach. The recipe of tests for a particular block of the Automaton Diagram is applied through the necessary connective logic with the results brought to the output of the DUT.

If the equivalent of a memory bit map were available for a microprocessor, showing the location and route of every metal or polysilicon line, and if the features of every transistor, resistor, and capacitor were known, and it were possible to describe this to a computer program, and inject failures that exactly model those in the real world,

the results would certainly be better than those obtained with present test and fault isolation capabilities. However, this is still impossible even with small analog circuit test programs, and so could not be implemented for digital LSI/VLSI even if the design data were available. The single-stuck-at fault model, with knowledge of some pattern sensitivities, is still the best "a priori" approach.

It was beyond the scope of this effort to fully implement heuristic test generation and data reduction algorithms. However, extensive computer aids were developed to run under operator control. This report, although not a complete user's manual for all utilities described herein, should be sufficient to provide the basis for a complete test generation facility. Further documentation can be obtained by qualified DOD contractors or users whose testing can be shown to benefit the DOD.

2. Specification Writing

The specification for an individual device (called a "slash sheet") contains all the information needed to use and test the device. The information is presented in a concise, easily understood format, but includes references to the base documents, MIL-M-38510 and MIL-STD-883, for predefined requirements or procedures.

The basic goals of specification writing are to assure device reliability, standardization, and availability during a system's life cycle. To achieve these goals a specification must be written with an eye toward both clarity and completeness in order to expeditiously establish a qualified sources list, and for use by system builders. By clarity it it meant that the information will be easily understood by the user without the extensive cross referencing of other documents, especially those that are unique to a specific application. References to other military specifications will include details such as applicability, environmental, mechanical, and electrical requirements, exceptions and options. The latter two items should be made as rare as practical. By completeness it is meant that that all pertinent device characteristics, parametric limits, test conditions and special considerations must be included in detail. Device characteristics include mechanical, thermal, and electrical properties. The parameters include input and output voltage and current levels, and timing relationships. The test conditions are the stimuli applied to the device under test to obtain the expected results. These includes any special drive circuitry or output loads connected to the Device Under Test (DUT). Special considerations may include such things as anti-static protection or unusual timing constraints. Burn-in circuits must be described.

2.1 Outline of a Slash Sheet

Slash sheets have been written in many formats, often making interpretation unnecessarily difficult. The following outline is offered as a guide to what the slash sheet paragraphs describe. The numbers in parentheses are the paragraph numbers of slash sheet MIL-M-38510/444 for the 2914, Vectored Priority Interrupt Encoder, which may be referenced for examples.

The slash sheet indentifier is "MIL-M-38510/" followed by a three digit number that is unique to a function, and a two digit number (called dash number) which specifies the device type. The number is assigned by the issuing agency. The slash sheet title lists the microcircuit family, the technology, and the functional name of the device.

The paragraphs are as follows:

(1.1) -- Scope

The scope gives a brief description of the device family, the technology and the options.

(1.2) -- Part Number

The part number to be marked on the device is given, usually with reference to MIL-M-38510. It includes the device type (1.2.1), testing class (1.2.2), and case outline (1.2.3).

(1.3) -- Absolute Minimum and Maximum Ratings

The thermal and electrical conditions which are never to be exceeded should be listed. These parameters depend primarily on the technology used in manufacturing, but should include the following at the *minimum*:

— Supply voltage range,

-- Current applied at the outputs,

-- Voltage applied at the outputs,

-- Voltage applied at the inputs,

-- Power dissipation (including provision for test loading, if applicable),

-- Storage temperature range,

-- Soldering temperature and duration,

— Thermal resistance (junction to case),

-- Junction temperature.

Selected items in the above list may reference the device option (by dash number) when appropriate. Also, some parameters may apply only to given device pins. For example, open collector outputs may have different voltage or current limits than active pull up outputs.

(1.4) — Recommended Operating Conditions

The environmental and electrical conditions expected for normal operation are listed. The items usually listed are:

-- Supply voltage range,

-- Minimum input high level voltage,

-- Maximum input low level voltage,

-- Loading considerations (such as maximum capacitance on a MOS technology output),

-- Ambient operating temperature range.

These items may depend on the dash number or device pin.

(2) -- Applicable Documents

The principle military documents that are part of the device specification are listed, along with the statement that the issue in effect at the time of the invitation for

bid or request for proposal must be used. Also, a source for additional copies of the documents is given.

(3) -- Requirements

This paragraph lists all of the requirements that the device must meet. Information in other specifications, other paragraphs, tables or figures are included by reference as necessary.

(3.1) -- Detail Specification

This is a cover statement that makes the MIL-M-38510 base document a part of this specification by reference.

(3.2) -- Design, Construction and Physical Dimensions

The case outline(s), terminal connections, function block diagram, logical implementation (it is suggested that an Automaton Diagram be included) and schematic diagram requirements are given by reference. For the sake of consistency in future slash sheets, it is suggested that the case outline be in paragraph (1.2.3), the terminal connections in Slash Sheet Figure 1, the functional block diagram in Slash Sheet Figure 2, and the Automaton Diagram in Slash Sheet Figure 3. The schematic diagram need not be included but should be available to the procuring activity.

(3.3) -- Lead Material and Finish

Options selected from the MIL-M-38510 base document should be given.

(3.4) -- Electrical Performance Characteristics

This paragraph references Slash Sheet Table I, which lists the complete electrical parameters (design limits) of the device. The tables are discussed further in this report in section 2.2.

(3.5) -- Rebonding

The rebonding restrictions of the MIL-M-38510 base document should be noted.

(3.6) -- Electrical Test Requirements

Table II lists the required electrical subgroups from Group A for screening qualification and quality conformance needed to ensure that a device conforms to a given class.

The detailed electrical test requirements are given in Slash Sheet Table III. Since these tables list the exact electrical requirements, extreme care must be exercised to guarantee that they are clear, complete, and easily converted to the format required by the user's ATE.

(3.7) -- Marking

The marking shall conform to the MIL-M-38510 base document.

(3.8) -- Microcircuit Group Assignment

The group assignments in MIL-M-38510 do not presently include LSI devices, so this paragraph lists the microcircuit technology of the device.

(4) -- Quality Assurance Provisions

This paragraph details the qualification, quality conformance, and screening requirements referencing MIL-STD-883 and MIL-M-38510 where feasible.

(4.1) -- Sampling and Inspection

The sampling and inspection requirements are given with modifications to be applied as described in subparagraphs.

(4.2) -- Qualification Inspection

These requirements ensure that the device design and production techniques are adequate. They should include design dependent tests such as measurement of input capacitance and checks of operation under absolute maximum conditions.

(4.3) -- Screening

These tests are performed on reliability-critical parameters on every device to verify proper assembly and operation. Any options or criteria required by the MIL-M-38510 base document or MIL-STD-883 should be identified. The order of testing should be specified if it is significant. The Percent Defective Allowable (PDA) is given for certain subgroups.

(4.4) -- Quality Conformance Inspection

These tests are periodically performed on production lot samples to determine if there is lot-to-lot variability due to processing, packaging, etc. The subparagraphs list the tests for each group, including exceptions, options and additional requirements.

(4.5) -- Methods of Inspection

This paragraph gives any additional information, such as the voltage measurement reference point, that is needed by the user.

(4.6) -- Inspection of Packaging

This paragraph specifies the packaging inspection requirements and additions to MIL-M-38510.

(5) -- Packaging

The packaging requirements are given, usually by reference to the MIL-M-38510 base document. Additions such as protection for static sensitive devices should be included.

(6) -- Notes

(6.1) -- Notes

Miscellaneous additional information needed by the manufacturer or user is given.

(6.2) -- Intended Use

A general description of the expected device application is given.

(6.3) -- Ordering Data

A list of the details to be supplied by the procuring activity is given.

(6.4) -- Abbreviations, Symbols, and Definitions

The abbreviations, symbols and definitions are defined, usually by reference to MIL-STD-1331. Any symbols unique to this device are listed and defined.

(6.5) -- Logistic Support

Additional options such as lead material, length, or finish are given. Also, the default class and lead finish are given.

(6.6) -- Substitutability

The equivalent generic industry types are listed, along with a caution that they may have minor mechanical or packaging differences.

2.2 Detailed Test Requirements

In a slash sheet, Table I lists the DC and AC parameters of a device. Table II specifies when and what electrical tests are performed for each reliability class and the various inspection groups (environmental, die-attach, electrical, and packaging). Table III provides a breakout into subgroups of Table I tests over the military temperature range. The following paragraphs discuss techniques and limitations in developing Table III of slash sheet.

The subgroup numbers in Table III are assigned in accordance with MIL-STD-883, Method 5005, Table I. The symbols should be consistent with MIL-STD-1331. For consistency and ease of conversion to a tester computer program, the pin names should be placed in logical groups and in the same order that appears in the LIT pattern. The most important requirements are that Table III (test implementation), 1) be easily understood by the user and, 2) be easily converted to the format required by his tester.

2.2.1 DC Parametric Tests

The DC parametric tests verify the DC parameters of Table I of the slash sheet at the applicable device pins. The device is "conditioned" by applying a limited set of inputs, applying the required forced voltage or current to the pin under test and recording the resulting current or voltage. For LSI (or larger) devices, it is often convenient to apply a selected portion of the LIT pattern to establish the desired condition. The conditioning must ensure that bidirectional pins (if any) are in the proper state. Ideally, a pattern should be applied to input pins not under test that establishes a worst case condition on the pin under test (on a simple multiple emitter transistor gate, this is the limit of the opposite level). Subject to the constraints of worst case conditions, input and output pins not under test should be at the level opposite to the pin under test in order to verify pin to pin isolation, particularly during input current high and output voltage high tests. Note that the LIT will check most of the possible pin to pin shorts, but fault isolation is more convenient during the DC parametric testing.

The order of the tests in Table III should place the tests that may stress the DUT before ones that verify that the DUT is not damaged. Also, the loss of VCC from the tester should be detected in later tests so that tester caused errors can be identified. A suggested order of tests is listed below, but the actual tests required will vary with device technology and pin electronics implementation. For example, CMOS devices are not tested for input clamp diode voltage (VIC) and open collector output pins are not tested for output voltage high (VOH).

Order of DC parametric tests:

VOLL -- Voltage, output low-loaded,

IIB -- Current, input breakdown,

IOB -- Current, output breakdown,

VIC -- Voltage, input clamp diode,

VOC -- Voltage, output clamp diode,

IOS -- Current, output short circuit,

IIH -- Current, input high logic level,

IIL -- Current, input low logic level,

IOZH -- Current, output high impedance, high logic level (for three-state pins),

IOZL -- Current, output high impedance, low logic level (for three-state pins),

IOH -- Current, output high (for open collector pins),

VOL -- Voltage, output low logic level,

VOH -- Voltage, output high logic level,

ICC -- Current, power supply.

2.2.2 Logic Integrity Test (LIT)

The LIT portion of the slash sheet Table III is used to verify that the device will perform the required function. The table must specify the power supply voltage(s), the input voltages (for logic 0 and logic 1), timing relationships and output voltages and loading. A sample load circuit is shown in Figure 2.1. The user may be given the option of omitting the tests for VOH and VOL if the LIT is performed with all outputs loaded and the outputs detected exceed the VOH and VOL levels.

The logic levels to be applied to the inputs and expected logic levels at outputs (collectively called the test "vectors") must be provided in detail. Since the number of vectors needed to fully check an LSI device can be large (usually greater than 2000), it is recommended that they be stored on magnetic tape and optionally printed in the slash sheet. The tape storage will also allow the use of computers to reformat the vectors to be compatible with the user's test system. The vector set manipulation is discussed further in sections 3 and 4.

A sample of a convenient vector set notation is shown in Figure 2.2.

The checklist in section 4.1 should be used as a guide when developing the vectors.

2.2.3 AC Parametric Tests

The AC parametric tests check the setup and hold times and the propagation times listed in Table I of the slash sheet. Special vectors with appropriate signal timing and levels should be developed so that a minimum set of signals is required to sensitize the pin(s) under tests to the outputs. The vector set should be extensively commented to show the signal being checked, the pin(s) under test and test limits. Also, the format should ideally be the same as used for the LIT.

The AC parametric tests are normally performed with the outputs loaded. The load circuit(s) should be given in the slash sheets. An example is shown in Figure 2.1. See section 3.1.7 for a discussion of a program that will calculate the value of the load voltage and resistors.

Each independent portion of the vector set should initialize the DUT and test timing parameters of a logical group of signals, such as microprocessor instruction inputs. Some of the advantages of this kind of grouping are: easier determination of the failing pin; more efficient debugging by applying the vectors repeatedly for observation on an oscilloscope; easier evaluation of the vectors by the user.

2.2.4 Limitations and Comments

The slash sheet writer must remember that the slash sheet is for use by the DOD and its vendors and customers, meaning wide distribution to many companies, as well as other countries, may occur. Therefore, copyrighted or proprietary material should not be used.

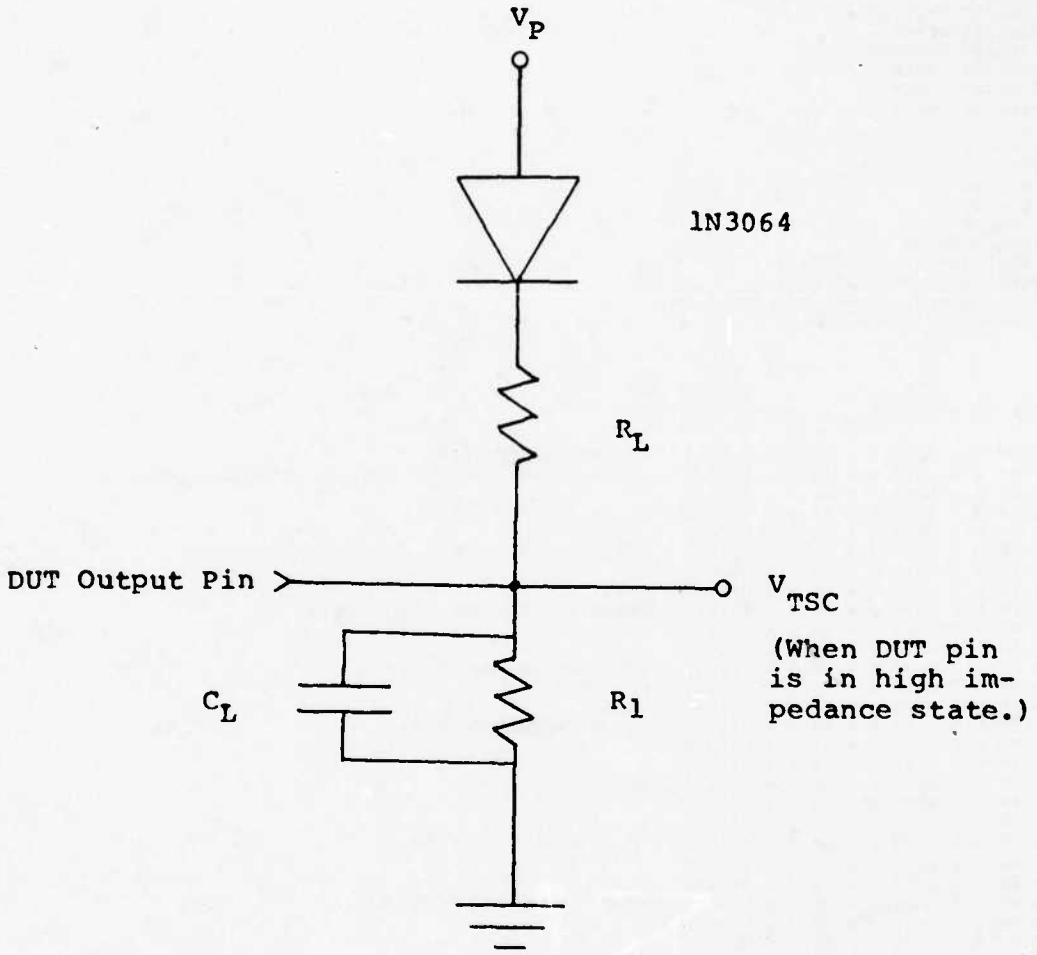


Figure 2.1. Output Load Circuit

```
FIGURE 1
LOGIC INTEGRITY TEST

/* PINLIST INPUTS=M7,M6,M5,M4,M3,M2,M1,M0,
/* CONTINUE S2,S1,S0,P7,P6,P5,P4,P3,P2,P1,P0,
/* CONTINUE IE,I3,I2,I1,I0,GE,GAR,ID,LB,CP
/* PINLIST OUTPUTS=M7,M6,M5,M4,M3,M2,M1,M0,
/* CONTINUE S2,S1,S0,V2,V1,V0,GS,GAS,SV,IR,PD,RD
/* PINLIST M=M7,M6,M5,M4,M3,M2,M1,M0
/* PINLIST S=S2,S1,S0
/* PINLIST P=P7,P6,P5,P4,P3,P2,P1,P0
/* PINLIST I=IE,I3,I2,I1,I0
/* PINLIST GE=GE
/* PINLIST GAR=GAR
/* PINLIST ID=ID
/* PINLIST LB=LB
/* PINLIST CP=CP
/* VCC=4.5V
/* VA=2.0V
/* VB=0.8V
/* VH=2.4V
/* VL=0.5V
/* CONNECT LOADS
/* PERIOD=500NS
/* FORCE INPUTS WITH VA,VB
/* COMPARE OUTPUTS WITH VH,VL
/* APPLY INPUTS AT 0NS FOR 500NS,NRZ
/* COMPARE AT 210NS FOR 20NS


                 G                    G
           I   GAIL C               GASIPR
M   S P    E I ERDB P    M   S V    SSVRDD

00 0 FF 0 0 0010 1 * XX X X X1XXXX    MASK REGISTER TEST
00 0 FF 0 0 0010 0 * XX X X X1XXXX    TEST LOADING THROUGH MASK REGISTER DATA SELECTOR, 0 POSITION
00 0 FF 0 0 0010 1 * XX X X 011110
00 0 FF 0 5 0010 1 * XX X X 011110
00 0 FF 0 5 0010 0 * XX X X 011110
ZZ 0 FF 0 7 0010 1 * 00 X X 011110    READ MASK REGISTER (SELECTOR POSITION 7)
ZZ 0 FF 0 7 0010 0 * 00 X X 011110
ZZ 0 FF 0 7 0010 1 * 00 X X 011110
00 0 FF 0 8 0010 1 * XX X X 011110    PRESET M REGISTER, POSITION 8
00 0 FF 0 8 0010 0 * XX X X 011110
00 0 FF 0 8 0010 1 * XX X X 011110
ZZ 0 FF 0 7 0010 1 * FF X X 011110    READ THROUGH POSITION 7
ZZ 0 FF 0 7 0010 0 * FF X X 011110
ZZ 0 FF 0 7 0010 1 * FF X X 011110
00 0 FF 0 C 0010 1 * XX X X 011110    CLEAR THROUGH POSITION C
00 0 FF 0 C 0010 0 * XX X X 011110
00 0 FF 0 C 0010 1 * XX X X 011110
ZZ 0 FF 0 7 0010 1 * 00 X X 011110    READ
ZZ 0 FF 0 7 0010 0 * 00 X X 011110
ZZ 0 FF 0 7 0010 1 * 00 X X 011110
55 0 FF 0 B 0010 1 * XX X X 011110    OR FUNCTION, POSITION B
55 0 FF 0 B 0010 0 * XX X X 011110
55 0 FF 0 B 0010 1 * XX X X 011110
ZZ 0 FF 0 3 0010 1 * 55 X X 011110    READ THROUGH POSITION 3
ZZ 0 FF 0 3 0010 0 * 55 X X 011110
ZZ 0 FF 0 3 0010 1 * 55 X X 011110
AA 0 FF 0 D 0010 1 * XX X X 011110    COPY THROUGH POSITION D
AA 0 FF 0 D 0010 0 * XX X X 011110
AA 0 FF 0 D 0010 1 * XX X X 011110
ZZ 0 FF 0 7 0010 1 * 55 X X 011110    CHECK COPY
ZZ 0 FF 0 7 0010 0 * 55 X X 011110
ZZ 0 FF 0 7 0010 1 * 55 X X 011110
AA 0 FF 0 E 0010 1 * XX X X 011110    BIT SET M REGISTER THROUGH POSITION E
AA 0 FF 0 E 0010 0 * XX X X 011110
AA 0 FF 0 E 0010 1 * XX X X 011110
55 0 FF 0 A 0010 1 * XX X X 011110    BIT CLEAR M REGISTER THROUGH POSITION A
55 0 FF 0 A 0010 0 * XX X X 011110
55 0 FF 0 A 0010 1 * XX X X 011110
55 Z FF 0 6 0010 1 * XX 0 X 011110    READ STATUS REGISTER, I = 6
```

Figure 2.2. Beginning of 2914 Logic Integrity Test, including test conditions and first forty vectors

Other factors that the writer must consider are the limitations caused by the technology used to manufacture the device, the packaging, and the type of clocking required. The technology will dictate the type of tests to be performed and such considerations as handling precautions and load capacitance. The packaging may cause changes in Slash Sheet Table III (e.g., a device type with multiple ground pins). This is usually the case when a device is available in both the dual-in-line package and the flat package. Thus, Table III should specify tests with all ground pins connected to ground.

2.3 Test Techniques

The methods of implementing slash sheet tests are as numerous as the test systems used to implement them. In the past, many of the slash sheets were developed with little thought in the implementation of these tests on ATE. However in the last few years, the use of the LIT for setting up the device outputs for measurements has increased. This technique has gained widespread acceptance since it is easily adapted to many types of ATE. As larger and more versatile ATE (S-3260/3270, Sentry, etc) come on line at the various test sites, other test techniques will be developed that implement additional DC and AC tests. However, the limitations of the various ATE should be considered as these techniques are developed.

2.4 Tester Limitations

A great deal of emphasis has been placed on the development and the refinement of algorithms for test program generation, as evidenced by the proliferation of conferences, the literature, and even this effort. With all of the fine theoretical work going on, it is important to realize that test program implementation on actual ATE is the final goal. The limitations on the ATE should always be kept in mind, or else "clever" test vector sets may have to be discarded in favor of those which have the simple virtue of being practically implemented.

There are two related reasons why a section on tester limitations has been included in this report. The first, and most important, is that the state of the art of test systems is such that not all desired operations can be performed. An obvious example would be the storage and delivery of an arbitrarily large number of test patterns at a high test rate. When this is necessary, it is usually done through system field testing, with a resultant lack of repeatability and testing confidence. The second reason for including this section is that tradeoffs have been made between test and tester capability to keep the cost of ATE down. As test writers who have used more than one type of ATE know, no piece of ATE possesses every desirable existing feature. Therefore, it is necessary to intimately know the capability of the machine on which a test is to be implemented. If a test is to be independently implemented on more than one machine, the set of features to be employed must be restricted to the set common to both. The expensive and error-prone manual conversion of data to accomodate different tester capabilities must be avoided.

For the implementation of MIL-M-38510 testing on LSI devices, there are two types of ATE commonly used. The Tektronix S-3260/3270 is used extensively by the government and by DOD contractors. The Fairchild Sentry family (Series V, VII, VIII) is used extensively by IC manufacturers. Although these machines are both high speed, clock rate LSI device testers, they are greatly dissimilar in architecture and operation. This has made the effective transfer of test programs between these systems very difficult. Both systems have desirable features not incorporated in the other. There is a

narrow band of commonality, and all tests designed under this effort were intended to fall in that area. This compatibility between testers has not been completely verified for these tests; only with time can this approach be validated.

Please note that no advertisement or denigration of either of these ATE families of any other family is intended by the authors of this report, the government, or the contractor.

A comparison of tester architectures will highlight the differences between the S-3260/3270 and the Sentry family of testers. It will also indicate some of the good and bad features of both, and thus may indicate to the test writer what features should not be used (sections 2.4.1 and 2.4.2). A discussion of the common features will indicate those used to implement the test programs resulting from this effort (section 2.4.3). The last section will deal only with the S-3260/3270, detailing some of the mundane difficulties encountered while implementing tests (section 2.4.4). It should be noted that a similar section covering the Sentry was not included only because of a lack of firsthand experience on the part of the authors.

2.4.1 Comparison of Architectures

From an architectural point of view, the S-3260/3270 and the Sentry have little in common. A fundamental difference is in the method of applying the test vectors. The S-3260/3270 uses a set of 64 shift registers to store and deliver pattern information to the DUT. The S-3260 has a shift register length of 1032 bits (1K plus 8), while the S-3270 has a shift register length of 4104 bits (4K plus 8). These shift registers contain all of the data and their control (direction, force, compare) information.

The Sentry, on the other hand, uses a fast "local memory" which contains statements for the data and control information. The local memory may be up to 4K bits in depth, and handles up to 60 pins. The various data and control statements are stored in a form which is more nearly a computer program than typical test vectors, although the effect is the same. This local memory is scanned by a controller which then applies the necessary conditions, through the pin electronics, to the DUT.

Although the architectures described so far are significantly different, the method of pattern storage is not intrinsically a great problem. The difficulty lies in modes of operation. The S-3260/3270 is by nature a very straightforward machine. Patterns are loaded into the shift registers, and fired in a burst to the DUT. More patterns are loaded, and burst again. The Sentry, with a local memory, encourages looping, subroutining, and changing local memory contents on the fly. This does not necessarily disadvantage the S-3260/3270, as its linear organization allows the shift registers to be loaded with the error information from the DUT, generating the fault signatures which were necessary to the fault isolation experiments performed during this effort (see section 5). This would have been considerably more difficult on the Sentry, although there is optional hardware for the Sentry which allows the collection of failure data and which could have been used to gather the necessary fault signatures (the Dynamic Fail Module, or DFM). On the other hand, the S-3260/3270 has a Pattern RAM (PRAM) option which possesses many of the features of the Sentry local memory.

2.4.2 Modes of Operation

In spite of the differences in the basic architectures of the two machines mentioned above, test vector implementation with either a shift register or local memory is strictly an implementation detail. The real difficulties occur due to differences in modes of clocking or phasing of data.

The Sentry is capable of delaying data when connecting to a particular phase. The waveform can retain the same shape, but be translated in time. It is unfortunate that on the S-3260/3270, only data that are applied "broadside," on the beginning of the cycle, can be Non-Return-Zero (NRZ). A connection to a phase implies ANDing the clock with the data to form Return-Zero (RZ) waveforms. The data may be explicitly Return-Zero-Inverted (RZI), but the data is still modified during a fixed clock cycle.

The minimum cycle time (period) for the S-3260/3270 is 48ns in "Mode 1" operation, or 96ns in "Mode 3" (these are described shortly). For the Sentry, 100ns is the minimum when using RNG0 (range 0, the fastest range). This is mitigated by Sentry's ability to OR timing generators, which will in some cases allow the data rate or clock rate to be effectively doubled. Using this feature, however, may make it difficult to use patterns developed for a Sentry on an S-3260/3270.

2.4.3 Features Common to the S-3260/3270 and Sentry

The following is a list of parameters which should indicate what is allowed when implementing a test intended to run on both the S-3260/3270 and the Sentry. The goal of this effort was to keep always within these specifications for the LIT and AC parametric tests, but occasionally it has been necessary to develop a unique test, or set of tests, intended for one or the other machine.

As a note, the S-3270 is a more powerful machine than the S-3260, so it was usually necessary to take the lower performance parameters as the limiting factors. As upgrades become more common, it may be possible for these guidelines to be revised.

2.4.3.1 Cycle time

Tektronix: 96ns minimum, 8ns steps (Using Mode 3).

Sentry: 100ns minimum, 10 ns steps.

It is usually convenient to chose a fairly long cycle time, relative to the clocking phases. This will tend to avoid the problems with "dead time" after and before the start of test cycles.

2.4.3.2 Number of Timing Generators

Tektronix (S-3260): 5 for drivers, 2 for comparators.

Tektronix (S-3270): 10 for drivers, 4 for comparators.

Sentry: 6 for drivers, 2 for comparators (strobes).

Although there are two comparator timing generators available, they are almost always used together, and are set to the same timing values.

An additional restriction on the use of the S-3260/3270 clock phases is that not every phase is available at every pin. For example, the S-3260 has 64 sectors, every one of which has available the two comparator phases and DATAPHASE. Of the remaining four timing phases, PHASE 1 is available only at sectors 1-16, PHASE 2 only at sectors 17-32, Phase 3 only at sectors 33-48, and PHASE 4 only at sectors 49-64. This makes it absolutely necessary to use great care in the design of the test fixture (called a socket card assembly), so that the necessary phases are available to the correct DUT pins. For comprehensive AC testing of a device with many pins, pin assignment may become a major problem. The Sentry family is not as limited in the area of clock phase assignment.

It would be desirable to automate the design of the S-3260/3270 fixtures as much as possible, to allow the best possible allocation of sectors during AC testing. Although a set of algorithms were developed during this effort for that purpose, they were never implemented in software, and so are not documented in this report.

2.4.3.3 Timing Generator Capabilities

Tektronix: 8ns minimum width,
1ns steps,
17ns forbidden zone before end of cycle.

Sentry: 10ns minimum width,
0.16ns steps,
10ns forbidden zone after start of cycle,
10ns forbidden zone before end of cycle.

Note that there are several "forbidden" zones, during which times a clock phase edge should not be programmed to occur. This may become troublesome when doing setup and hold tests.

2.4.3.4 Number of pins

Tektronix: 64.

Sentry: 60.

This is a very fuzzy area. The literature is full of misleading figures for test rates as functions of various mixes of input-only pins, output-only pins, and bidirectional pins. It has been found that the number of bidirectional pins handled is the best measure of pin capability.

The Sentry is able to perform the functions of forcing input data, comparing output data, masking output data, and inhibiting input data simultaneously on any pin, or column of the vector set.

The S-3260/3270, on the other hand, when used in Mode 3, can only force and inhibit input data, or compare and mask output data on any particular sector. There is a Mode 4 which can do all four operations simultaneously, at half the data rate and half

the pattern depth. In Mode 3, every bidirectional pin will require two sectors to be allocated to it to achieve full capability. There are other approaches available which provide more bidirectional pins (involving tradeoffs between the number of pins, test speed, pattern depth, reduced capability, etc.) but Mode 3 was deemed to be the most flexible for this effort.

2.4.3.5 Data Modes

Common waveform modes: NRZ, RTZ, RTO (see below).

There are a multitude of possible modes in which data may be applied to a DUT. One common mode is to apply the data broadside to the device, using NRZ, unmodified by any clock phases. This is shown in Figure 2.3. This is a mode available on both the S-3260/3270 and the Sentry.

If a clock phase is used, the result can be Return-To-Zero (RTZ). This is shown in Figure 2.4, with a start time and duration such that the pulse stays within the test cycle. Although it is possible to program a start time and duration such that the pulse falls outside the current test cycle and into the next, this should be avoided as the result may differ on different ATE. In particular, making the pulse width (duration) the same as the cycle time on the S-3260/3270, will not give the effect of only delaying the pulse. This restriction does not hold for the Sentry, but it should be kept in mind when designing tests that may be implemented elsewhere.

To use the RTZ mode, both the Sentry and the S-3260/3270 specify Return-Zero (RZ). Here, the stimulus applied to the DUT pin is zero for any time outside the active portion of the clock phase, and is one if the data is one, zero otherwise.

Corresponding to the RTZ mode is one called Return-To-One (RTO). This is shown in Figure 2.5. The same clocking restrictions on RTZ data hold for RTO data.

To use the RTO mode on the Sentry the pin is programmed to be Return-Zero by the use of the "SET RZ" statement, and then inverted by the use of the "SET INVERT" ("SET I") statement. Note that this does not invert the sense of the data, but rather inverts the "zero" portion of the "SET RZ" statement, making it actually into a "SET RETURN-TO-ONE."

The use of RTO presents a slight conceptualization problem when implemented on the S-3260/3270. It is necessary to have the RTO data stored in the ".PAT" file as the inverse of the desired data, and then applied in the Return-Zero-Inverted (RZI) mode. As all of the test vector generation for this effort was done through the use of ALICE, and was intended to be used on both the S-3260/3270 and Sentry family, it was found to be more convenient to generate the data in the RTO mode, and cause the data to be inverted when it was converted into the ".PAT" format by the TEKTRONIX program (see section 3.1.8).

This exemplifies a lesson learned: it is much easier to let the human do the creative portion of the test generation, and let the computer do the busywork. It is hard to imagine a better demonstration vehicle than the generation of test vectors where certain columns have their sense reversed.

2.4.4 Test Implementation Limitations of the S-3260/3270

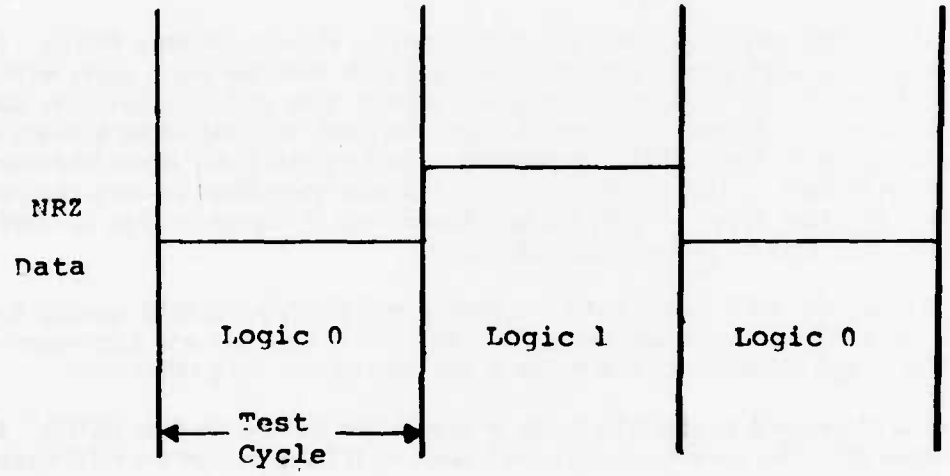


Figure 2.3. Non-Return-Zero (NRZ) Data

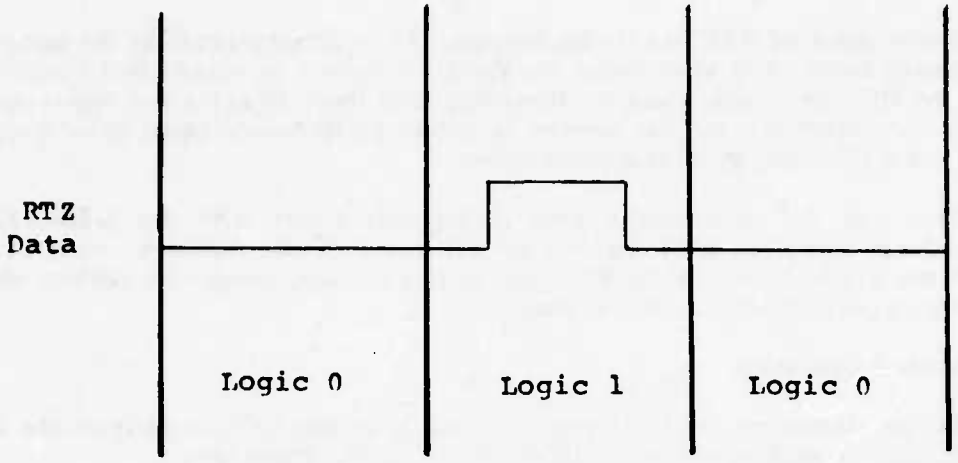


Figure 2.4. Return-To-Zero (RTZ) Data

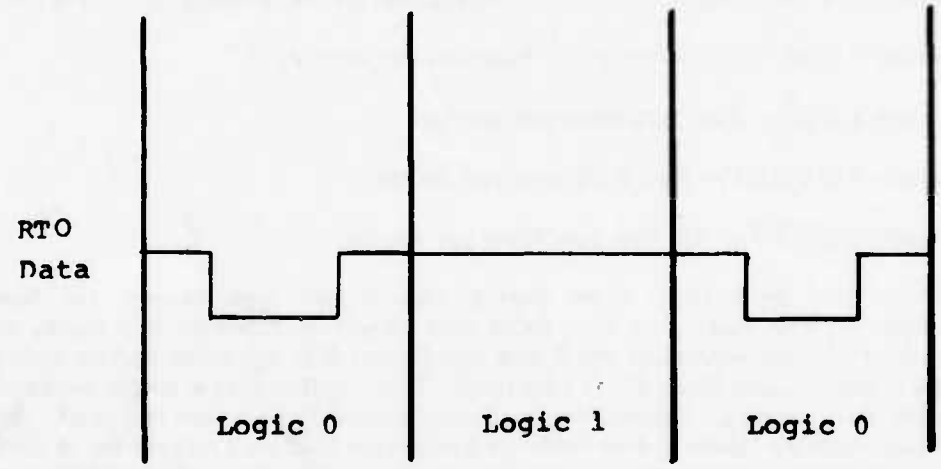


Figure 2.5. Return-To-One (RTO) Data

Every piece of ATE has its limitations. As in signal processing the sampling of a signal should occur at at least twice the signal frequency, so should test equipment outperform the DUT by a wide margin. However, with more complex and higher-speed devices, it is not uncommon for the devices to attain performance equal to or greater than that of the ATE itself, in certain parameters.

This was the case encountered during this effort with the S-3260/3270. Different modes of operation had to be used to test unique device features. Some of the operation of the S-3260/3270 will be discussed in this section, along with options which were used when a performance limit was reached.

2.4.4.1 Mode 3 Operation

Before discussing the test system modes, it is best to first describe the four functions available on each of the 64 available sector cards. These are:

Force (F) -- Force data,

Inhibit (I) -- Do not force data,

Compare (C) -- Compare for expected data,

Mask (M) -- Do not compare for expected data.

There are four available modes of operation on the S-3260/3270. They are:

Mode 1 (F, I, C, M) -- Only one function per sector,

Mode 2 (FC) -- Two functions per sector,

Mode 3 (FI, CM) -- Two functions per sector,

Mode 4 (FICM) -- All four functions per sector.

Most test generation done during this effort was geared for Mode 3 implementation. In this mode, one can force and inhibit or compare and mask, on any sector card. An input pin would use the Force and Inhibit (FI) functions and an output pin would use the Compare and Mask (CM) functions. This implies that a single bidirectional pin will require two sectors. Consequently, two columns of data are required. Special care by the user must be taken in this mode to make sure that an input sector is inhibited when its corresponding output sector is comparing. Likewise, the mask function must be used when forcing. A special utility program was written to perform this function for proper bidirectional pin handling.

With the restriction that the inhibit operation is always the complement of the mask operation, all four functions can be performed on a single sector card. That is, when forcing data, comparisons are masked and when comparing data, force data is inhibited. This capability of the Tektronix systems is referred to as I/O bus switching. Mode 3 with I/O switching can accomodate a single bidirectional pin with a single sector card and one column of data. A utility program to perform bidirectional pin inhibiting and masking is not necessary since these functions are performed automatically by the hardware. In some cases I/O bus switching cannot be used. An example of such a

situation is when several patterns must be applied to a device before it achieves a completely known state. While a device is going through this kind of initialization sequence, it is unknown whether certain pins are inputs, outputs or in the high impedance state. Therefore it is desirable to both inhibit and mask such pins simultaneously. Although I/O bus switching could have been used in many cases on this effort, it was avoided for convenience since the LASAR modeling of bidirectional pins resulted in two independent columns in the pattern. The test systems used all had enough capacity to keep inputs and outputs on different sectors. Had I/O bus switching been used, the utility program mentioned above would have been replaced by one which merged the two columns of patterns from the LASAR output into a single column for the Tektronix systems (in Mode 3 with I/O bus switching a "1" means force with a logical one and mask, a "0" means force with a logical zero and mask, an "H" means compare with a logical one and inhibit and an "L" means compare with a logical zero and inhibit).

A limitation with Mode 3 is that it can only run at 10MHz, with a maximum shift register depth of 516 patterns (S-3260 only). This means that the shift registers must be reloaded more often, losing more error information (see section 2.4.4.3). Also, multiple shift register loading is one of the slowest parts of the testing function in a production environment. There is a trade-off among the number of functions available in terms of number of pins, shift register depth, and speed of pattern application.

In one situation, when trying to test the Divide-By-Six Clock of the 15530 Manchester Encoder/Decoder at 15MHz, the test had to be implemented in a mode other than Mode 3, because of the data rate required. Using Mode 1 (capable of a 20MHz test rate) was the only alternative.

To use Mode 1 requires a choice of action. Since Mode 1 has only one function per sector available to it, it would be necessary to either create another column of data and physically wire another sector to the test fixture adaptor (clearly difficult), or to use a method called "parallel chaining." Parallel chaining uses the sector adjacent to the one called out in the pinlist, in a "piggyback" situation. This method allows the data to be in the same format as in Mode 3 (no additional columns of data are required), but a separate pinlist is needed to declare that the parallel chaining method is to be used. The pinlist is a statement which is compiled with the test program, and contains the necessary information to correspond the pattern columns with the pin assignments that describes the physical construction of the test fixture. The adjacent sectors required for control functions in parallel chaining must be otherwise unused, however one paralleling sector can control a bus of eight or more other consecutive sectors with some degradation in the overall test speed (8ns/sector).

2.4.4.2 Double Pulsing

With the Sentry it is possible to have a double pulse clock mode. This effectively doubles the maximum frequency of the test system. Only one phase (start and duration) can be set to occur in any given cycle period or periods with the S-3260/3270. Pulses can be stretched over several time periods, but the start time will always be referenced to the beginning of a cycle and patterns still change each time period. Thus the S-3260/3270 does not have quite the timing flexibility available on the Sentry systems.

2.4.4.3 Loss of Error Information

A problem encountered when trying to recover the fault signature is the loss of four vectors of failure information per shift register load. This is due to a four cycle delay (one cycle if the complements of the error data are stored) in putting the error information back into the shift register where the patterns were once stored (recirculation of data). To get around this problem, locations were manually chosen in the pattern set as the last vectors of each shift register load, and the outputs were masked for the last four vectors. These positions were chosen, using the information from LASAR, as places which contributed to neither the TCL or to the fault dictionary. For the very last load of the shift register four dummy vectors were tacked on, with arbitrary inputs and masked outputs. This allowed the final four "good" vectors to be flushed out and their data retained. The four dummy vectors are not added to the other load statements because they would change the desired state of the DUT (unless it is purely combinational or the device is completely reinitialized during the next applied vector).

The outputs must be masked for the four chosen vectors because the error flag would be set by a failure but that failure data could not be accessed. By judiciously choosing the break points, any error which would have shown up during those four vectors should also show up elsewhere in the vector set. There is, of course, the possibility that a failure can occur in a place in the vector set which was not predicted by the single-stuck-at fault model used by LASAR, and by bad luck would not be caught.

A further consideration, which complicates the choice of places to mask the outputs, is that all bidirectional pins must be in the input mode. Otherwise, the masking of the pin implies that it must be forced (using the convention built into the TEKTRONIX utility or I/O bus switching hardware), and this could either damage the device or change its state.

2.4.4.4 Timing Tolerance

While implementing the AC timing tests, the limitations of the accuracy of the S-3260/3270 must be kept in mind. One device in particular, the 2914, has very fast setup and hold times, on the order of 10ns. The accuracy of the standard test system could be off by as much as 3.5ns, as the comparator skew can be up to 1.5ns. The measurement of a 10ns timing relationship with a possible error of 3ns was very undesirable. Without the benefit of a faster piece of equipment the best that could be done was to make sure that the timing was adjusted as close to zero skew as possible using tester calibration and verification programs. The practice of guardbanding may become very difficult to implement in cases such as this.

2.4.4.5 Time Measurement System

The Delta-T subsystem on the S-3260/3270 is used to measure times, such as pulse width or propagation time. The time measurement is enabled at the beginning of a specific programmed vector number and then starts and stops when the voltage it is expecting is more than the programmed HICOMPARE or is less than the programmed LOCOMPARE value during the programmed compare times. HICOMPARE is used to detect rising transitions while LOCOMPARE detects falling transitions. The Delta-T subsystem can measure time transitions from a reference to a pin or from pin to pin. There are five ranges for the time measurement system with an accuracy of

+/- ((1% of range) + (1% of reading) + 2ns)

Therefore on the fastest range (100ns) the accuracy would be +/- 3.4ns for a 40ns measurement. Tektronix offers a T.I.M.E. option which would improve that number to about +/- 0.9ns. The T.I.M.E. option uses a software autocalibration technique. It would be convenient if this feature were available for normal DRIVE and COMPARE skew compensation. However, something along these lines would be very difficult. This is because of limitations of the present S-3260/3270 architecture (one phase signal can be used by many sectors).

2.4.4.6 Programming Language Limitations

This section lists a few suggestions concerning system software improvements. It is of course understood that a perfect operating system could not be implemented on a computer the size of the one used in the S-3260/3270, but is is hoped that this section may provide some impetus toward further improvement.

The first major inconvenience encountered was that some large test programs would only translate (compile) using the second terminal's memory partition (Background 1). This meant that for developing and running the test both Foreground (FG) and Background (BG) were required, i.e., most of the test system was occupied with a single task. This is a common occurrence with a large program since more core is allocated to BG1. Having a dynamic core allocation would be much more convenient. An improvement to the operating system that would address this problem is currently under development by Tektronix.

Also, some improvements could be made on the text, pattern, and pin editors. The text editor RESEQuence command could give equal spacing based on the number of lines per section rather than increments of ".1," ".01," ".001," and ".0001." At least increments of ".005" and ".0005" could be added. There also should be a way to have a search and replace command (like the text editor's SAR) to address only the line which a pointer is on, without having to call out the actual line number. Perhaps a line search and replace command (LSAR) in conjunction with existing pointer commands could accomplish this.

It would be convenient to have something comparable to the SAR command available in the Pin Assignment Program (PAP) and the Pattern editor (PEDIT). Of course, the Pattern editor SAR would need to be two dimensional to allow replacement of rows and columns of data.

If System Software were to be completely redesigned, it might be desirable to make a single expanded text editor capable of editing pin assignment, pattern and program source files. Thus only one set of editing command definitions would be necessary.

Similarly, improvements to the test language, TEKTEST, might be considered. Possible improvements include adding structural programming constructs, parameterized subroutine and functions (written in TEKTEST) that can be separately compiled and linked when loaded, provisions for additional data structures include multiple dimensions, additional basic types, (bit, bytes, pattern characters, etc.) and user defined derivative types. Also, getting away from the BASIC-like line numbering, and instead addressing by label, would contribute to accuracy and readability. At present, TEKTEST is similar to FORTRAN and is well suited for implementing test programs. It is well established and efficient in execution but somewhat restricted in provisions for

encouraging consistently prepared, well documented, and efficiently generated test procedures.

3. Computer-Aided Test Implementation Tools

A great deal of time was spent during this effort in the identification of common testing problems. The intention was to isolate the mechanical activities from the creative ones; automate the former, leaving more time for the latter.

Obvious candidates for automation were: the writing of test vectors; editing of pattern files; reduction of test data; and conversion of data from one format to another. Creative tasks were considered to be anything which was non-repetitive, unique, or requiring an action for which no well-defined algorithm could be found. For example, the formulation of a test plan was creative, but the implementation of each task was automated as much as practical.

This section will describe utilities which were implemented on the RADC Multics facility and on the S-3260/3270. The utilities will be outlined in broad detail, rather than in the form of a user's manual. Brief operating guidelines and examples are given in sections 4 and 5. For users who will be generating tests with these, more complete documentation and examples can be made available on request.

3.1 Programs Running on RADC Multics

Test vector operations involve the manipulation of very large sets of data. Problems with vector sets are so unique that at least one Automatic Test System provides a separate editor for their manipulation (S-3260/3270 PEDIT). However, the test writer still faces two major problems: how to generate the vectors and how to interpret the vectors.

This problem is very similar to the one presented to early machine code programmers, and the solution is the same: develop a High Level Language (HLL) and debugging aids.

An HLL for test vector generation has been developed, ALICE, which shares several similarities with true programming languages. There is even the capability to allow the test writer to become independent of a specific device pin list or clocking scheme.

Described here are some utilities that allow the RADC Multics facility to read/write magnetic tapes in the format required by the S-3260/3270. This has accelerated the use of the techniques for manually developing input vectors, and for using systems such as LASAR to generate the expected responses. It should be noted that many of the computer utilities developed or improved as a result of this effort run on the RADC Multics computer. Some utilities were written in text editor language, as that is often the quickest way to do a simple job, but most were written in PL/I. Not completely described here will be utilities dealing with the Sentry magnetic tape formats. Not enough usage has occurred yet to permit these to be completely generalized and debugged.

The following is a list of some of the most important utilities (all written in PL/I). Not listed will be routines which performed a very specialized function, or routines which exist only as support (e.g., I/O) for another utility.

The utilities (in roughly the order in which they would be used) are:

1) ALICE -- "High Level Language" for test vector generation,

2) TECO -- "Assembly Level Language" for test vector generation,

3) DUP -- Removes duplicate test vectors from a pattern set,

4) ALICENUM -- Generates a line number commented ALICE listing,

5) ADD_OUTPUTS -- Allows test writer to manually specify output response and add comments to an input pattern set,

6) REDUND -- Converts a pattern file into one showing only the change of a device pin state,

7) LOADMODULE -- Allows rapid design of device pin loads required during Logic Integrity Test (LIT) and AC parametric tests,

8) TEKTRONIX -- Changes an ALICE-type pattern set into S-3260/3270 Mode 3 format,

9) SENTRY -- Changes an ALICE-type pattern set into Fairchild Sentry "SET F" format,

10) CPAC -- Compares two files, and lists the rows and columns which differ,

11) AANDB -- Performs some set operations on two files,

12) SENPASS, SENPASS2 -- Converts Sentry Application Program and Truth Table into S-3260/3270 ".PAT" files

13) NQUINE, SWAPCOL, PLAGEN -- Help to reduce a boolean function to a PLA model for LASAR,

14) ATE_TAPE_UTIL -- A set of utilities for handling magnetic tapes from the S-3260/3270 and Sentry family.

3.1.1 ALICE

The Automatic LASAR Input Coding Editor (ALICE) accepts data in a convenient "vector" notation and outputs in TECO language (section 3.1.2). ALICE allows pins to be grouped (such as sixteen address lines) under a single variable name, and the test writer may assign values in binary, octal, decimal, or hexadecimal notation. A single statement defines the device pinlist (or pattern columns). Comments may be passed through ALICE to TECO, and then placed on desired pattern lines. An ISSUE statement allows the test writer to specify a subroutine or TECO statement to be executed after each ALICE statement, which allows a clocking or timing scheme to be implemented.

An example is shown in Figure 3.1a.

3.1.2 TECO

```
ASSIGN I=1,2,3,4;HEX
ASSIGN M=5,6,7;OCTAL
ASSIGN EXTERNAL=8,9,10,11,12;DECIMAL
ASSIGN VARIOUS=13,14;BINARY
ASSIGN VARIOUS2=13,14;CBINARY
ISSUE CALL GO
/ THIS COMMENT WOULD BE PRINTED IN TECO
// THIS COMMENT IS LOCAL TO ALICE
// THE FOLLOWING IS NECESSARY FOR TECO:
/PINLIST = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
START:
I=3,M=7,EXTERNAL=21,VARIOUS=00   /THIS COMMENT GOES TO PATGEN
I=2,VARIOUS=01
M=0
VARIOUS2=10
I=C   /LAST STATEMENT
STOP
GO:
/ THIS SUBROUTINE WILL TOGGLE PIN 15, WHICH
/ IS NOT LISTED IN THE ASSIGNS
L 15
H 15
L 15
RETURN
END
```

Figure 3.1a. Sample ALICE Program

The TEst COmpiler (TECO) is a language for test pattern generation. ALICE, the High Level Language, compiles a test pattern file into TECO, which resembles Assembly Level Language. TECO executes the instructions in the new data file, resulting in vectors which are suitable for input to LASAR or (with modification) to any ATE.

This TECO is not the text editor with the same name. It started as a strict subset of the TECO which Digitest Corporation offered on their D4LASAR system. The RADC version has since been enhanced by the addition of many new functions.

An ALICE file contains pure ALICE statements intermixed arbitrarily with TECO statements. ALICE provides the capability of grouping pins for convenient handling, while TECO provides the control structures and special case handling.

An example is shown in Figure 3.1b, and output in Figure 3.1c.

3.1.3 DUP

Occasionally, due to the use of subroutines and do-loops to make ALICE code more homogeneous, duplicate pattern lines will appear in the output data file from TECO. If these are undesirable (as is the case when LASAR will be using the patterns) one may use DUP to remove the duplicate lines.

If two lines are identical except for the presence of a comment on one, the one with the comment is retained.

3.1.4 ALICENUM

When the test writer is debugging or evaluating the test he has written, he generally looks at the response from LASAR or error data from his ATE. When ready to make changes, he should make those changes in the ALICE file, rather than in the actual stimulus data file. ALICENUM provides an ALICE listing with each ALICE statement annotated to show the actual vector number on which it took effect.

An example is shown in Figure 3.1d.

3.1.5 ADD_OUTPUTS

ALICE is primarily used for the generation of input stimulus patterns, rather than the response or output of the device. Although this can be done, it is very difficult if the program flow involves loops or subroutine calls. Usually, the response for large pattern sets is provided by a simulator such as LASAR, or ATE in a "learn" mode.

For small files (usually less than 200 vectors) the program ADD_OUTPUTS can be used to interactively specify outputs and comments. ADD_OUTPUTS is easiest to use when there are only occasional output states to be specified (leaving the bulk of the output states masked). This occurs very often in the design of AC parametric tests.

3.1.6 REDUND

When adapting a LIT or functional test for use as an AC or DC parametric test, it is often necessary to track down rare changes in state of a particular input or

```
  / ASSIGN I=1,2,3,4;HEX
  / ASSIGN M=5,6,7;OCTAL
  / ASSIGN EXTERNAL=8,9,10,11,12;DECIMAL
  / ASSIGN VARIOUS=13,14;BINARY
  / ASSIGN VARIOUS2=13,14;CbINARY
  / ISSUE CALL GO
  / THIS COMMENT WOULD BE PRINTED IN TECO
  /PINLIST=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
START:
  / I=3,M=7,EXTERNAL=21,VARIOUS=00 /THIS COMMENT GOES TO PATGEN
B L 1 2
B H 3 4
B H 5 6 7
B L 9 11
B H 8 10 12
B L 13 14
CALL GO
  / I=2,VARIOUS=01
B L 1 2 4
B H 3
B L 13
B H 14
CALL GO
  / M=0
B L 5 6 7
CALL GO
  / VARIOUS2=10
B LL 14
B HH 13
CALL GO
  / I=C /LAST STATEMENT
B L 3 4
B H 1 2
CALL GO
STOP
GO:
  / THIS SUBROUTINE WILL TOGGLE PIN 15,WHICH
  / IS NOT LISTED IN THE ASSIGNS
L 15
H 15
L 15
RETURN
END
```

Figure 3.1b. Sample TECO Program (output of ALICE)

```
0011111110101000    THIS COMMENT GOES TO PATGEN
0011111110101001
0011111110101000
0010111110101010
0010111110101011
0010111110101010
0010000101010 10
0010000101010 11
0010000101010 10
0010000101010HL0
0010000101010HL1
0010000101010HL0
1100000101010HL0    LAST STATEMENT
1100000101010HL1
1100000101010HL0
```

Figure 3.1c. Sample PATGEN Program (output of TECO)

```
    ASSIGN I=1,2,3,4;HEX
    ASSIGN M=5,6,7;OCTAL
    ASSIGN EXTERNAL=8,9,10,11,12;DECIMAL
    ASSIGN VARIOUS=13,14;BINARY
    ASSIGN VARIOUS2=13,14;CBINARY
    ISSUE CALL GO
    / THIS COMMENT WOULD BE PRINTED IN TECO
    // THIS COMMENT IS LOCAL TO ALICE
    // THE FOLLOWING IS NECESSARY FOR TECO:
    /PINLIST = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
    START:
 1  I=3,M=7,EXTERNAL=21,VARIOUS=00   /THIS COMMENT GOES TO PATGEN
 4  I=2,VARIOUS=01
 7  M=0
10  VARIOUS2=10
13  I=C   /LAST STATEMENT
    STOP
    GO:
    / THIS SUBROUTINE WILL TOGGLE PIN 15, WHICH
    / IS NOT LISTED IN THE ASSIGNS
    L 15
    H 15
    L 15
    RETURN
    END
```

Figure 3.1d. Output of ALICENUM

output pin. REDUND allows the user to create a new listing from the LIT file, showing the state of pins only when they change, replacing the other occurances with some innocuous character of the user's choice. Thus, a quick visual scan is less likely to miss difficult-to-detect transitions of single or multiple pins.

An example is shown if Figure 3.1e.

3.1.7 LOADMODULE

During the LIT and AC parametric test the DUT is generally run with the outputs loaded. The load, shown in Figure 2.1, is designed to force the DUT output pins to source current at the maximum IOH when the output pins are at minimum VOH; to sink maximum IOL when at maximum VOL; to float to some specified intermediate voltage (VTSC) when the pin is in a high-impedance state.

LOADMODULE calculates two resistor values (RL, R1) and a pullup voltage (VP) when given the desired IOH, IOL, VOH, VOL, and VTSC. Since the resistor values yielded are standard values, LOADMODULE for convenience also calculates the actual expected parameters, which are generally only a few percent from the desired values.

3.1.8 TEKTRONIX

ALICE generates data (through TECO and DUP) which are directly compatible with LASAR. LASAR simulates the DUT, and generates the response data. Unfortunately, no digital simulation packages yet available can adequately and realistically handle three-state, or worse, bidirectional pins. These mechanisms require elaborate workarounds based on a thorough knowledge of the DUT's actual behavior.

The usual workaround for a three-state pin is to use a "Don't Know" generator in the model to specify "X" as the output to signify the high impedance condition. A bidirectional pin is modeled as two pins (input and output), and may involve the use of the Don't Know generator to denote that the pin is in the input state.

The TEKTRONIX utility accepts data in the form provided by LASAR (or manually-generated data) and converts that into S-3260/3270 Mode 3 data (using the symbols 0, 1, L, H), reserving two columns per bidirectional pin.

When test vectors have been developed which require one or more pins to be applied in the "RTO" mode (see section 2.4.3.5) it will be necessary for the columns corresponding to those pins to be inverted. This is done when TEKTRONIX is invoked, by listing those pin numbers as arguments.

3.1.9 SENTRY

The SENTRY utility accepts the same data as TEKTRONIX, but changes it into Sentry "SET F" statements. Bidirectional or masked data is handled by use of the MA, MB, DA, and DB registers, which are set or enabled as needed.

3.1.10 CPAC

When debugging a LASAR model or performing fault signature analysis on Multics, it is often necessary to compare two large files of ones and zeros and note the

```
001111110101000
^^^^^^^^^^^^^^1
^^^^^^^^^^^^^^0
^^^0^^^^^^^^^^1^
^^^^^^^^^^^^^^1
^^^^^^^^^^^^^^0
^^^^000^^^^^^^^
^^^^^^^^^^^^^^1
^^^^^^^^^^^^^^0
^^^^^^^^^^^HL^
^^^^^^^^^^^^^^1
^^^^^^^^^^^^^^0
110^^^^^^^^^^^^
^^^^^^^^^^^^^^1
^^^^^^^^^^^^^^0
```

Figure 3.1e. Output of REDUND

differences. As this is obviously a case where the computer is more capable than the human, the CPAC (Compare ASCII Columns) utility was developed. The output is a list of bits which differ, in the form of "ROW,COLUMN."

3.1.11 AANDB

This utility performs three set operations on a pair of files. If the files were indeed named "A" and "B," this would result in three files, containing the elements which are:

-- In A and in B,

-- In A but not in B,

-- Not in A but in B.

The only other case, "in A or in B," is directly obtained from the use of two commands which are already available on Multics (concatenate, then sort while retaining only unique elements).

This utility is especially useful when data must be "sieved," as in the case where a LASAR fault dictionary specifies a list of significant bits (see section 5.5.1) and requires that any other bits in a fault signature be struck from it. Here, the "in A and in B" case is useful.

To compare two large fault signatures to see if one differs from the other only by the addition or deletion of bits, the "in A but not in B" and "not in A but in B" cases are employed.

3.1.12 SENPASS, SENPASS2

Due to fundamental differences between the S-3260/3270 and Sentry family, it is impossible to automatically convert any arbitrary complete Sentry program into an S-3260/3270 program. However, it is possible to perform the simpler job of converting only the test vector information in an automatic fashion.

This process is done in two passes. First, the portion of the Sentry "Application Program" dealing solely with the actual test vectors is extracted. This consists of the "Truth Table" and the FACTOR statements that cause the Truth Table to be scanned. This is input to an interpreter which implements a subset of the FACTOR language (SENPASS). The output is a set of vectors, one column per pin, with the functions of input, output, bidirectionality, and three-state being denoted by the characters 0, 1, L, H, Z, X.

The second pass is done by a program (SENPASS2) which breaks the bidirectional pins into two columns, and reorders the pins in a more convenient order (previously they were in ascending pin number order). The test vectors are now in S-3260/3270 Mode 3 form.

If double clocking or other non-S-3260/3270 features are used, it is necessary that the person supervising the conversion to find a way around them. It is planned to implement a utility that will be able to "unwind" any arbitrary clocking scheme, and

generate the appropriate broadside vectors. At present, this process is manual and extremely difficult for large vector sets.

3.1.13 NQUINE, SWAPCOL, PLAGEN

A boolean function may be expressed in the form of "minterms," or the list of every binary code which, when applied to the inputs of a combinatorial network, results in a logical 1 (or conversely, a logical 0 if more convenient). The utility NQUINE performs the sort of reduction possible with Karnaugh Maps [1] or Veitch Diagrams [2], using a variation of the Quine-McCluskey Method [3]. This results in a list of "prime implicants" which cover the function. This can be done for a network with up to 144 inputs. Of course, execution time increases dramatically as the number of inputs and minterms increases. However, this has proved useful in the quick design of decoding logic.

The utility SWAPCOL is mentioned only for completeness. NQUINE has a mode where only partial reductions are made, i.e., the resulting implicants are not necessarily prime [4]. Here, each minterm in the original function becomes part of one and only one implicant. This speeds evaluation by many orders of magnitude, but yields a result which is not nearly as good as the complete reduction. SWAPCOL, which can scramble the columns in a directed manner, can allow the NQUINE "partial mode" to do a much more complete reduction. The cost saving is so great that ten or so runs can be made with random swapping, the best result chosen, and "unswapping" performed to recover the minimized original function.

The PLAGEN utility is used to automatically generate a LASAR model that implements, in NAND gates, the function described by the output of NQUINE. This is in keeping with the policy during this effort of making purely mechanical jobs the responsibility of the computer, to minimize human error.

A final note on the NQUINE utility is necessary. In the mode where prime implicants are the result, there will almost always be multiple coverage of minterms. This results in three types of prime implicants: superfluous or "absolutely eliminable" prime implicants, which should be discarded; essential prime implicants, whose deletion changes the function, and should be identified for retention; non-essential prime implicants, some combinations of which can be deleted. A utility was developed for this, called PETRICK, which uses instead of Petrick's Method [5], a more computationally efficient algorithm However, it was found that even halving the final number of prime implicants was hardly worth the effort of running PETRICK.

3.1.14 ATE_TAPE_UTIL

This is a set of programs, called PEDIT, PCREATE, EEDIT, ECREATE, PWRITE, ECHECK, READ_TAPE_NSTD, and SENFROM. These allow the RADC Multics facility to perform magnetic tape transfers of information with the S-3260/3270 and, in a limited fashion, with the Sentry family. Those utilities are forced to perform rather circuitous exercises in bit stuffing to read and write in the internal formats of the other machines. Of all of the utilities described here, these are certainly the least transportable.

3.2 Programs Running on the S-3260/3270

The proliferation of incompatible automated test equipment has slowed the evaluation of new devices. Device test programs written for one ATE will seldom run properly on another ATE. In some cases, these programs require extensive modification before they can be run on similar equipment. The following programs were developed to reduce this problem and to assist with the evaluation of the test and results.

The utilities are:

1) SIMPL -- Test implementation language,

2) MATSIM -- Aid to SIMPL programming,

3) SIMTEK -- Creates a ".TST" file that executes SIMPL commands,

4) C0DE -- Sets key for data logging,

5) BATLOG -- Assign LUN's for data logging,

6) BATRED -- Assigns LUN's for DATRDB,

7) DATRDB -- Displays LIT failures from log files,

8) PATPLO.ASC -- Assigns initial LUN's for plotting waveforms,

9) PATPLO.EDT -- Assigns LUN to pattern file selected by user,

10) WAVE -- Plots waveforms from above pattern file,

11) FISO.ASC -- Assigns LUN's for fault isolation,

12) FISOV1 -- Clears LUN's assigned by BATLOG,

13) FISO.EDT -- Isolates component that caused a LIT failure,

14) TABPRT -- Assigns LUN's for fault dictionary operations,

15) XYZPRT -- Prints or reformats fault dictionary,

16) RESIS -- Calculates load module parameters.

3.2.1 SIMPL Language

The SIMPL language was developed as a first attempt to simplify test implementation on the S-3260/3270. All standard DC tests, the LIT, and some AC tests have been implemented. Special functional tests have not been included at this time.

3.2.2 MATSIM Program

Generation of a device test using the SIMPL opcodes can be accomplished by referring to the Opcode Reference Chart. However, the MATSIM program was developed to assist the user in specifying the various arguments required for each opcode.

Future changes to this program will allow English Language statements to be translated into the appropriate SIMPL statements. Also included should be "checker" routines which will verify that any particular SIMPL device program is valid and will meet a particular test station configuration.

3.2.3 SIMTEK Program

The SIMTEK Program is a series of files that, when combined with a particular device pinlist/pinarray, results in a S-3260/3270 ".TST" program capable of executing the SIMPL opcodes. The program reads a SIMPL statement, executes it, then reads the next one, continuing until the opcode FINIS is read. LIT error information is saved in the disk files ERRPAT.LOG and CONTRL.LOG for later failure analysis.

The complete program, as it exists now, requires a large amount of system memory (approximately 13K). For small tests this may be excessive. However, LSI and VLSI devices presently require larger tests when written in TEKTEST. When these device tests are implemented in SIMPL, the overall test size will generally be lower. Changes in the current operating system are anticipated that will help in this area.

Recommended changes to a SIMPL Interpreter are:

-- The elimination of the ".PIN" file, pinlist and pinarray files from the main SIMTEK program,

-- The above files should be stored in a special device test file that can be accessed to by the main test routine,

-- Device test pattern files should be separated from the test file and be referenced by the main test program.

These changes will result in a few small files that describe the DUT test and one main test file for all devices. This would make better use of the system disk and computer memory and would eliminate much of the file manipulation that is currently required.

3.2.4 C0DE Routine

The C0DE Routine requests a key for the ".LOG" files from the operator and stores it for use during testing. The routine also illuminates the lights on the TSCU for the operator to verify before beginning the test.

As an aside, the name C0DE is spelled with a zero, rather than the letter "O," to allow the user to dial the name on the hexadecimal switches of the TSCU.

3.2.5 BATLOG Routine

The batch log file (BATLOG) is a set of instructions that direct the system "LOG" program to assign the LUN's needed by the DUT test program. After all tests have been completed it closes the various LUN's and reassigns them to the keyboard.

3.2.6 BATRED Routine

The batch reduce file (BATRED) is a set of instructions that direct the system "REDUCE" program to assign the LUN's needed by the DATRDB program and then execute DATRDB.

3.2.7 DATRDB Program

The data reduction (DATRDB) program queries the operator for additional information, such as whether the line printer should be used for the output, or to accept the key of the log file data to be reduced. The device name, pin names, and LIT error data are read from log files created by SIMTEK (see section 3.2.3) and presented to the operator in the requested format. The program then queries the operator on whether it should rerun with the same data, rerun with new data or stop.

The output format may be one of the following three choices:

-- Number of failures per DUT output pin,

-- Contents of the log file containing the LIT errors,

-- Row and column numbers for each failed bit.

3.2.8 PATPLO.ASC Routine

The pattern plot ASCII file (PATPLO.ASC) is a set of instructions for the REDUCE program. These direct the creation of temporary disk files that are used by PATPLO.EDT, which is then automatically executed.

Note that the programs PATPLO.ASC, PATPLO.EDT and WAVE.EDT were written by Tektronix, Inc. The WAVE.EDT program has been extensively modified to meet the needs of this effort.

3.2.9 PATPLO.EDT Program

This program asks the operator for the name of the pattern file to be plotted. The name along with other LUN assignments are placed into one of the temporary files created by PATPLO.ASC. Then the REDUCE program is directed to execute these assignments. This automatically runs the WAVE.EDT program.

3.2.10 WAVE Program

The wave form plotting program (WAVE.EDT) queries the operator for the portion of the pattern file to be plotted. It then reads the file and displays a pseudo timing diagram on the CRT that reflects the levels that would be forced on inputs or expected on outputs during testing on the S-3260/3270. The wave forms are not true data because no timing terms or data inversions are included. Also, Mode 3 is assumed.

3.2.11 FISO.ASC Routine

The fault isolation ASCII file (FISO.ASC) is a set of instructions for the REDUCE program. This set prompts the operator to ensure that the line printer is on. Then the LOG program is directed to execute the instructions in the ISOV1 file (see section 3.2.12). When LOG returns control to REDUCE the logical units are assigned as

needed by FISO.EDT and displayed for the operator to verify. When the operator types a line feed, the FISO.EDT program is executed.

3.2.12 FISOVI Routine

This file contains the instructions needed by the LOG program to close the files used by the SIMTEK program and clear all LUN assignments. This operation ensures that FISO.EDT can proceed without interference.

Note that this operation is entirely transparent to the operator unless an error condition is encountered.

3.2.13 FISO.EDT Program

The fault isolation (FISO) program queries the operator for his output device choice and the key of the log file that contains the LIT error data from SIMTEK. The algorithm described in section 5.5 is then performed and the results displayed. The operator is asked to indicate whether the program should rerun with the same key, rerun with a new key, or stop. The following paragraphs will describe the TEKTEST implementation of the algorithm more fully.

The number of output pins, the maximum number of elements in each Z-entry (called DYSOGN), the number bits in the XTABLE, and the number of entries in the ZTABLE are read from the XTABLE, along with the device name and the date of the LASAR run. The LIT errors contained in the log files from SIMTEK are compared to the XTABLE entries, with the relative position of matches stored in an array. This is the error signature, "S" (called XCODE in the program).

The contents of S are compared with each entry of the ZTABLE. Each time a match is found, a counter (called INTER) is incremented. When the end of the ZTABLE entry or the signature is reached, INTER contains the number of elements in the intersection (SZ) of the S and Z. The number of ZTABLE entries (ZNUM) is then compared to DYSOGN. If the two are equal, the actual number of elements (ETAB) used from S in the first comparison is used to calculate the goodness of fit (RANK). Otherwise, the total number of S entries is used. The equation for the first case is:

$$\text{RANK} = \frac{\text{INTER}}{(\text{ETAB} + \text{ZNUM} - \text{INTER})}$$

In the second case, ETAB is set equal to the number of elements is S and then the above equation is used. The RANK is stored for later ordering.

If RANK equals 1, the signature to ZTABLE comparison is terminated, and the corresponding YTABM entry is printed. Otherwise, the comparison continues until the ZTABLE is exhausted. Then the ten highest RANKs are extracted from storage and the related YTABM entries are printed. If the RANK to be printed is less than 0.1, the printing stops. Also, the printing continues beyond ten entries if the following RANKs are equal to the RANK of the tenth entry. Thus the circuit component(s) that caused the LIT failure is isolated.

The YTABM file is the YTABLE reformated to a more easily read format (see the descriptions of the TABPRT and XYZPRT programs in sections 3.2.14 and 3.2.15).

3.2.14 TABPRT

The table print file (TABPRT) is a set of instructions for the REDUCE program. This set directs the creation of a disk file called YTABM.ASC and the assignment of the LUN's needed by the XYZPRT program, which is then automatically executed. If the file YTABM already exists, an error message is displayed and the data that would have been stored in YTABM is sent to the CRT.

3.2.15 XYZPRT

This program asks the operator for the output destination, and whether the XTABLE, YTABLE or ZTABLE is desired. If the YTABLE is selected, the operator may print it, store a reformated version in YTABM.ASC or both. The XTABLE and the ZTABLE files are printed only.

The YTABM.ASC file must be created before the fault isolation routine FISO.EDT (see section 3.2.12) will perform correctly.

3.2.16 RESIS

This program performs the same function as the MULTICS LOADMODULE program (see section 3.1.7). RESIS selects the two load resistors and the load power supply voltage that result in the lowest error in IOH and IOL. The resistors are selected from the table of standard five-percent values.

References:

1. Donald D. Givone, Introduction To Switching Circuit Theory, McGraw-Hill, 1970, pp. 110-136.

2. Thomas R. Blakeslee, Digital Design With Standard MSI & LSI, John Wiley & Sons, 1975, pp. 40-49.

3. Givone, op. cit., pp. 136-144.

4. Sheldon B. Akers, "On the Specification of and Analysis of Large Digital Functions," Proceedings of the Seventh Annual International Conference on Fault-Tolerant Computing (FTCS-7), pp. 88-93.

The footnote on page 91 is of special interest.

4. Test Development and Implementation

4.1 Procedure For LSI Functional Test Development

This section describes the approach which is used for the evaluation of the functional tests. The steps outlined here indicate what sort of tests should be applied to a DUT. These provide a rough set of recipes for test generation, but were originally designed for cases where a test was submitted and had to be verified for completeness.

4.1.1 The approach consists of the following steps:

4.1.1.1 Generate a detailed functional block diagram by partitioning the LSI device into basic functional blocks such as registers, data selectors, arithmetic and logic functions. Identify all data paths.

4.1.1.2 Test each of the basic functional blocks using proven test patterns which result in a high TCL. In some cases, these blocks can be exhaustively tested with few vectors.

4.1.1.3 Generate test patterns to verify the integrity of the data and control paths.

4.1.1.4 Verify that all instructions perform the specified operations.

4.1.1.5 Include test patterns that check for known processor sensitivities. This may include vendor and user-supplied data.

4.1.2 A gate-level diagram, timing diagram, Automaton Diagram, and a block diagram, showing the major functional areas of the microprocessor and the interconnecting data and control paths, would be valuable for test development/evaluation.

However, in many instances only a timing diagram and an insufficiently detailed block diagram are available. It is then necessary to develop a detailed functional block diagram or Automaton Diagram based upon the best available information and verify its accuracy with the manufacturer.

Once this is done, verification that each of the functional blocks is tested, using proven test patterns, must be done. Following is a list of the types of tests required:

1) Verify the independence of each IC pin. This is accomplished by checking that each pin assumes a "1" and a "0" state while all of the other pins, either individually or collectively, are in the complementary state. The states of input pins have to be monitored unless they are sensitized to the outputs.

2) Verify that the control lines perform the intended functions and perform independently of the previous instruction. This is accomplished by activating each control line and checking that the address, data, and status lines assume the correct states. Independence is verified by activating the control lines in a gallop type test and checking the response.

3) Check that the proper priority is maintained when two or more control signals are applied at the same time.

4) Verify the high impedance capability of the applicable data and status lines by placing them in the high impedance state and measuring the leakage current. This should be done with the input of the three-state buffer driven to both a "1" and a "0."

5) Verify the independence of each line in a data path and that each line can pass a "1" and a "0." This is accomplished by checking that each line assumes a "1" and a "0" state while the others, either individually or collectively, are in the complementary state.

6) Verify that any identifiable multiplexer can pass both a "1" and a "0" in each selection position.

7) For serial arithmetic adders/subtractors with unknown mechanizations, apply all possible inputs (0 & 0, 0 & 1, 1 & 0, 1 & 1) to each input pair with the carry equal to "0" and then with the carry equal to "1." This should be done in both the add and subtract modes. The carry is either the external carry into the adder or the carry from the next least significant bit. If the subtractor is a 1's or 2's complement subtractor, these tests will also verify that the inverting circuitry can invert both a "1" and a "0."

8) For logic operations apply the following input conditions to each input pair of the ALU:

   0 & 0, 0 & 1, 1 & 0, 1 & 1 during EXOR operations,

   0 & 0, 0 & 1, 1 & 0 during OR operations,

   0 & 1, 1 & 0, 1 & 1 during AND operations.

9) For shift left and shift right operations, verify the shift of the 0 to 0, 0 to 1, 1 to 1, and 1 to 0 transitions in each direction. Use data patterns that will ensure that a shift in the wrong direction will be detected.

10) Check all flip-flops for 0 to 0, 0 to 1, 1 to 1, and 1 to 0 transitions.

11) Verify that the carry-in has no effect on the ALU during logic functions.

12) Verify that special outputs such as carry, carry generate, carry propagate, overflow, etc. from an ALU operate properly, i.e., can assume both a "1" and a "0" state and that they occur at the proper time. If equations are provided for their generation, verify that each of the terms in the equations affects the outputs.

13) Partially verify the instruction set by performing each instruction or opcode at least once. Checking that only the intended instruction is performed verifies that the decode circuitry is functioning properly.

14) Verify register independence, bit independence, and integrity of unique registers such as accumulators, stack pointers, index counters, storage registers, etc. Register independence is verified by writing into one of the registers and checking that the others are not affected. Bit independence is shown by having each bit in the "0" and "1" state with all other bits in the complementary state. Integrity is verified by ensuring that transitions from 0 to 0, 0 to 1, 1 to 1, and 1 to 0 are possible for each bit of each static register.

15) If the device contains a RAM, or the unique registers are configured as a RAM, the following items should be checked as a minimum:

15.1) Address Uniqueness

Verify that there are "W" independent word locations in a "W" word memory or that the unique registers are independent. This can be accomplished by writing into an address or register and verifying that it was the only address or register affected. The standard RAM tests which can be used for this are Walking-one, Walking-zero, Galloping-one, Galloping-zero, or Write Recovery.

15.2) Bit Independence

Show the independence of each bit in the "0" and "1" state with respect to all other bits which are in the complementary state. This can best be accomplished with the standard Walking-one, Walking-zero test.

15.3) Cell Integrity

Verify that transitions from 0 to 0, 0 to 1, 1 to 1, and 1 to 0 are possible for all bits. This can be accomplished with the Walking-one, Walking-zero test on RAM's that are more than one bit per word. For single-bit RAM's, separate tests have to be performed for the 0 to 0 and 1 to 1 transitions.

15.4) Intercell Disturbance

Maximize the number of internal transitions to test for cell-to-cell interaction. This can be accomplished with the standard Galloping-one, Galloping-zero test.

15.5) Data Retention

This test is primarily for Dynamic RAM's and verifies the data written in the memory cells are unaffected by the delay between refresh cycles. Until recently, the data retention test was only performed on dynamic memories. However, static memories have also been found to have problems retaining data. Thus, the Static Hold test was developed for these parts. It consists of writing a data bit in memory and waiting before reading the data.

15.6) Write Recovery

Verify that transitions from a write to read do not cause access time failures. This is accomplished by checking all possible transitions from a write to a read.

15.7) Read Modify Write Recovery

Verify that transitions from a read to a write do not cause incorrect information to be written into the device. This is accomplished by checking all possible transitions from a read to a write.

15.8) Sense Amplifier Recovery

Tests should be included to ensure that sense amplifiers respond properly to a complementary bit after repeated reads of like data.

15.9) Sense Amplifier Response

Test for sense amplifier frequency response by repeatedly reading 0/1 data patterns at the minimum read cycle time.

16) Check dynamic registers and buses for proper operation. The test should verify that sufficient charge is transferred in the minimum transfer time and that sufficient charge is available after the maximum storage time. This is accomplished by varying the power supply voltages, and clock amplitudes, periods, widths, and delays to set up the worse case conditions described above.

4.2 Writing a LIT using ALICE and LASAR

All of the manually-developed tests written and used in this effort were generated using ALICE, that is to say, all of the LIT vector sets and AC parametric tests were originally in the form of an ALICE program on the RADC Multics computer. In the case of the AC parametric tests, the outputs generally have to be supplied manually, and then transported to the S-3260/3270, or whatever ATE is the target machine. Those outputs may be generated by the ALICE program, as are the inputs, but it was found to be easier to use the simple-minded editor ADD_OUTPUTS (see section 2.1.5).

When doing the LIT generation, the outputs are generally specified by means of a simulator. Experiments were performed with the use of PL/1 programs to supply the response for a given input set in lieu of a true "functional simulator" (since no such off-the-shelf packages were available). However it was easier in every case to use the LASAR package.

Available under this contract was computer time on an SMC minicomputer running LASAR. The version used was D4LASAR, which later became S1LASAR. GEOS, with permission of the Navy, aided RADC in the acquisition of the Navy-owned D2LASAR. This package was installed on the RADC GCOS facility. This allowed models to be debugged in-house, and taken to GEOS's facility for the more involved test vector evaluation and fault dictionary generation.

As an aside, a few months before the end of this contract a change in the GCOS operating system made D2LASAR unusable. Due to extraordinary efforts by the Computer Facility support personnel the D2LASAR package was quickly installed on the RADC Multics facility, running within the simulated GCOS environment. This actually made the model and vector generation easier as the rest of the utilities and data bases were also resident on Multics.

4.2.1 Developing the ALICE Program

The ALICE program is a shorthand representation of the conditions the test writer wishes to apply to the DUT. The items to be tested are described in section 4.1, although complete recipes are not given.

An example will now be shown. Figure 4.1a is a partial Automaton Diagram of a hypothetical device. The block labeled simply, "random logic," is the portion to which the recipe (shown in Figure 4.1b) is to be applied. Using the partial pinout information shown in Figure 4.1c, the ALICE program in Figure 4.1d is generated.

Note that the preamble, defining the vectors of pins, and a pair of subroutines called "EXECUTE" and "CLOCK" occupy the bulk of the ALICE program, while the actual testing takes only a few lines. This is because of the brevity of this example. The economy of this approach becomes enormous when performing even a few more tests than in this example.

Note also the use of two variables, "TEMPA" and "TEMPB." These are dummy pin vectors, which the user loads with data which are used in subroutine EXECUTE. These data are eventually applied to the D-input pins at the proper times.

After the first few lines, which merely set up the initial conditions, the sequence is as follows:

1) The user sets TEMPA and TEMPB with the data eventually to be applied to the A- and B-inputs of the logic block under test. Subroutine EXECUTE is automatically called, because of the ISSUE statement. The ISSUE statement names an action to be taken after every ALICE statement containing an assignment operator (the equal sign, "="), unless the issuance is suppressed by placing a semicolon at the end of the ALICE statement.

2) Within EXECUTE, the contents of TEMPA are copied to the D-input pins. XR1 is set to allow R1 to be loaded, and subroutine CLOCK is called, causing a positive pulse on pin 18 (CP) to load R1.

3) The contents of TEMPB are copied to the D-input pins, XR1 is set to the copy position (to preserve R1) and XR2 is set to allow R2 to be loaded. The clock is pulsed, causing R2 to be loaded.

4) XR2 is set to the copy position (which is not really necessary) and the clock is pulsed once more. At this time, the contents of R3 are now loaded with the output of the logic block, and the Y-outputs are the desired response to this test cycle.

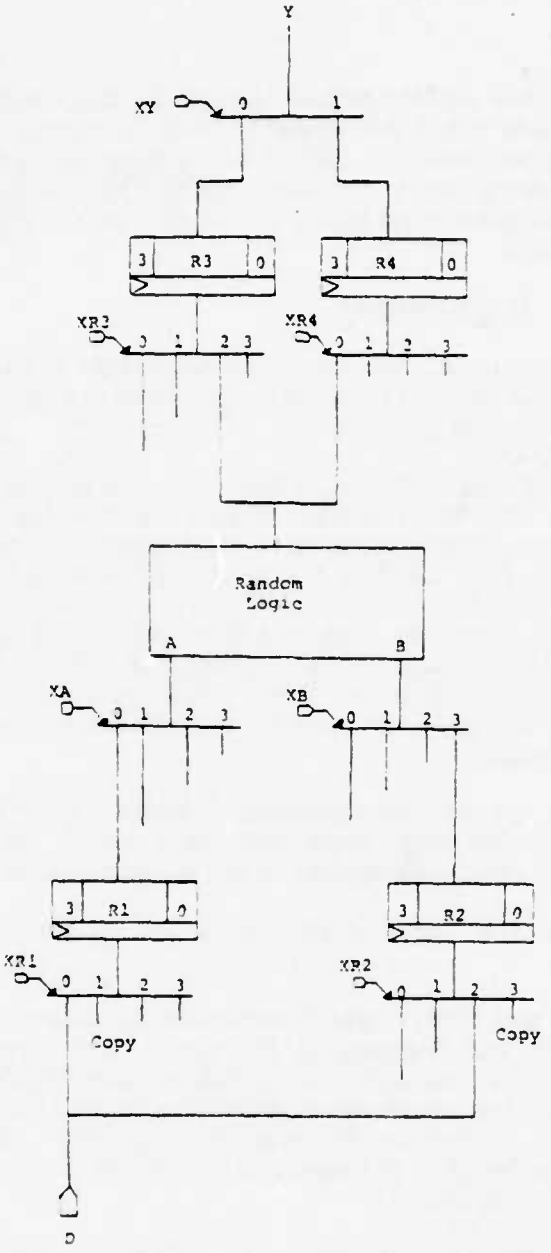


Figure 4.1a. Hypothetical Automaton Diagram

A	B
0000	0101
0001	0111
0011	1001
0010	1010
0110	1011
0100	1101
1100	1110
1000	1111

Figure 4.1b. Hypothetical 100% test of random logic block in hypothetical DUT (inputs only)

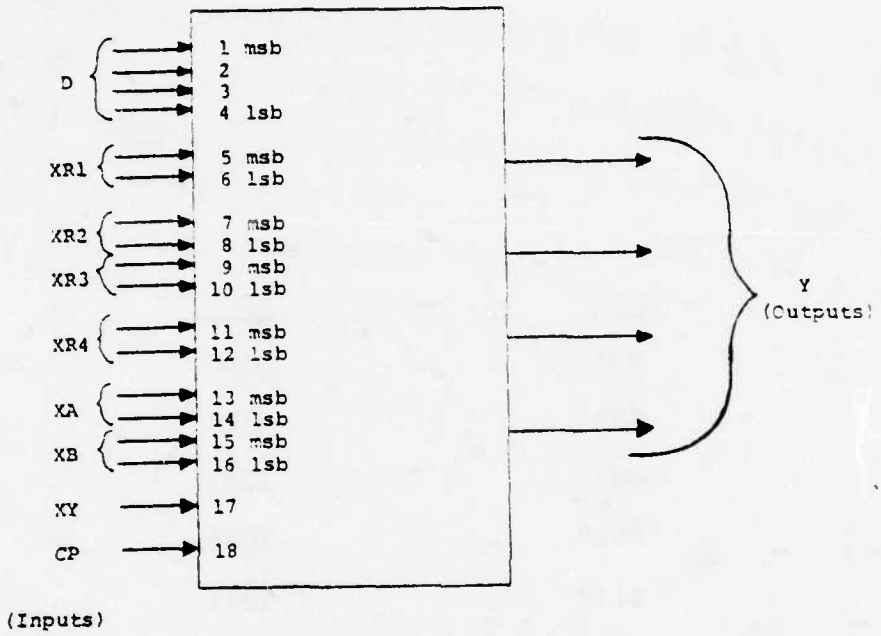


Figure 4.1c. Hypothetical DUT

```
ASSIGN D = 1, 2, 3, 4; HEX
ASSIGN XR1 = 5, 6; DECIMAL
ASSIGN XR2 = 7, 8; DECIMAL
ASSIGN XR3 = 9, 10; DECIMAL
ASSIGN XR4 = 11, 12; DECIMAL
ASSIGN XA = 13, 14; DECIMAL
ASSIGN XB = 15, 16; DECIMAL
ASSIGN XY = 17; BINARY
ASSIGN CP = 18; BINARY
ASSIGN TEMPA = 100, 101, 102, 103; HEX
ASSIGN TEMPB = 104, 105, 106, 107; HEX
ISSUE CALL EXECUTE
/ PINLIST = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
/ CONTINUE 11, 12, 13, 14, 15, 16, 17, 18
/ INITIALIZE EVERYTHING, AND SET UP CONSTANT DATA
/ WHICH WILL BE XA, XB, XR3, XR4, XY
XY = 0, XR3 = 2, XR4 = 3, XA = 0, XB = 3, XR1 = 0, XR2 = 2, D = 0;
TEMPA = 0, TEMPB = 0, CP = 0;
APPLY
/ NOW EXECUTE DESIRED SEQUENCE
TEMPA = 0, TEMPB = 9
TEMPA = 1, TEMPB = 7
TEMPA = 3, TEMPB = 5
TEMPA = 2, TEMPB = A
TEMPA = 6, TEMPB = 8
TEMPA = 4, TEMPB = D
TEMPA = C, TEMPB = E
TEMPA = B, TEMPB = F
STOP
CLOCK:
L 18
H 18
L 18
RETURN
EXECUTE:
/ THIS SUBROUTINE DOES THE TEST APPLICATION
/ TO BURIED LOGIC
/ LOAD R1 WITH TEMPA
S COPY 1 100 2 101 3 102 4 103
XR1 = 0;
CALL CLOCK
/ LOAD R2 WITH TEMPB
S COPY 1 104 2 105 3 106 4 107
XR1 = 1, XR2 = 2;
CALL CLOCK
XR2 = 3;
CALL CLOCK
/ THE OUTPUT IS NOW VALID
RETURN
END
```

Figure 4.1d. ALICE program that implements the test in Figure 4.1b

5) The subroutine returns, and the cycle repeats until done.

The input vectors resulting from this are shown in Figure 4.1e.

4.2.2 Running LASAR

Once the input vectors have been prepared using ALICE, the next step is to generate the responses. When using the D2LASAR package on Multics, the interface problem was slight, because there were system utilities to convert the data into and out of the format required by the GCOS environment.

When using D4LASAR or S1LASAR, it was necessary to use a medium other than a disk file, since the package was running on a different computer. Here, cards or magnetic tape proved to be the best (indeed, the only) routes for input. The output data were returned via magnetic tape only.

The outputs were loaded onto Multics, and converted for use on either ATE. That process is discussed in section 4.2.3.

The difficulty in interfacing with LASAR was not in the limitations of the medium, but rather in the necessity for somewhat elaborate model workarounds. These problems are not unique to LASAR, but plague every designer and user of present-day simulation packages. This is the problem of three-state and bidirectional pins.

The three-state case is the easier of the two to handle. For the case shown in Figure 4.2a, the "X" generator and data selector in Figure 4.2b may be used. The response from LASAR will be

B = A for enable ≥ 1,

B = "X" for enable = 0.

This character set is interpreted later (see section 4.2.3).

The bidirectional case requires the pin to be broken into an input and output signal. This is shown in Figure 4.3a and Figure 4.3b. The response from LASAR will be

Bout = A and C = A, for enable = 1,

Bout = "X" and C = Bin, for enable = 0.

(Note that Bin is supplied by ALICE, and Bout by LASAR.)

Depending on the function of the real device this truth table may have to be modified.

4.2.3 Using the Output of LASAR

Once the data generated by LASAR have been re-installed on Multics, the test vector set must be converted into a form suitable for use on ATE. If no three-state or bidirectional pins were present, this would be straightforward.

```
00300010101100110O
0000001010110011C0
0090001010110011O1
0000001010110011O0
0101011010110011O0
0101011010110O11O1
0101011010110011C0
0101011111011001O0
0101011111O11001O1
0101011111011001O0
0001001110110011O0
0001001110110011O1
0001001110110011O0
0111011010110011O0
0111011010110011C1
0111011010110011O0
0111011111011001O0
0111011111011001O1
0111011111011001C0
0011001110110011O0
0011001110110011O1
0011001110110011O0
1001011010110O11O0
1001011010110011O1
1001011010110011O0
1001011111011001O0
1001011111011001O1
1001011111011001C0
0010001110110011O0
0010001110110011C1
0010001110110011O0
1010011010110O11C0
1010011010110011C1
1010011010110011O0
1010011111011001O0
1010011111011001C1
1010011111011001C0
0110001110110011O0
0110001110110011C1
0110001110110O11O0
1011011010110O11O0
1011011010110011O1
1011011010110011C0
1011011111011001C0
1011011111011001C1
1011011111011001O0
```

```
0100001110110011C0
0100001110110011C1
0100001110110011O0
1101011010110011O0
1101011010110O11C1
1101011010110011O0
1101011111011001O0
1101011111011001O1
1101011111011001C0
1100001110110011O0
1100001110110011O1
1100001110110011O0
1110011010110011C0
1110011010110011O1
1110011010110011C0
1110011111011001O0
1110011111011001C1
1110011111011001O0
1000001110110011O0
1000001110110011O1
1000001110110011C0
1111011010110011C0
1111011010110011O1
1111011010110011C0
1111011111011001O0
1111011111011001O1
1111011111011001O0
```

Figure 4.1e. Output of ALICE program

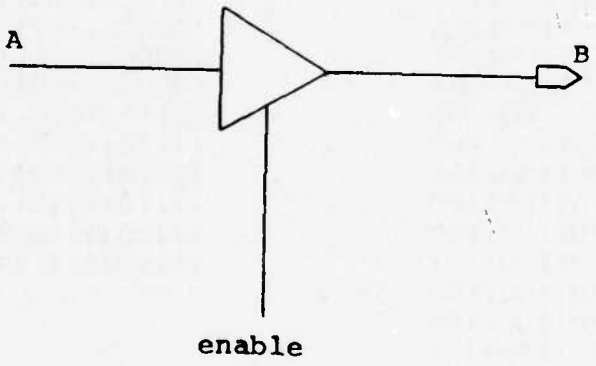


Figure 4.2a. Three-State Output

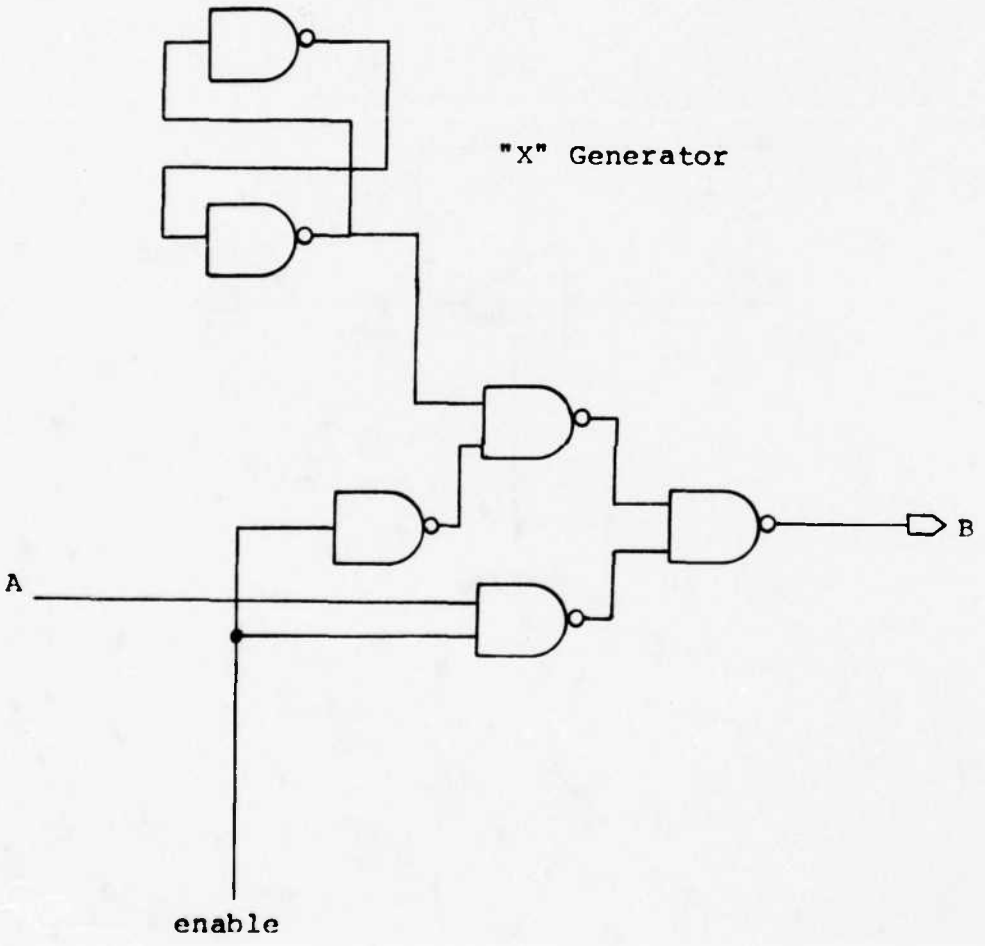


Figure 4.2b. LASAR workaround for three-state output

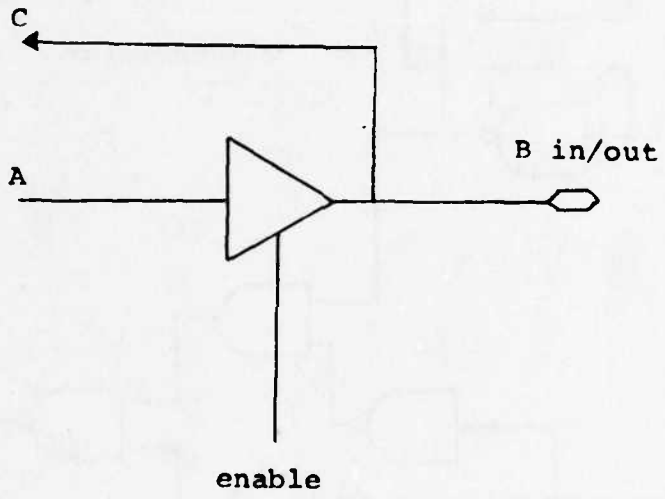


Figure 4.3a. Bidirectional Input/Output

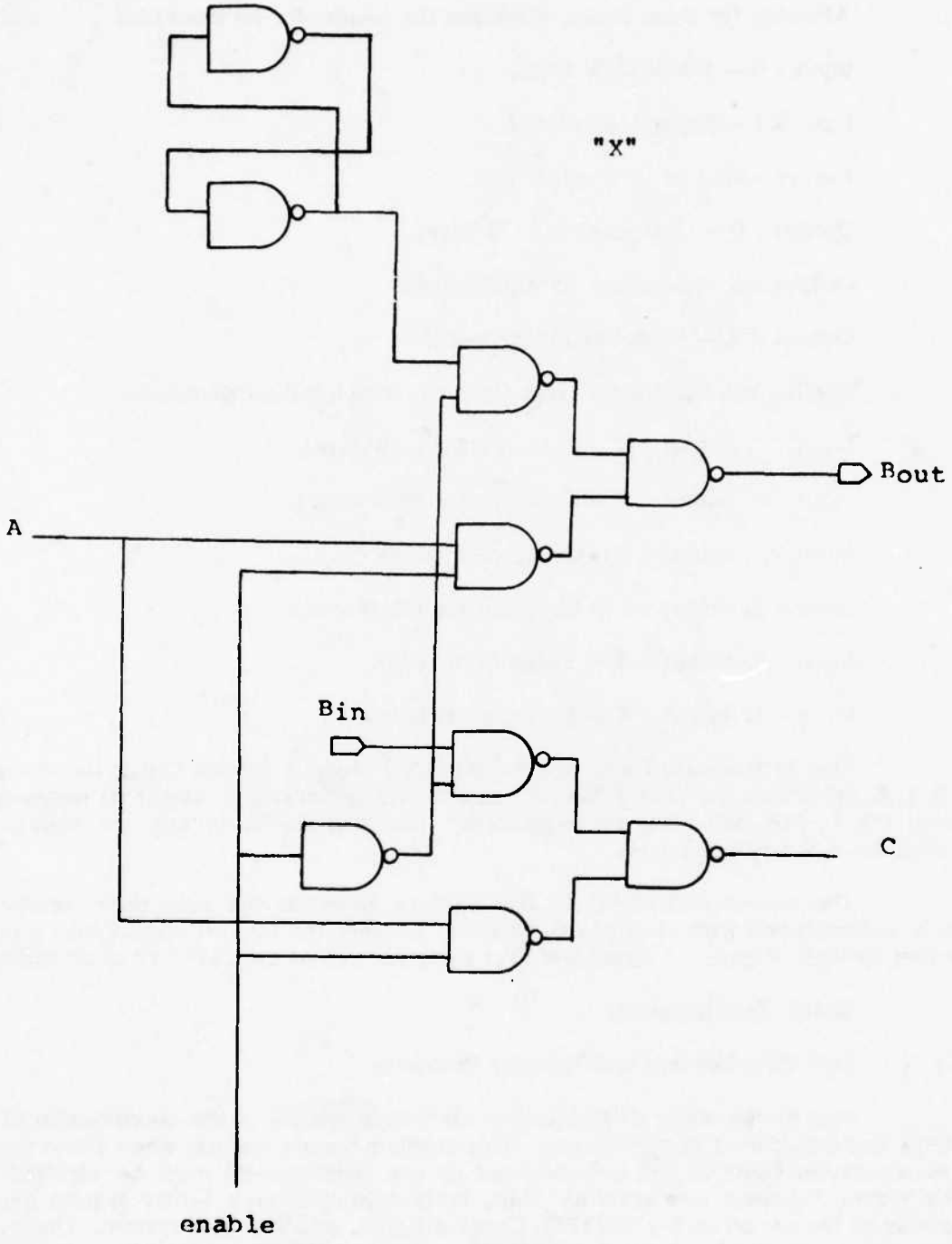


Figure 4.3b. LASAR workaround for bidirectional pin

Allowing for those cases, these are the results for an input pin:

Input = 0 -- Force LOW level,

Input = 1 -- Force HIGH level.

For an output or three-state pin:

Output = 0 -- Compare for LOW level,

Output = 1 -- Compare for HIGH level,

Output = X -- Mask (do not compare).

For the input and output signals comprising a bidirectional pin:

Input = 0, output = 0 -- Compare for LOW level,

Input = 0, output = 1 -- Compare for HIGH level,

Input = 1, output = 0 -- Compare for LOW level,

Input = 1, output = 1 -- Compare for HIGH level,

Input = 0, output = X -- Force LOW level,

Input = 1, output = X -- Force HIGH level.

This is translated into the S-3260/3270 Mode 3 format (using the character set 0, 1, L, H) or into the Sentry "SET F" statements (generating statements necessary to control the F, MA, MB, DA, DB registers). The utilities to do this are described in sections 2.1.8, 2.1.9, and 2.1.14.

The output of LASAR is also used to generate the slash sheet vector set. This is accomplished with Multics utilities that convert the LASAR output into a human oriented format. Figure 2.2 shows the first forty vectors of the 2914 LIT as an example.

4.3 SIMPL Test Language

4.3.1 Test Development and Transfer Problems

One of the major difficulties in electronic testing is the coordination of test systems and transfer of test programs. This problem becomes acute when a test from an IC manufacturer (with a test implemented on one test system) must be verified on a user's system (another test system). Thus, tests developed on a Sentry system must be redeveloped for use on an S-3260/3270, General Radio, or other test system. During this effort, differences between the S-3260/3270 systems at GEOS and RADC required major program changes as programs were transported from one facility to another.

Even between users of a particular test system there can be a program transfer problem of a different nature. The development of many similar test programs by different individuals often leads to a diversity of incompatible programming solutions

to particular test problems. This is undesirable for a number of reasons:

1) Many of these "solutions" do not actually solve the problem correctly,

2) The operation of a program is difficult to explain to someone who did not develop it and who uses a different approach, thus, the program cannot easily be understood and supported by others,

3) Because there is no "learning" (improvement of techniques) from one test programming project to another, the same mistakes may be repeated.

Proven, consistent, and standardized approaches to common test problems can result in higher quality and more efficient test designs.

4.3.2 Possible Solutions

Many major users of test equipment have been developing techniques that facilitate the test programming process. The approaches include:

4.3.2.1 Standardization of Test Systems.

This approach requires that all ATE users have access to one particular test system design with the same options and system software revision level. This approach often restricts evolutionary hardware improvements. The program transfer problem is minimized, but, even though a consistent language is used, similar programs can take on many divergent forms.

4.3.2.2 Conversion Programs

This approach uses a computer to convert a program in one format to another format and may require the user to develop unique parts of these conversion programs for every different configuration test system. In most cases it is impossible to directly convert a program (e.g., the conversion from a Sentry test to a S-3260/3270 test). Again, this technique addresses transfer between testers but not other test development problems.

4.3.2.3 Table Driven Test Generators

This is a generator of the DC tests, LIT, and some AC test statements (but not the test patterns). The generator interrogates the user for tables of device limits and outputs a matrix-driven, tester dependent program that will test devices to these limits. It provides a structured but complex and hard to follow program. The generation process is lengthy, but not difficult, and can be performed by a user with minimal knowledge of IC test philosophy, the particular tester language, or tester characteristics. A unique and comprehensive test generator must be developed for each type of ATE. Such a test generation program is difficult to develop, debug, and understand, and requires a large amount of computer memory. In addition, it is difficult to transport it to other ATE.

4.3.2.4 Text Processing of Tester Source Language

A text writing program can be written that displays to the test engineer a previously developed standard test program line-by-line. At certain preselected points in the text, the engineer is allowed to change the standard program to reflect the unique conditions of the device test specification. This process continues until all desired unique conditions are entered. The output of this program is a source program for the particular device to be tested on a particular tester. This technique provides an efficiently generated, well structured, and readable test program because the portion of a test that are the same from one device to another are automatically given the same code in exactly the same arrangement. The user must have a good understanding of the configuration and language of each target tester. This approach is only as good as the previously developed standard program.. It is difficult to provide limit checking and other techniques of guaranteeing the correctness on the generated program using this technique.

4.3.2.5 Test Program Written in a High Level Tester Independent Language

This approach would allow the user to develop test programs that are independent of any test system. A translator or interpreter program that accepts the tester independent program and converts or executes its code would have to be developed for each type of test system. However, any test programs developed in a tester independent language could be transported to other test systems, and function properly.

Control over the structure of tests developed depends on the level of the language. A tester independent language often must be written at a fairly high level in order to be "above" tester peculiarities. In a higher level language, structure can be forced on the programmer in such a way that more concise, correct and consistent test programs result. For example, a single high level "Measure VOL" statement can represent several possible sets of a dozen or so lower level tester statements. In actuality, the single high level statement is implemented in a predetermined and repeatable set of statements. Effective use of a high level tester independent language requires an experienced test engineer familar with IC testing, but not necessarily experienced in the operation and capabilities of any particular tester.

4.3.3 SIMPL Test Language

4.3.3.1 Philosophy of SIMPL

The SIMPL Test Language is a high level, almost tester independent language that simplifies the development of tests and their transportation between the GEOS S-3263/3270 and the RADC S-3260/3270 test systems. In the beginning of this effort, both GEOS and RADC had 7-phase S-3260 systems. However, when GEOS upgraded their system to a 14-phase system, major problems occurred in trying to develop, maintain and modify two different test programs for a particular device. Use of the SIMPL Test Language solved this problem and resulted in quickly generated, consistent, and easily debugged test programs.

It should be noted again that SIMPL as currently implemented and described below, is not completely tester independent. Certain test system modes and functional waveforms are unique to the S-3260/3270. However, most references to tester

peculiarities have been eliminated and it is probably feasible to eliminate those few that remain.

4.3.3.2 SIMPL Opcodes

The SIMPL opcodes developed to date allow the user to perform all DC parametrics and the LIT. Some AC parametrics (transitions and propagation time measurements) can also be performed. Tests that require special hardware (e.g., hardware pattern generators) have not been implemented. However, test programs for devices requiring these (RAMs, ROMs, and others) can be generated in SIMPL with only the special routines developed in TEKTEST by the user.

The following are the SIMPL Setup statements (reference Table 4.1):

INIT -- Initializes the test system, sets all inputs to 0.0V, sets all Power Supplies to 0.0V, etc.,

FINIS -- Resets the test system and prints the test results,

PWRSUP -- Sets the power supply voltage and checks the current supplied,

CONLD -- Connects load modules,

DRILEV -- Sets the Input Drive Levels,

CMPLEV -- Sets the Comparator Levels,

CYCLE -- Sets the system timing and mode,

PHASE -- Sets the specified phase to the appropriate delay,

PINFNT -- Sets each pin function (force, compare, inhibit, mask), mode (RTZ, RTO, etc.) and the data source ("1," "0," or pattern),

PATPAR -- Sets up the patterns for subsequent measurement tests,

EXCTST -- Executes the specified patterns for LIT or AC parametric verification, saving all errors for future failure analysis.

The following SIMPL opcodes make a specified measurement on the indicated pin and compare the results with the test limits:

MVCC -- Measure Power Supply Voltage,

MICC -- Measure Power Supply Current,

MVLDM -- Measure Load Module Voltage,

MILDM -- Measure Load Module Current,

MIISOL -- Measure Input Isolation Current,

CHART.EDT:SIM 31JUL80GEOS DISK NAME: E34 RADC F/I
DATE 27-AUG-80 TIME 14:10 PAGE 1 OF 5

*OPERATION	OPCODE	ARG1	ARG2	ARG3	ARG4	ARG5	NOTES
*INITIALIZE	INIT	-	-	-	-	-	
*POWER SUPPLY *SETUP	PWRSUP	NO.	0 = DISABLE, ELSE ENABLE	VOLTS	IMIN	IMAX	
*CONNECT LOADS	CONLO	I1	I2	LOAD MODULES NO, 1 OR 2	0 = DISCONNECT, ELSE CONNECT	-	1
*SET DRIVE *LEVELS	DRILEV	I1	I2	0 = DISCONNECT, ELSE CONNECT	VLOW	VHIGH	12
*SET COMPARE *LEVELS	CMPLEV	I1	I2	0 = DISCONNECT, ELSE CONNECT	VLOW	VHIGH	12
*SET CYCLE *TIME	CYCLE	MODE 1=F,I,C,M 2=F,C 3=FI,CM 4=FICM	CYCLE TIME				
*SET PHASES	PHASE	I1	I2	PHASE NUMBER	FROM	FOR	2
*PIN FUNCTION	PINFNT	I1	I2	1 = F 2 = I 3 = C 4 = M 12 = FI 34 = CM	DATA 0 = LOGIC ZERO 1 = LOGIC ONE 2 = PATTERN	TYPE 0 = NRZ 1 = RZ 2 = INVERT 3 = RZI 4 = RC 5 = RI	3 13 13
*PATTERN *PARAMETER	PATPAR	I1	I2	FIRST VECTOR	LAST VECTOR	OTHER VECTOR	4
*EXECUTE TESTS	EXCTST	I1	I2	FIRST VECTOR	LAST VECTOR	OTHER VECTOR	5
*MEASURE DEVICE *SUPPLY VOLTAGE	MVCC	VI1	POWER SUPPLY NO.	FORCED CURRENT	VMIN	VMAX	6
*MEASURE LOAD *MODULE VOLTS	MVLOW	I1	I2	FORCED CURRENT	VMIN	VMAX	

Table 4.1. SIMPL Opcodes

•OPERATION	OPCODE	ARG1	ARG2	ARG3	ARG4	ARG5	NOTES
•MEASURE LOAD •MODULE CURRENT	MILOM	I1	I2	FORCED VOLTAGE	IMIN	IMAX	
•MEASURE DEVICE •CURRENT	MICC	VI1	POWER SUPPLY NO.	FORCED VOLTAGE	IMIN	IMAX	
•MEASURE INPUT •ISOLATION •CURRENT	MIISOL	I1	I2	FORCED VOLTAGE	IMIN	IMAX	
•MEASURE CONTI- •NUITY ON INPUT	MVICON	I1	I2	FORCED CURRENT	VMIN	VMAX	
•MEASURE OUTPUT •VOLTAGE LOW •LOADED	MVOLL	I1	I2	FORCED CURRENT	VMIN	VMAX	II
•MEASURE INPUT •BREAKDOWN •CURRENT	MIIB	I1	I2	FORCED VOLTAGE	IMIN	IMAX	
•MEASURE OUTPUT •BREAKDOWN •CURRENT	MIOB	I1	I2	FORCED VOLTAGE	IMIN	IMAX	II
•MEASURE OUTPUT •HIGH IMPEDANCE •CURRENT HIGH	MIOZH	I1	I2	FORCED VOLTAGE	IMIN	IMAX	11
•MEASURE OUTPUT •HIGH IMPEDANCE •CURRENT LOW	MIOZL	I1	I2	FORCED VOLTAGE	IMIN	IMAX	11
•MEASURE INPUT •CLAMP DIODE	MVIC	I1	I2	FORCED CURRENT	VMIN	VMAX	
•MEASURE OUTPUT •CLAMP DIODE	MVOC	I1	I2	FORCED CURRENT	VMIN	VMAX	II
•MEASURE CONTI- •NUITY ON OUTPUT	MVOCON	I1	I2	FORCED CURRENT	VMIN	VMAX	

Table 4.1. SIMPL Opcodes, continued

*OPERATION	OPCODE	ARG1	ARG2	ARG3	ARG4	ARG5	NOTES
*MEASURE OUTPUT *SHORT CIRCUIT *CURRENT	MIOS	I1	I2	FORCED VOLTAGE	IMIN	IMAX	11
*MEASURE INPUT *CURRENT HIGH	MIIH	I1	I2	FORCED VOLTAGE	IMIN	IMAX	
*MEASURE INPUT *CURRENT LOW	MIIL	I1	I2	FORCED VOLTAGE	IMIN	IMAX	
*MEASURE OUTPUT *VOLTAGE LOW	MVOL	I1	I2	FORCED CURRENT	VMIN	VMAX	11
*MEASURE OUTPUT *VOLTAGE HIGH	MVOH	I1	I2	FORCED CURRENT	VMIN	VMAX	11
*MEASURE TIME	MTIME	I1	I2	VECTOR TRIGGER	TMIN	TMAX	11 14 15 16
*MEASURE DELTA *TIME BETWEEN *2 OUTPUT PINS	MDTIME	I1	I2	VECTOR TRIGGER	TMIN	TMAX	11 14 15
*UNIQUE TESTS	UNIQUE	-	-	-	-	-	7
*TEST COMPLETE	FINIS	-	-	-	-	-	8
*LOOP ON TEST	LOOP	-					17
*PAUSE ON TEST	PAUSE	-					17
*PRINT MESSAGE *TO OPERATOR	PRINT	TEXT TO BE PRINTED (110 CHARACTERS MAX.)					17

Table 4.1. SIMPL Opcodes, continued

```
CHART.EDT:SIM   31JUL80E03          DISK NAME:  E34 RADC F/I
DATE  27-AUG-80      TIME  14:10      PAGE   4 OF   5

*NOTES
*     1. I1 AND I2 ARE INDEX NUMBERS FOR THE PINLIST ALL.
*        FOR EXAMPLE: FORCE ALL(I1,I2) WITH... PINLIST ALL
*        CONTAINS THE INPUT PINS FOLLOWED BY THE OUTPUT PINS.
*        IT DOES NOT INCLUDE THE POWER AND GROUND PINS.
*        ABS VALUE OF I2 WILL ALWAYS BE GREATER THAN
*        THE ABS VALUE OF I1.
*
*     2. THE PHASE NUMBER IS BASED ON A SEVEN PHASE SYSTEM
*        AS FOLLOWS:
*           PHASE NO.     QUADANTS
*               1         FORCE 1 TO 16
*               2         FORCE 17 TO 32
*               3         FORCE 33 TO 48
*               4         FORCE 49 TO 64
*               5         LOCOMPARE 1 TO 64
*               6         HICOMPARE 1 TO 64
*               7         DATAPHASE 1 TO 64
*              56         BOTH HICOMPARE AND LOCOMPARE
*
*        THE "FROM","FOR" AND"CYCLE TIME" ARGUMENTS ARE IN
*        SECONDS. USE THE APPROPRIATE MULTIPLIERS, SUCH AS
*        N,U OR M FOR NANO,MICRO OR MILLI.
*
*        ONLY MODE 3 IS ALLOWED
*
*     3. FOR ARG3, 12 MEANS BOTH FORCE AND INHIBIT, AND 34
*        MEANS BOTH COMPARE AND MASK. ARG5 IS SIGNIFICANT ONLY
*        WHEN ARG3 IS 1 OR I2.
*
*     4. PATPAR SELECTS THE VECTORS FROM FILE *PATFIL.PAT*
*        FOR PARAMETRIC TESTING. NO ERRORS ARE SAVED. ARG5
*        (OTHER VECTOR) IS INTERRUPTED AS FOLLOWS:
*
*            0 DO NOT APPEND MASFIL.PAT AND DO NOT ALLOW MORE
*              LOGGING OF ERRORS.
*            1 APPEND MASFIL BUT DO NOT ALLOW LOGGING.
*            2 DO NOT APPEND BUT ALLOW MORE LOGGING
*            3 APPEND MASFIL AND ALLOW MORE LOGGING
*
*       NOTE:TO CLEAR THE PATTERN MOVE FLAG, ISSUE A " PATPAR 0 " STATEMENT
*
*     5. SPECIFIED BY ARG3 AND ARG4, THE ERRORS ARE SAVED.
*        IF ARG 1 IS NOT ZERO, ERRORS ARE RECORDED IN
*        ACCORDANCE WITH CONSOLE SWITCHES. ARG5 IS
```

Table 4.1. SIMPL Opcodes, continued

INTERRUPTED AS IN NOTE 4.

6. ARG1(VI1) IS THE INDEX FOR PINLIST MVCC.
   ARG3 IS ASSUMED TO BE ZERO, BUT MUST BE INCLUDED.

7. UNIQUE MERELY CALLS PART 150. THE USER MAY
   PROGRAM THIS PART AND ANY PART NUMBER GREATER
   THAN 200 TO PERFORM TESTS THAT CANNOT BE
   IMPLEMENTED BY THE OTHER OPCODES.

8. FINIS TERMINATES THE TEST AND DISPLAYS THE RESULTS.
   THIS MUST BE THE LAST LINE OF THE .ASC FILE.

9. LOOP CAUSES THE NEXT EXCTST TO BE EXECUTED
   REPEATEDLY UNTIL THE ADVANCE P/B IS DEPRESSED.

10. PAUSE CAUSES A HALT ON THE PIN OF THE FOLLOWING
    DC PARAMETRIC TEST. THIS ALLOWS MANUAL MEASUREMENT
    OF THE OUT PINS.

11. THIS ROUTINE REQUIRES PATPAR STATEMENT TO SET UP
    REQUIRED OUTPUTS.

12. THIS ROUTINE REQUIRES PINFNT STATEMENT TO SETUP
    DRIVE/COMPARE ELECTRONICS

13. TYPE 4 AND 5 AVAILABLE ON TEKTRONIX 3270 ONLY.
    TYPE 2 IS NOT AVAILABLE ON THE TEKTRONIX 3270

14. THIS TEST MUST BE PRECEDED BY 2 "PHASE" CODE THAT SETS
    THE COMPARE WINDOW WIDE ENOUGH TO DETECT THE OUTPUT
    TRANSITION.

15. THE INDEX PIN NUMBER WILL CONTAIN A SIGN INDICATING THE EXPECTED
    TRANSITION DIRECTION.

    + POSITIVE DIRECTION

    - NEGATIVE DIRECTION

16. THIS TEST WILL ONLY CONTAIN INDEX PINS WITH THE SAME
    EXPECTED TRANSITION DIRECTION.

17. THIS INSTRUCTION WILL HOLD THE TEST UNTIL THE OPERATOR
    PRESSES THE ADVANCE BUTTON

Table 4.1. SIMPL Opcodes, continued

MIIB -- Measure Input Breakdown Current,

MIOB -- Measure Output Breakdown Current,

MVICON -- Measure Input Continuity,

MVOCON -- Measure Output Continuity,

MVIC -- Measure Input Clamp Diode,

MVOC -- Measure Output Clamp Diode,

MIIL -- Measure Input Current Low,

MIIH -- Measure Input Current High,

MIOZH -- Measure Output High-Impedance Current High,

MIOZL -- Measure Output High-Impedance Current Low,

MVOL -- Measure Output Voltage Low,

MVOLL -- Measure Output Voltage Low Loaded,

MVOH -- Measure Output Voltage High,

MIOS -- Measure Output Short Circuit Current,

MTIME -- Measure Time between a system reference time and the specified pin transition,

MDTIME -- Measure Time between two specified pins' transitions.

The following SIMPL codes implement utility functions of the language.

UNIQUE -- Initiates a user defined test routine

PAUSE -- Pauses on each pin of a DC measurement test,

LOOP -- Causes the pattern applied by the next EXCTST statement to be repeated continuously,

PRINT -- Causes a comment to be printed.

4.3.3.3 Program Restrictions

As previously stated, the SIMPL language was developed to minimize tester interface and some test development problems. However, when developing a device test, normal bench test practices and procedures should be followed. Thus the user selects the appropriate SIMPL opcodes in the same sequence required for a bench test. For example:

-- Setting VCC before setting drive and compare levels,

-- Setting cycle and phase timing before setting the pin function.

4.3.4 SIMPL Interpreter Programs

4.3.4.1 Philosophy

The SIMPL test language opcodes are system independent which allows a device or module test program to be transported from one test system to another. An interpreter will be required for each test system to decode the SIMPL statements and then implement the required functions.

4.3.4.2 SIMFAIR Interpreter (NOT YET IMPLEMENTED)

This interpreter would be written in Sentry Test Language (FACTOR) to implement SIMPL statements on the Sentry. Although the feasibility of such a program was verified, the actual implementation was not accomplished because a Sentry was not available for this effort.

4.3.4.3 SIMTEK Interpreter

This program was written in the TEKTEST Language for use on S-3260/3270 test systems. At present, SIMTEK interpreters have been written for four systems:

GEOS -- S-3263

GEOS -- S-3270

RADC -- S-3260

RADC -- S-3270

Each of these systems had slightly different hardware configurations which required minor software variations to execute the SIMPL opcodes. Most of the devices in this report were tested using SIMPL programs on several of the systems.

4.3.5 Developing a SIMTEK Test

In order to show the uses of SIMPL and its associated programs, a test will be developed using the 2914 device slash sheet as an example. The steps shown do not necessarily have to be performed in the order shown. However, they show the procedure used to implement the LIT of the 2914.

4.3.5.1 Philosophy

The original idea of the SIMPL language was to have a single generic program capable of testing many devices. However, due to the constraints of the TEKTEST Language, a different complete ".TST" program is required for each device.

In order to develop a device test file (".TST") the following are required:

4.3.5.1.1 The ".PIN" file

This file relates a particular pin name to its associated sector card. It also indicates whether the indicated pin name is to be an input or output pin.

4.3.5.1.2 The ".PAT" file

The pattern file (".PAT") contains test vectors to be applied and compared to the DUT.

4.3.5.1.3 The PINLIST

The PINLIST is a list of pin names that are associated with the test program. This section is required in the main program.

4.3.5.1.4 The Main Program

The main program contains the PINLIST as well as the pattern file references. Included are all the TEKTEST instructions required to implement the device test.

Each of these sections is required by the system translator which converts them into a test file (".TST").

A SIMPL program contains opcode statements to be executed. The TEKTEST program reads an opcode from a SIMPL program (stored in an ".ASC" file), and proceeds to the section of the main program that will implement that function. Thus, a module of code was developed to implement each SIMPL opcode.

Certain repeated functions (printing results, testing pin functions, etc) were developed into subroutines that could be called by other routines. This meant that only one section of a device program had to be developed per function.

Figure 4.4 shows the layout of the SIMTEK program.

4.3.5.2 Development of Load Modules and Socket Card Assemblies

The socket card assembly is used to interface the DUT with the test systems sector cards. The user determines the phases required for each device pin such that the proper phase times may be applied during each test. Also any output load modules should be developed at this time. Figures 4.5 and 4.6 are examples of the load module and sector card assemblies.

4.3.5.3 Development of Pin Assignment File

The pin assignment (".PIN") file indicates which sector is connected to a particular device pin name. The device pin name will be used in the PINLIST file. Figure 4.7 is an example of the pin assignment file.

4.3.5.4 Development of PINLIST (SC005.EDT) File

The pinlist file is required by the main (".EDT") program. It is one of the two sections (section 5) to be developed for a SIMTEK program by the user. The other

Section			
1	SC001	Header & Test Setup Info.	User Supplied *
2	SC002	Table Of Contents	Standard Prog.
3	SC003	Subroutine/Functions	
4	SC004	Array Declarations	
5	SC005	Pinlist/Pin Array Info.	User Supplied *
7	SC007	Console Switch	Standard Prog.
9	SC009	Tester Definitions Simtek Required Sections	Tester Dependent Prog.
10	SC010	Simtek Opcode Interpreter	Standard Prog.
15	SC015	Tester Initialization	Tester Dependent Prog.
20	SC020	Power Supply Setup	Standard Prog.
25	SC025	Pwr. Sup. Setup & Test. Subroutine	Tester Dependent Prog.
30	SC030	Set Drive Levels	Standard Prog.
35	SC035	Set Compair Levels	
40	SC040	Connect Loads	
45	SC045	Set Phases	Tester Dependent Prog.
50	SC050	Declare Pin Function Definition	
55	SC055	Set Mode & Cycle Time	Standard Prog.
60	SC060	Load Measurement Pattern Vectors	
65	SC065	Pattern Move Subroutine	
70	SC070	Logic Integrity Test (LIT)	
75	SC075	LIT Log Routine	
80	SC080	DUT Supply Current Test	
85	SC085	DUT Supply Voltage Test	
90	SC090	Input Voltage Test	
95	SC095	Input Current Test	
100	SC100	Output Voltage Test	
105	SC105	Output Current Test	
110	SC110	Load Module Voltage Test	
115	SC115	Load Module Current Test	
120	SC120	Measure Time Test	
125	SC125	Measure Delta Time Test	
135	SC135	Check For Bidirectional Output Pin	
136	SC136	Check Test Limits	
137	SC137	Check For Bidirectional Input Pin	
140	SC140	Print Test Results	
145	SC145	End Of Test Routine	
150	SC150	Unique Test	Standard Prog. or User Supplied *

SIMTEK Program Layout

DOC
13/Aug./80

Figure 4.4. SIMTEK Program Layout

Load Module

Load - 2914 -1

Pin	Name
1	L1
2	XGB
3	+15V (Bus 1)
4	+5V (Bus 2)
5	Gnd.
6	VS6 (Bus 3)
7	VS7 (Bus 4)
8	N+1
9	N-1
10	B

Circuit

Load - 2914 -2

Pin	Name
1	L1
2	XGB
3	+15V (Bus 1)
4	+5V (Bus 2)
5	Gnd.
6	VS6 (Bus 3)
7	VS7 (Bus 4)
8	N+1
9	N-1
10	B

Circuit

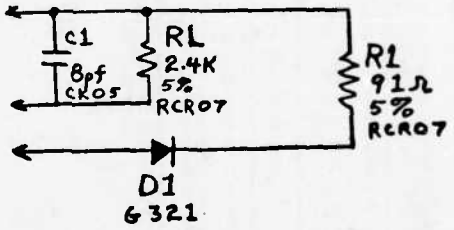


Figure 4.5. 2914 Load Modules

2914
Socket Card

← Phase 11 (Hi Comp)
← Phase 9 (Lo Comp) →
Phase 13

Sector	Phase	2942	Load #	I/O	DUT Pin No.	Sig. Nam.
W1	1	Y0				
X2	1	Y1				
Y3	1	Y2				
Z4	1	Y3				
W5	1	Y4	-2	0	3	GPSIG
X6	1	Y5				
Y7	1	Y6				
Z8	1	Y7		I/O	6	INT/INTT
W9	2	Y8	-2	0	7	RIPDIS
X10	2	Y9	-2	0	8	PARDIS
Y11	2	Y10	-1	0	9	IRQ
Z12	2	Y11		I/O	10	VCCM/VCCF
W13	2	INH		I	13	SIØ
X14	2	D0		I	12	SI1
Y15	2	D1		I	11	SI2
Z16	2	D2				
W17	3	D3	-2	0	17	V1
X18	3	D4	-2	0	11	SØ2
Y19	3	D5	-2	0	12	SØ1
Z20	3	D6	-2	0	13	SOØ
W21	3	D7	-2	0	14	STOVFL
X22	3	D28	-2	0	15	GAS
Y23	3	D8	-2	0	18	VØ
Z24	3	D9	-2	0	16	V2
W25	4	D10		I	2	MI3
X26	4	D11		I	40	MI2
Y27	4	D12		I	38	MI1
Z28	4	D13		I	36	MIØ
W29	4	D14		I	25	MI4
X30	4	D15		I	23	MI5
Y31	4	D29		I	21	MI6
Z32	4	D16		I	19	MI7

← Phase 10 (Lo Comp)
← Phase 12 (Hi Comp) →
Phase 14

Sector	Phase	2942	Load #	I/O	DUT Pin No.	Sig. Nam.
W33	5	D12				
X34	5	-				
Y35	5	D18				
Z36	5	-				
W37	5	D19				
X38	5	D20				
Y39	5	D21	-2	0	19	MO7
Z40	5	D30	-2	0	21	MO6
W41	6	W	-2	0	23	MO5
X42	6	-	-2	0	25	MO4
Y43	6	-	-2	0	36	MOØ
Z44	6	-	-2	0	38	MO1
W45	6	-	-2	0	40	MO2
X46	6	D22	-2	0	2	MO3
Y47	6	D23		I/O	27	LTB/LTBT
Z48	6	D24		I/O	29	CLK/CLKT
W49	7	D31		I/O	28	IØ/IØT
X50	7	D25		I/O	31	IØ/IØT
Y51	7	D26		I/O	32	I2/I2T
Z52	7	D27		I/O	33	I3/I3T
W53	7	X0		I/O	34	ISEN/ISENT
X54	7	X1				
Y55	7	X2		I/O	5	GEN1/T
Z56	7	X3		I/O	4	GAR/GART
W57	8	X4		I/O	22	P6/P6T
X58	8	X5		I/O	24	P5/P5T
Y59	8	X6		I/O	26	P4/P4T
Z60	8	X7		I/O	35	PØ/PØT
W61	8	X8		I/O	37	P1/P1T
X62	8	X9		I/O	39	P2/P2T
Y63	8	X10		I/O	20	P7/P7T
Z64	8	X11		I/O	1	P3/P3T

DOC
8/27/80

Figure 4.6. 2914 Socket Card Layout

```
OK1:2914.PIN;603    DATE: 27-AUG-80    TIME: 13:42:09

  LINE      SECTOR    PIN       OUT PIN
 NUMBER     NUMBER    NAME      OR COMMENT

   1.0000     5WA0    GPSIG     PIN 3
   2.0000    56ZAI    GAR       PIN 4
   3.0000    56ZA0    GAR7      PIN 4
   4.0000    55YAI    GEN1      PIN 5
   5.0000    55YA0    GEN1T     PIN 5
   6.0000     8ZAI    INT       PIN 6
   7.0000     8ZA0    INTT      PIN 6
   8.0000     9WA0    RIPOIS    PIN 7
   9.0000    10XA0    PAROIS    PIN 8
  10.0000    11YA0    IRQ       PIN 9
  11.0000    12ZAI    VCCM      PIN 10
  12.0000    12ZA0    VCCF      PIN 10
  13.0000    13WAI    SI0       PIN 13
  14.0000    14XA1    SI1       PIN 12
  15.0000    15YAI    SI2       PIN 11
  16.0000    18XA0    SO2       PIN 11
  17.0000    19YA0    SO1       PIN 12
  18.0000    20ZA0    SO0       PIN 13
  19.0000    21WA0    S70VFL    PIN 14
  20.0000    22XA0    GAS       PIN 15
  21.0000    24ZA0    V2        PIN 16
  22.0000    17WA0    V1        PIN 17
  23.0000    23YA0    V0        PIN 18
  24.0000    25WAI    MI3       PIN 2
  25.0000    26XAI    MI2       PIN 40
  26.0000    27YAI    MI1       PIN 38
  27.0000    28ZAI    MI0       PIN 36
  28.0000    29WAI    MI4       PIN 25
  29.0000    30XAI    MI5       PIN 23
  30.0000    31YAI    MI6       PIN 21
  31.0000    32ZAI    MI7       PIN 19
  32.0000    39YA0    MO7       PIN 19
  33.0000    40ZA0    MO6       PIN 21
  34.0000    41WA0    MO5       PIN 23
  35.0000    42XA0    MO4       PIN 25
  36.0000    43YA0    MO0       PIN 36    56.0000    58XAI    P5      PIN 24
  37.0000    44ZA0    MO1       PIN 38    57.0000    58XA0    P5T     PIN 24
  38.0000    45WA0    MO2       PIN 40    58.0000    59YAI    P4      PIN 26
  39.0000    46XA0    MO3       PIN 2     59.0000    59YA0    P4T     PIN 26
  40.0000    47YAI    LTB       PIN 27    60.0000    60ZAI    P0      PIN 35
  41.0000    47YA0    LTBT      PIN 27    61.0000    60ZA0    P0T     PIN 35
  42.0000    48ZAI    CLK       PIN 29    62.0000    61WAI    P1      PIN 37
  43.0000    48ZA0    CLK7      PIN 29    63.0000    61WA0    P1T     PIN 37
  44.0000    49WAI    I0        PIN 28    64.0000    62XAI    P2      PIN 39
  45.0000    49WA0    I0T       PIN 28    65.0000    62XA0    P2T     PIN 39
  46.0000    50XAI    I1        PIN 31    66.0000    63YAI    P7      PIN 20
  47.0000    50XA0    I1T       PIN 31    67.0000    63YA0    P7T     PIN 20
  48.0000    51YAI    I2        PIN 32    68.0000    64ZAI    P3      PIN 1
  49.0000    51YA0    I2T       PIN 32    69.0000    64ZA0    P3T     PIN 1
  50.0000    52ZAI    I3        PIN 33
  51.0000    52ZA0    I3T       PIN 33
  52.0000    53WAI    INSEN     PIN 34
  53.0000    53WA0    INSENT    PIN 34
  54.0000    57WAI    P6        PIN 22
  55.0000    57WA0    P6T       PIN 22
```

Figure 4.7. 2914 ".PIN" File

section is described in section 4.3.5.7. Reference Figure 4.8 for an example of section 5. This pinlist file consists of two parts described below.

4.3.5.4.1 PIN Array

The input and output pin numbers are listed in an integer array called PIN. These pins are in the same order as in the pattern file and the pinlist. A second integer array (PVCC) lists the power supply pin numbers.

4.3.5.4.2 Pinlists

The pin names in the pin assignment file are listed in the format required for TEKTEST pinlists. Four pinlists are mandatory for the SIMTEK Program:

-- Pinlist ALL for all inputs and all outputs (required for DC and LIT tests),

-- Pinlist ALLTIM for all inputs (connected to comparators) and alls (required for time measurements),

-- Pinlist MVCC for all device power pins (required for power supply voltage measurements). These pin names are isolated from the DUT VCC pin(s) by 1K resistor(s),

-- Pinlist FVCC for all device power pins (required for power supply current measurements). These pin names are connected directly to the DUT VCC pins.

4.3.5.5 Development of a Pattern (".PAT") File

The pattern file that is to be used to test the DUT must be stored in a file called PATFIL.PAT. The order of the columns must be consistent with pinlist ALL and the PIN array. The first and last vector number of each block of test should be saved for future use in developing SIMPL codes. Figure 4.9 is an example of a PATFIL.PAT file. This file was generated from the slash sheet vector set shown in Figure 2.2.

4.3.5.6 Development of a Dummy Pattern (".PAT") File

In the S-3260/3270 test system, the last four error bits of information remain in the test tables shift register (see section 2.4.4.3). The addition of four dummy vectors (stored in MASFIL.PAT) at the end of the test pattern will result in the last four errors being shifted out. The vectors developed by the user should not cause any internal state changes in the DUT. Figure 4.10 shows an example of a MASFIL.PAT file.

4.3.5.7 Development of the Header (SC001.EDT) File

The header file documents the device test, listing pertinent information such as:

-- Test Specification,

-- Device Name,

```
SC005.EDT;G03   02JUL80 GEOS          DISK NAME:  E21 RKT RADC
DATE  27-AUG-80      TIME  13:39      PAGE   1 OF   2

   5.0100 * PIN ARRAY PINLIST FOR THE 2914 DEVICE
   5.0200 *
   5.0300 * PIN VECTOR LIST
   5.0400 *
   5.0500 * INPUTS
   5.0600 *
   5.0700   PIN(1) = 19
   5.0800   PIN(2) = 21
   5.0900   PIN(3) = 23
   5.1000   PIN(4) = 25
   5.1100   PIN(5) = 2
   5.1200   PIN(6) = 40
   5.1300   PIN(7) = 38
   5.1400   PIN(8) = 36
   5.1500   PIN(9) = 11
   5.1600   PIN(10) = 12
   5.2200   PIN(11) = 13
   5.2300   PIN(12) = 20
   5.2400   PIN(13) = 22
   5.2500   PIN(14) = 24
   5.2600   PIN(15) = 26
   5.2700   PIN(16) = 1
   5.2800   PIN(17) = 39
   5.2900   PIN(18) = 37
   5.3000   PIN(19) = 35
   5.3100   PIN(20) = 34
   5.3200   PIN(21) = 33
   5.3300   PIN(22) = 32
   5.3400   PIN(23) = 31
   5.3500   PIN(24) = 28
   5.3600   PIN(25) = 5
   5.3700   PIN(26) = 4
   5.3800   PIN(27) = 6
   5.3900   PIN(28) = 27
   5.4000   PIN(29) = 29
   5.4100   LASTINPUT = 29
   5.4200 *
   5.4300 * OUTPUTS
   5.4400 *
   5.4500   PIN(30) = 19
   5.4600   PIN(31) = 21
   5.4700   PIN(32) = 23
   5.4800   PIN(33) = 25
   5.4900   PIN(34) = 2
   5.5000   PIN(35) = 40
   5.5100   PIN(36) = 38
   5.5200   PIN(37) = 36
   5.5300   PIN(38) = 11
   5.5400   PIN(39) = 12
   5.5500   PIN(40) = 13
   5.5600   PIN(41) = 16
   5.5700   PIN(42) = 17
   5.5800   PIN(43) = 18
   5.5900   PIN(44) = 3
   5.6000   PIN(45) = 15
   5.6100   PIN(46) = 14
   5.6200   PIN(47) = 9
```

Figure 4.8. 2914 PINARRAY

```
SC005.EDT;G03   02JUL80 GEOS          DISK NAME:  E21 BKT RADC
DATE  27-AUG-80      TIME  13:39      PAGE   2 OF   2

    5.6300    PIN(48) = 8
    5.6400    PIN(49) = 7
    5.6500    LASTOUTPUT = 49
    5.6600 *
    5.6700 * PINLIST
    5.6800 *
    5.6900    PINLIST M0=M07,M06,M05,M04,M03,M02,M01,M00
    5.7000    PINLIST M1=M17,M16,M15,M14,M13,M12,M11,M10
    5.7100    PINLIST P=P7,P6,P5,P4,P3,P2,P1,P0
    5.7200    PINLIST I=INSEN,I3,I2,I1,I0
    5.7300    PINLIST V=V2,V1,V0
    5.7400    PINLIST GEN=GEN1
    5.7500    PINLIST S0=S02,S01,S00
    5.7600    PINLIST S1=S12,S11,S10
    5.7700
    5.7800    PINLIST INL=M1,S1,P,I,GEN,GAR,INT,LTB,CLK
    5.7900    PINLIST OUTL=M0,S0,V,GPSIG,GAS,STOVFL,IRQ,PARDIS,RIPDIS
    5.8000
    5.8100    PINLIST INTIME = M0,S0/
    5.8200    P7T,P6T,P5T,P4T,P3T,P2T,P1T,P0T/
    5.8300    INSENT,I3T,I2T,I1T,I0T/
    5.8400    GEN1T,GART,INTT,LTBT/
    5.8500    CLKT
    5.8600
    5.8700    PINLIST ALLTIM = INTIME,OUTL
    5.8800
    5.8900    PINLIST ALL = INL,OUTL
    5.9000
    5.9100 * MVCC PINLIST
    5.9200    PINLIST MVCC = VCCM
    5.9300    PVCC(1) = 10
    5.9400    PINLIST PVCC = VCCP
```

Figure 4.8. 2914 PINARRAY, continued

```
DATE  28-AUG-80      TIME  10:07      PATTERN FILE  PATFIL.PAT;G03

  1                    0            0             0           0
    TO                 1            2             3           4
       49  12345678 901 23456789 01234 5678 9 01234567 890 123 456789

  0.0100  *>>>>>>>>>>> PATTERN FILE FOR THE 2914 DEVICE <<<<<<<<<<<
  0.0200  *
  0.0300  *SPACE 8-3-8-5-4-1-8-3-3-6,10,43,54
  0.0400  *
  0.0500  *
  0.0600  *
  0.0700  *
  0.0800  *MMMMMMMM SSS PPPPPPPP IIIII GGIL C MMMMMMMM SSS VVV GGSIPR
  0.0900  *76543210 210 76543210 E3210 EADR L 76543210 210 210 SADRDD
  0.0950  *                              R   K

  1.0000    00000000 000 11111111 00000 0010 1 00000000 000 000 0H0000
  2.0000    00000000 000 11111111 00000 0010 0 00000000 000 000 0H0000
  3.0000    00000000 000 11111111 00000 0010 1 00000000 000 000 LHHHHL
  4.0000    00000000 000 11111111 00101 0010 1 00000000 000 000 LHHHHL
  5.0000    00000000 000 11111111 00101 0010 0 00000000 000 000 LHHHHL
  6.0000    00000000 000 11111111 00101 0010 1 00000000 000 000 LHHHHL
  7.0000    LLLLLLLL 000 11111111 00111 0010 1 LLLLLLLL 000 000 LHHHHL
  8.0000    LLLLLLLL 000 11111111 00111 0010 0 LLLLLLLL 000 000 LHHHHL
  9.0000    LLLLLLLL 000 11111111 00111 0010 1 LLLLLLLL 000 000 LHHHHL
 10.0000    00000000 000 11111111 01000 0010 1 00000000 000 000 LHHHHL

 11.0000    00000000 000 11111111 01000 0010 0 00000000 000 000 LHHHHL
 12.0000    00000000 000 11111111 01000 0010 1 00000000 000 000 LHHHHL
 13.0000    LLLLLLLL 000 11111111 00111 0010 1 HHHHHHHH 000 000 LHHHHL
 14.0000    LLLLLLLL 000 11111111 00111 0010 0 HHHHHHHH 000 000 LHHHHL
 15.0000    LLLLLLLL 000 11111111 00111 0010 1 HHHHHHHH 000 000 LHHHHL
 16.0000    00000000 000 11111111 01100 0010 1 00000000 000 000 LHHHHL
 17.0000    00000000 000 11111111 01100 0010 0 00000000 000 000 LHHHHL
 18.0000    00000000 000 11111111 01100 0010 1 00000000 000 000 LHHHHL
 19.0000    LLLLLLLL 000 11111111 00111 0010 1 LLLLLLLL 000 000 LHHHHL
 20.0000    LLLLLLLL 000 11111111 00111 0010 0 LLLLLLLL 000 000 LHHHHL

 21.0000    LLLLLLLL 000 11111111 00111 0010 1 LLLLLLLL 000 000 LHHHHL
 22.0000    01010101 000 11111111 01011 0010 1 00000000 000 000 LHHHHL
 23.0000    01010101 000 11111111 01011 0010 0 00000000 000 000 LHHHHL
 24.0000    01010101 000 11111111 01011 0010 1 00000000 000 000 LHHHHL
 25.0000    LLLLLLLL 000 11111111 00011 0010 1 LHLHLHLH 000 000 LHHHHL
 26.0000    LLLLLLLL 000 11111111 00011 0010 0 LHLHLHLH 000 000 LHHHHL
 27.0000    LLLLLLLL 000 11111111 00011 0010 1 LHLHLHLH 000 000 LHHHHL
 28.0000    10101010 000 11111111 01101 0010 1 00000000 000 000 LHHHHL
 29.0000    10101010 000 11111111 01101 0010 0 00000000 000 000 LHHHHL
 30.0000    10101010 000 11111111 01101 0010 1 00000000 000 000 LHHHHL

 31.0000    LLLLLLLL 000 11111111 00111 0010 1 LHLHLHLH 000 000 LHHHHL
 32.0000    LLLLLLLL 000 11111111 00111 0010 0 LHLHLHLH 000 000 LHHHHL
 33.0000    LLLLLLLL 000 11111111 00111 0010 1 LHLHLHLH 000 000 LHHHHL
 34.0000    10101010 000 11111111 01011 0010 1 00000000 000 000 LHHHHL
 35.0000    10101010 000 11111111 01011 0010 0 00000000 000 000 LHHHHL
 36.0000    10101010 000 11111111 01011 0010 1 00000000 000 000 LHHHHL
 37.0000    01010101 000 11111111 01010 0010 1 00000000 000 000 LHHHHL
 38.0000    01010101 000 11111111 01010 0010 0 00000000 000 000 LHHHHL
 39.0000    01010101 000 11111111 01010 0010 1 00000000 000 000 LHHHHL
 40.0000    01010101 LLL 11111111 00110 0010 1 00000000 LLL 000 LHHHHL
```

Figure 4.9. 2914 PATFIL

```
DATE  27-AUG-80     TIME  15:37     PATTERN FILE  MASFIL.PAT;G03

1                    0          0          0          0
  TO                 1          2          3          4
      49  12345678 901 23456789 01234 5678 9 01234567 890 123 456789

  0.0100  *>>>>>>>> DUMMY PATTERN FILE FOR THE 2914 DEVICE <<<<<<<<
  0.0200  *
  0.0300  *SPACE 8-3-8-5-4-1-8-3-3-6,10,43,54
  0.0400  *
  0.0500  *
  0.0600  *
  0.0700  *
  0.0800  *MMMMMMMMM SSS PPPPPPPP IIIII GGIL C MMMMMMMM SSS VVV GGSIPR
  0.0900  *76543210 210 76543210 E3210 EADB L 76543210 210 210 SAORDD
  0.0950  *                                R   K

  1.0000   LLLLLLLL LLL LLLLLLLL 11111 LLLL 1 00000000 000 000 000000
  2.0000   LLLLLLLL LLL LLLLLLLL 11111 LLLL 1 00000000 000 000 000000
  3.0000   LLLLLLLL LLL LLLLLLLL 11111 LLLL 1 00000000 000 000 000000
  4.0000   LLLLLLLL LLL LLLLLLLL 11111 LLLL 1 00000000 000 000 000000
```

Figure 4.10. 2914 MASFIL

-- Required Hardware/Files,

-- Output Loads.

Figure 4.11 shows an example of a header file.

4.3.5.8 Development of a Device SIMTEK Program

As previously stated, a S-3260/3270 test file (".TST") is required for each device to be tested. Another requirement is for any test pattern files to be stored under the same User Identification (UID). Thus the following files, required for the device test program, should be developed and stored under the same UID.

-- Pin file - (name).PIN - see section 4.3.5.3,

-- Pinlist file - SC005.EDT - see section 4.3.5.4,

-- Pattern file - PATFIL.PAT - see section 4.3.5.5,

-- Pattern file - MASFIL.PAT - see section 4.3.5.6,

-- Header file - SC001.EDT - see section 4.3.5.7,

To simplify the development of a device test program, the SIMTEK files used by all test programs have been stored under the UID called ":SIM." Also included in this UID are the tester files unique to particular test systems:

GEOS S-3263 -- SC009A.EDT,

RADC S-3260 -- SC009B.EDT,

GEOS S-3270 -- SC009C.EDT,

RADC S-3270 -- SC009D.EDT.

Thus, to assemble the complete device test program:

1) Enter the EDIT Program,

2) Input the header file (SC001.EDT),

3) Save under XXXXXX.EDT (where XXXXXX is the device part number),

4) Merge the Pinlist file (SC005.EDT),

5) Merge Tester File SC009Y.EDT (where Y is the alpha character assigned to a particular test system).

At this point, typing PERFORM will allow the editor to merge all the other required files and save them in XXXXXX.EDT. Translation of this file with the ".PIN" file will result in a test file (".TST") capable of executing "SIMPL" language statements.

```
   1.EDT;G03   31JUL80GEO3          DISK NAME:  E21 8KT RADC
DATE  28-AUG-80     TIME  10:05     PAGE   1 OF   2

1.0100 * ************************************************************
1.0120 *                                                             *
1.0140 *                  TEKTRONIX TEST PROGRAM                     *
1.0160 *                                                             *
1.0180 *                CIRCUIT TEST ENGINEERING                     *
1.0200 *            GENERAL ELECTRIC ORDNANCE SYSTEMS                *
1.0220 *                PITTSFIELD, MASSACHUSETTS                    *
1.0240 *                                                             *
1.0260 *                                                             *
1.0280 *  . DUT DESCRIPTION          :VECTOR PRIORITY                 *
1.0300 *                              INTERRUPT CONTROLLER           *
1.0320 *  . PROGRAMER                :B.K.TEAGUE/SLASAR               *
1.0340 *                                 D. O'CONNOR                 *
1.0360 *                                                             *
1.0380 *  FOR SUPPORTING DOCUMENTS, CONSULT ETEC AND OTHER           *
1.0400 *  CABINET FILES UNDER THE FOLLOWING IDENTIFIED TEST          *
1.0420 *  SPECIFICATION NUMBER AND ADAPTER NUMBERS, AS WELL AS       *
1.0440 *  LISTINGS OF THE FOLLOWING IDENTIFIED DISK OR MAGNETIC      *
1.0460 *  TAPE FILES.                                                *
1.0480 *                                                             *
1.0500 *  . TEST SPECIFICATION       :MIL-M-38510/440                 *
1.0520 *                                                             *
1.0540 *  . TEST TYPE/CONDITIONS     :TIMING AND FUNCTIONAL           *
1.0560 *                                                             *
1.0580 *  . EDIT FILE                :2914                            *
1.0600 *                                                             *
1.0620 *  . PIN ASSIGNMENT FILE      :2914                            *
1.0640 *                                                             *
1.0660 *  . PATTERN FILE             :PATFIL                          *
1.0680 *                                                             *
1.0700 *  . TEST FILE                :2914                            *
1.0720 *                                                             *
1.0740 *  . THIS LISTING IS          :TEKTEST EDIT                    *
1.0760 *                                                             *
1.0780 *  . SOCKET CARD ASSEMBLY #   :2036                            *
1.0800 *                                                             *
1.0820 * ************************************************************
1.0840 *
1.0860 *   SWITCH POSITION        FUNCTION
1.0880 *          0               DATA LOGGING
1.0900 *          1               LOGIC INTEGRITY TESTING
1.0920 *          2               DC MEASUREMENT TESTS
1.0940 *          3               TIME MEASUREMENT TESTS
1.0960 *          4               UNIQUE TESTS
1.0980 *          5
1.1000 *          6
1.1020 *          7               SUPRESS ALL PRINTING
1.1040 *          8               PRINT TO .ASC FILE
1.1060 *          9               PRINT TO LP
1.1080 *         10               MATRIX PRINTED
1.1100 *         11               PRINT TEST RESULTS
1.1120 *         12               MATRIX MOD. = SINGLE STEP
1.1140 *         13               PAUSE ON TEST
1.1160 *         14               LOOP ON TEST
1.1180 *         15               REQUEST OUT SN
1.1200
1.1220
```

Figure 4.11. 2914 Header

```
SC001.EDT;G03   31JUL80EOS              DISK NAME:  E21 RKT RADC
DATE  28-AUG-80      TIME  10:05        PAGE    2 OF    2

   1.1240 *  LUN ASSIGNMENTS
   1.1260
   1.1280
   1.1300 *      ASSIGN LUN = 7   FOR .ASC FILE FOR MATRIX TEST RESULTS
   1.1320 *      ASSIGN LUN = 8   FOR .ASC FILE MATRIX
   1.1340 *      ASSIGN LUN = 9   FOR .LOG FILE  FOR FAILURE PARAMETERS
   1.1360 *      ASSIGN LUN = 10  FOR .LOG FILE FOR SAVED ERRORS INFO.
   1.1380 *      ASSIGN LUN = 11  TRACE INFORMATION (CLOSED)
   1.1400 *      ASSIGN LUN = 12  FOR LP
   1.1420 *      ASSIGN LUN = 13  FOR KB
   1.1440 *      ASSIGN LUN = 14  CLOSED
   1.1460 *      ASSIGN LUN = 15  FOR PAPER PUNCH
   1.1480 *
   1.1500
   1.1520 *  . LOAD BOARD              $N/A
   1.1540 *  . DIP ADAPTER             $N/A
   1.1560 *  . CHIP ADAPTER            $N/A
   1.1580 *
   1.1600
   1.1620 *
   1.1640 *
   1.1660 *  . LOAD MODULES
   1.1680 *
   1.1700 *
   1.1720 *       TYPE          LOCATION        SECTOR
   1.1730 *      2914-1          LOAD 1           11
   1.1740 *      5914-2          LOAD 1           5
   1.1760 *      2914-2          LOAD 1           9
   1.1780 *      2914-2          LOAD 1           10
   1.1800 *      2914-2          LOAD 1           17
   1.1820 *      2914-2          LOAD 1           18
   1.1840 *      2914-2          LOAD 1           19
   1.1860 *      2920-2          LOAD 1           20
   1.1880 *      2914-2          LOAD 1           21
   1.1900 *      2914-2          LOAD 1           22
   1.1920 *      2914-2          LOAD 1           23
   1.1940 *      2914-2          LOAD 1           24
   1.1960 *      2914-2          LOAD 1           39
   1.1980 *      2914-2          LOAD 1           40
   1.2000 *      2914-2          LOAD 1           41
   1.2020 *      2914-2          LOAD 1           42
   1.2040 *      2914-2          LOAD 1           43
   1.2060 *      2914-2          LOAD 1           44
   1.2080 *      2914-2          LOAD 1           45
   1.2100 *      2914-2          LOAD 1           46
```

Figure 4.11. 2914 Header, continued

4.3.6 Development of SIMPL Language Device Test

The SIMPL language was developed to minimize tester problems and also simplify the development of device tests. Also the sequence of statements is similar to the steps followed when implementing the test on the bench. The following describes the 2914 LIT test shown in lines 1.08 through 1.2 of the SIMPL file in Figure 4.12.

-- Reset (initialize) all equipment,

-- Connect power supply 6 to the DUT and set it to 5.0V with current limits of 100mA minimum and 300mA maximum,

-- Set the drive levels of the pins denoted by pin array index numbers 1 through 29 and connect them to the DUT with input low set to 0.0V and input high set to 3.0V,

-- Set the compare levels of the pins denoted by index numbers 30 through 49 and connect them to the DUT with the low comparator set to 1.0V and the high comparator set to 2.0V,

-- Connect load module position 1 on the pins denoted by index numbers 30 through 49,

-- Set the system to Mode 3 with a cycle time of 400ns,

-- Set the comparator phase connected to the pins denoted by index numbers 30 through 49 to a delay of 340.0ns with a width of 20.0ns,

-- Set the pin function of the pins denoted by index numbers 1 through 29 to force and inhibit with the pattern in the NRZ mode,

-- Set the pin function of the pins denoted by index numbers 30 through 49 to mask and compare with the pattern,

-- Execute the LIT test on the pins denoted by index numbers 1 through 49, applying vectors 1 through 512 with no dummy patterns (wait for additional vectors).

The following describes the implementation of a portion of the 2914 DC tests (reference Figure 4.12, lines 1.22 through 1.38):

-- Set up for a pattern move on the pins denoted by index numbers 1 through 29, applying vectors 1 through 204 with no additional vectors and no logging allowed (sets the pattern move flag),

-- Measure output voltage high on the pins denoted by index numbers 30 through 49 after moving the pattern (if the flag is set) by forcing -1.0mA with limits of 2.4V minimum and 5.0V maximum,

-- Set up a pattern move on index number 0 (this clears the pattern flag for any subsequent measurement),

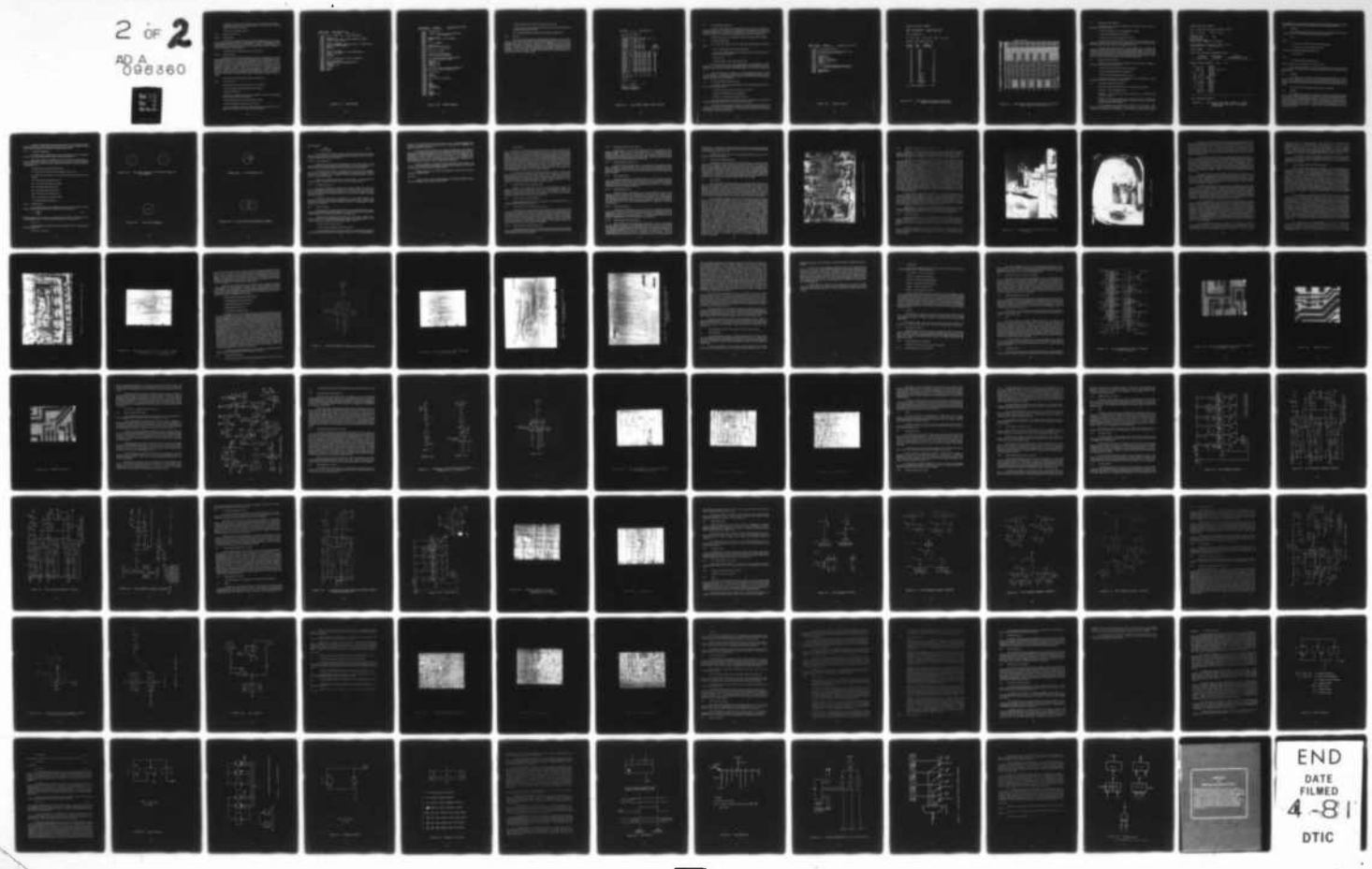
```
S2914.ASC:G03   31JUL80GEOS          DISK NAME: E21 BKT RADC
DATE  27-AUG-80     TIME  13:41      PAGE   1 OF   1

1.0100 *   TEST FOR THE 2914 DEVICE
1.0200
1.0300   2914 DEVICE TEST
1.0400
1.0500   INIT
1.0600
1.0700 * LOGIC INTEGRITY TEST
1.0800   PWRSUP  6  1  5.0  100.0MA  300.0MA
1.0900   DRILEV  1   29  1  0.0V   3.0V
1.1000   CMPLEV  30  49  1  1.0V  2.0V
1.1100   CONLD   30   49   1    1
1.1200   CYCLE   3 400.0NS
1.1300   PHASE   30  49  56  340.0NS  20.0NS
1.1400   PINFNT  1   29  12     2     0
1.1500   PINFNT  30  49  34     2     0
1.1600   EXCTST  1   49  1      512   2
1.1700   EXCTST  1   49  517    1024  2
1.1800   EXCTST  1   49  1025   1506  2
1.1900   EXCTST  1   49  1507   2000  2
1.2000   EXCTST  1   49  2001   2156  0
1.2100
1.2200 * OUTPUT VOLTAGE HIGH
1.2300   PATPAR  1 29  1        13        0
1.2400   MVOH   30 37 -1.0MA    2.4V      5.0V
1.2500   PATPAR  1 29  1        430       0
1.2600   MVOH   38 40 -1.0MA    2.4V      5.0V
1.2700   PATPAR  1 29  1        415       0
1.2800   MVOH   41 43 -1.0MA    2.4V      5.0V
1.2900   PATPAR  1 29  1        204       0
1.3000   MVOH   44 44 -1.0MA    2.4V      5.0V
1.3100   PATPAR  1 29  1        3         0
1.3200   MVOH   45 48 -1.0MA    2.4V      5.0V
1.3300   PATPAR  1 29  1        204       0
1.3400   MVOH   49 49 -1.0MA    2.4V      5.0V
1.3500   PATPAR  0
1.3600
1.3700 * ICC TEST
1.3800   MICC    1  6  5.5V   15.0MA  310.0MA
1.3900
1.4000 * END OF TEST
1.4100   FINIS
```

Figure 4.12. 2914 SIMPL File

AD-A096 360 GENERAL ELECTRIC CO PITTSFIELD MA ORDNANCE SYSTEMS F/G 9/2
TEST GENERATION AND FAULT ISOLATION FOR MICROPROCESSORS AND THE--ETC(U)
NOV 80 W H DEBANY, D A O'CONNOR, B K TEAGUE F30602-78-C-0235
UNCLASSIFIED RADC-TR-80-274 NL

2 OF 2
AD A 096360



END
DATE FILMED
4-81
DTIC


A
963

-- Measure ICC on the pins denoted by power pin array PVCC index number 1, using power supply 6 with a forcing voltage of 5.5V and limits of 15.0mA minimum and 310.0mA maximum,

-- End of test (reinitialize system).

4.3.7 Setup to Log LIT Errors

4.3.7.1 C0DE Routine

The C0DE routine was developed to initialize the logging key as well as check the Test Station Control Unit (TSCU). The logging key is required for subsequent failure data analysis routines and is reset each time C0DE is executed. A translated version of C0DE should be stored under the DUT UID and is executed from the TSCU. (Figure 4.13 shows an example of the C0DE program)

The C0DE routine is described briefly in section 3.2.4.

4.3.7.2 BATLOG Routine

The BATLOG Routine was developed to setup the LUN assignments required by the SIMTEK programs. The ".ASC" file containing the device SIMPL program and the ".TST" file containing the translated SIMTEK program are to be specified by the user in the BATLOG file, which is then saved under the device UID. The program is run by depressing "Control-L" on the keyboard before running the device test. This operation clears the data files, stores the pinlists from the ".TST" file in ERRPAT.LOG, and assigns the peripheral devices. After the device testing is complete, depressing "Control-L" closes all files for later processing and assigns all LUN's to the keyboard. Note that all previously existing CONTRL.LOG and ERRPAT.LOG files will be lost. If these files contain data to be saved, they must be renamed before the first Control-L is typed.

An example of a BATLOG Routine is shown in Figure 4.14.

4.3.8 Testing a Device

The sequence of operations to test a device is as follows:

-- Insert disk, socket card assembly and load modules,

-- Enter the device UID,

-- Execute the C0DE routine by dialing "C0DE" on the TSCU, depressing START and typing the log key number,

-- Type "Control-L" (this starts the BATLOG routine),

-- Dial the appropriate test number in the TSCU,

-- Set the computer switches to their appropriate positions (all off will display a switch option menu when START is depressed),

```
  PROGRAM  CODE    TEST INITIALIZATION
DATE  27-AUG-80     TIME  15:48        PAGE   1 OF   1

   1.0100 * .TITLE,  PROGRAM  CODE    TEST INITIALIZATION
   1.0200 *
   1.0300   COMMON KEY
   1.0400   ACCEPT<13> ERASE,"INPUT OR RESET LOGGING KEY ",KEY,CR
   1.0500   IF (KEY GE 500) 1.13,1.06
   1.0600 *
   1.0700   LOOP 1.1 I = 1,50,1
   1.0800   PRINT<13> "COMPONENT LOGGING KEY IS SET TO: ',KEY:I3,"↑K",CR
   1.0900   DISPLAY 0,PASS,WITHIN,BELOW
   1.1000   CONTINUE
   1.1100   GO TO 1.19
   1.1200 *
   1.1300   LOOP 1.16 I = 1,50,1
   1.1400   PRINT<13> "KEY MUST BE IN 0 - 500 RANGE ↑G↑K",CR
   1.1500   DISPLAY 1,FAIL,ABOVE
   1.1600   CONTINUE
   1.1700   GO TO 1.04
   1.1800 *
   1.1900   PRINT<13> CR,CR,"CHECK OUT T.S,C.U. LIGHTS",CR
   1.2000 *
   1.2100   LOOP 1.25 I=1,1500,1
   1.2200   DISPLAY 88,PASS,FAIL,ABOVE,BELOW,WITHIN
   1.2300   WAIT 2MS
   1.2400   IF (ADVANCE) 1.26
   1.2500   CONTINUE
   1.2600 *
   1.2700   DISPLAY 0
   1.2800   STOP
```

Figure 4.13. CØDE Program

```
BATLOG.ASC;G03   3JUL80GE03          DISK NAME:  E21 BKT RADC
DATE  27-AUG-80     TIME  13:39      PAGE   1 OF   1

1.0100 *   PROGRAM BATLOG
1.0200 *
1.0300 *   PURPOSE:  TO ASSIGN L.U.N.S AND LOG FILES
1.0400 *             PRIOR TO TESTING.
1.0500 *
1.0600   #MESSAGE,↑[↑G↑L
1.0700   #MESSAGE, : TURN ON THE LINE PRINTER
1.0800   + ↑G
1.0900 *
1.1000 *   L.U.N. ASSIGNMENT
1.1100 *
1.1200   #ASSIGN * = KB
1.1300   #ASSIGN 7=SIMTEK.ASC/DE/O:100
1.1400 *
1.1500 * ASSIGN LUN 8 TO TEST.ASC FILE
1.1600 *
1.1700   #ASSIGN 8 = S2914.ASC
1.1800 *
1.1900   #ASSIGN 9=CONTRL.LOG/DE/O:25
1.2000   #ASSIGN 10=ERRPAT.LOG/DE/O:200
1.2100 *
1.2200 * NOTE: DISK SPACE FOR ABOVE FILES SHOULD BE
1.2300 *       VARIED ACCORDING TO USAGE.
1.2400   #CLOSE 11
1.2500   #ASSIGN 12 = LP
1.2600   #ASSIGN 13 = KB
1.2700   #CLOSE 14
1.2800   #ASSIGN 15 = PP
1.2900 *
1.3000 * THIS SECTION STORES THE PINLIST CONTAINED IN
1.3100 * THE .TST FILE IN THE ERRPAT.LOG FILE
1.3200 *
1.3300   #MESSAGE,↑[↑G↑L
1.3400 *
1.3500   #RESET *
1.3600   #PINLISTS,10
1.3700 *
1.3800 * INSERT .TST FILE NUMBER HERE
1.3900 *
1.4000   2914
1.4100 *
1.4200   #FILES
1.4300   #BEXIT
1.4400   #HISTORY,KB
1.4500   #MESSAGE,↑[↑G↑L
1.4600   + ↑G
1.4700 *
1.4800   #FILES
1.4900   #CLOSE *
1.5000   #ASSIGN * = KB
1.5100   #EXIT
```

Figure 4.14. BATLOG Program

– Depress START on the TSCU to execute the device test,

– Type "Control-L" after testing all devices that use this test number (this completes the BATLOG routine).

The final SIMTEK program is shown, after execution, in Figure 4.15.

4.3.9 Summary

With the development of RADC's TEKTRONIX and SENTRY Pattern routines as well as the creation of the SIMPL statements (see section 3), it is now feasible to transport device tests among test systems with considerably less difficulty than previously experienced. Test program development is simple and results in structured programs. The interpreter program for one series of test systems (S-3260/3270) has been developed. Device tests have been developed using the SIMPL statements and are executable on four test systems. Special programs (BATLOG, C0DE, BATRED, etc.) simplify testing and aid in failure analysis.

```
SIMTEK.4SC1G03                          DION HOME1  E21 OKT R40C
DATE  27-AUG-80      TIME  14:10        PAGE   1 OP   2

TITL        OUT 0 / M =    200.0

  1,010    *  TEST FOR THE 2910 DEVICE
  1,050 INIT

  1,070    *LOGIC INTEGRITY TEST
  1,080 PWRSUP    0    1   5.040 V  100.0MA  300.0MA  105.0MA      P000
  1,090 DWILEV    1   29   1.400    0.000    3.000
  1,100 CMPLEV   30   49   1.400    1.000    2.000
  1,110 COMLD    30   49   1.400    1.000
  1,120 CYCLE         3   400.0MSEC
  1,130 PHASE    30   49   56.00    340.0N   20.00N
  1,140 PIMPMT    1   29   12     2    0
  1,150 PIMPMT   30   49   30     2    0
  1,160 EXCTST    1   49   1.000    512.0    2.000                  P003
  1,170 EXCTST    1   49   517.0    1.024K   2.000                <<< FAIL >>>
  1,180 EXCTST    1   49   1.025K   1.536K   2.000                <<< FAIL >>>
  1,190 EXCTST    1   49   1.537K   2.000K   2.000                <<< FAIL >>>
  1,200 EXCTOT    1   49   2.001K   2.156M   0.000                <<< FAIL >>>

  1,220    *OUTPUT VOLTAGE HIGH
  1,230 PATPAR        1      29       1      13       0
  1,240 HVOH     30   19   -1.000MA  2.000 V  5.000 V  220.0MV <<< FAIL >>>
  1,240 HVOH     31   21   -1.000MA  2.400 V  5.000 V  3.460 V      P033
  1,240 HVOH     32   23   -1.000MA  2.400 V  5.000 V  3.455 V      P033
  1,240 HVOH     33   25   -1.000MA  2.400 V  5.000 V  3.455 V      P030
  1,240 HVOH     34    2   -1.000MA  2.400 V  5.000 V  3.465 V      P030
  1,240 HVOH     35   40   -1.000MA  2.400 V  5.000 V  3.460 V      P030
  1,240 HVOH     36   38   -1.000MA  2.400 V  5.000 V  3.455 V      P030
  1,240 HVOH     37   36   -1.000MA  2.400 V  5.000 V  3.455 V      P030
  1,250 PATPAR        1      29       1     030       0
  1,260 HVOH     38   11   -1.000MA  2.400 V  5.000 V  3.465 V      P030
  1,260 HVOH     39   12   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,260 HVOH     40   13   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,270 PATPAR        1      29       1     015       0
  1,280 HVOH     41   16   -1.000MA  2.400 V  5.000 V  3.475 V      P003
  1,280 HVOH     42   17   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,280 HVOH     43   10   -1.000MA  2.400 V  5.000 V  3.470 V      P000
  1,290 PATPAR        1      29       1     204       0
  1,300 HVOH     44    3   -1.000MA  2.400 V  5.000 V  3.480 V      P030
  1,310 PATPAR        1      29       1       3       0
  1,320 HVOH     45   15   -1.000MA  2.400 V  5.000 V  3.470 V      P000
  1,320 HVOH     46   14   -1.000MA  2.400 V  5.000 V  3.470 V      P000
  1,320 HVOH     47    9   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,320 HVOH     48    8   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,330 PATPAR        1      29       1     204       0
  1,340 HVOH     49    7   -1.000MA  2.400 V  5.000 V  3.465 V      P000
  1,350 PATPAR        0       0       0       0       0

  1,370    *ICC TEST
  1,380 HICC      1   10   5.500 V  15.00MA  310.0MA  203.0MA      P000

  1,400    *END OF TEST
  1,410 FINIS


2910 DEVICE TEST    0/M =   200.0


  FAILED      5    TESTS OUT OF       26    TESTS

   TEST TIME =      12.70  SEC.

  GENERAL ELECTRIC ORGANIZATIONAL SYSTEMS
ELECTRONIC TEST AND EVALUATION CENTER
       31 JUL 80      23:15:06
```

Figure 4.15. 2914 SIMPL Program, after running

5. LIT Failure Data Reduction

The LIT failure data stored by SIMTEK (See section 3.2.3) are read by the following programs. The data are either presented to the operator or processed in conjunction with the error dictionary to indicate the cause of the failure. Also, the operator may display the test pattern as timing waveforms for easier interpretation.

These programs run on the S-3260/3270 under the system REDUCE program.

5.1 Displaying Logged Data

The operator may observe the LIT failure data by performing the following steps:

— Enter the UID under which the data are stored,

-- Verify that a copy of BATRED.ASC is stored under the above UID (if not, copy it from the UID ":SIM"),

-- Type CTRL-R,

-- Follow the instructions presented on the CRT.

The above steps execute the BATRED.ASC and DATRDB.EDT routines described in sections 3.2.6 and 3.2.7, respectively. Figure 5.1a shows an example of the BATRED routine. Figure 5.1b shows the DATRDB output for the number of failures per pin.

When the program first requests the key, the operator must enter the first key in the log files. On subsequent requests, any key in the CONTRL.LOG file may be entered. The keys may be obtained by running the S-3260/3270 ANALYZE program, which is described in the system manuals.

5.2 Plotting the Pseudo Timing Waveforms

The operator may display the LIT pattern file as pseudo timing waveforms by performing the following steps:

— Enter the UID under which the pattern file is stored,

-- Type "RUN PATPLO:SIM" on the keyboard,

-- Follow the instructions presented on the CRT.

The above steps execute the PATPLO.ASC, PATPLO.EDT and WAVE.EDT routines described in sections 3.2.8, 3.2.9 and 3.2.10, respectively.

The sample waveform shown in Figure 5.2 was plotted from the pattern file in Figure 2.2. The user is expected to know from the pin names whether a given line represents an input pin or an output pin. In the areas marked with slash lines, the inputs are inhibited or the outputs are masked.

```
BATRED.ASC;G03   30JUN80GEOS          DISK NAME:  E21 BKT RADC
DATE  27-AUG-80     TIME  13:40      PAGE   1 OF   1

   1.0100 * BATRED ASSIGNMENT PROGRAM
   1.0200 *
   1.0300 * THIS ROUTINE SETS UP LUN ASSIGNMENTS FOR THE
   1.0400 * DATA REDUCTION ROUTINE IN :G01
   1.0500 *
   1.0600   #ASSIGN * = KB
   1.0700   #HISTORY,KB
   1.0800   #ASSIGN * = KB
   1.0900   #ASSIGN 9 = CONTRL.LOG
   1.1000   #ASSIGN 10 = ERRPAT.LOG
   1.1100   #CLOSE 11
   1.1200   #ASSIGN 12 = LP
   1.1300 *
   1.1400 * NOTE: FOR FURTHER PROCESSING OF FAILED BITS
   1.1500 *       INFORMATION ASSIGN LUN 15 TO PP OR MT
   1.1600 *
   1.1700   #ASSIGN 15 =
   1.1800   #RUN DATRDB:SIM
   1.1900   #EXIT
```

Figure 5.1a. BATRED Program

GENERAL ELECTRIC COMPANY

DATA REDUCTION FOR 2914 DEVICE TEST
DATE: 22 SEP 80 TIME: 16:39:42

DATA LOGGING KEY = 3

LOG FILE DATE: 29 AUG 80 TIME: 09:27:46

NUMBER OF FAILURES PER PIN

COLUMN NUMBER	PIN NAME	NUMBER OF FAILURES
30	M07	0
31	M06	0
32	M05	0
33	M04	0
34	M03	0
35	M02	0
36	M01	0
37	M00	0
38	S02	13
39	S01	12
40	S00	12
41	V2	87
42	V1	86
43	V0	84
44	GPSIG	72
45	GAS	49
46	STOVFL	66
47	IRQ	717
48	PARDIS	45
49	RIPDIS	43

DUT SERIAL NUMBER IS 1.000

Figure 5.1b. Data reduction output, showing the number of failures on each pin

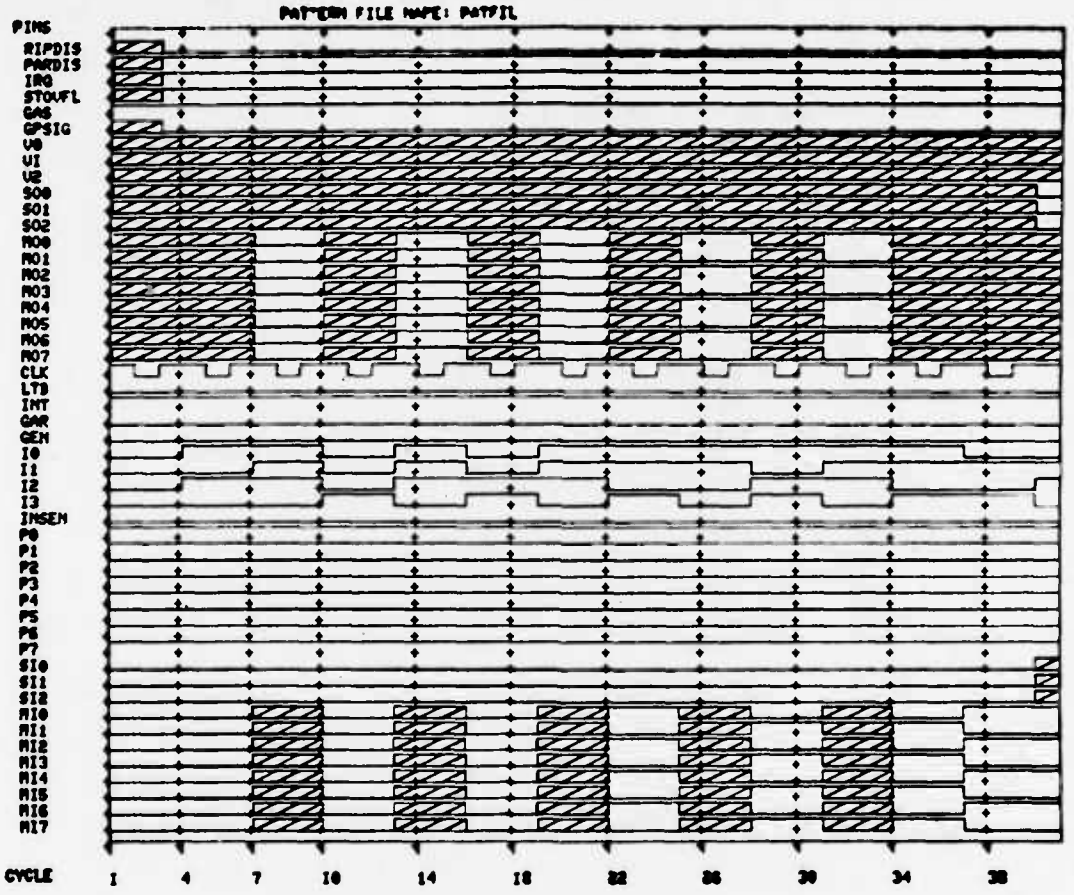


Figure 5.2. WAVE Output, showing the first forty vectors of the 2914 LIT (from Figure 2.2)

5.3 Isolating the Failed Element

The operator may obtain an indication of the cause of an LIT failure by performing the following steps:

-- Enter the UID under which the logged data is stored,

-- Type "RUN FISO:SIM" on the keyboard,

-- Follow the instructions presented on the CRT.

The above steps execute the FISO.ASC, FISOV1.ASC and FISO.EDT routines described in sections 3.2.11, 3.2.12 and 3.2.13, respectively. Note that the restriction on entering keys described in section 5.1 also applies to this section.

A sample fault isolation printout is shown in Figure 5.3. This figure was obtained by testing a 2914 with a known fault using the SIMTEK program. Then the fault isolation routines were performed using the data logged during testing. Note that the printout list several points which may have caused the failure. Also, the points are listed in a coded format that was derived by LASAR from the device model. In order to fully interpret the printout, the user must have the compiled LASAR model.

5.4 Reformatting the YTABLE

The operator may reformat the YTABLE or print the XTABLE, YTABLE, or ZTABLE by performing the following steps:

-- Enter the UID under which the tables are stored,

-- Type "RUN XYZPRT:SIM" on the keyboard,

-- Follow the instructions presented on the CRT.

The above steps execute the XYZPRT.ASC and XYZPRT.EDT routines described in sections 3.2.14 and 3.2.15, respectively.

5.5 Fault Signature Matching Algorithm

The fault dictionary generated by LASAR is divided into three parts:

XTABLE -- A list of "significant bits,"

YTABLE -- A list of possible faults associated with a particular "fault isolation set number,"

ZTABLE -- A list of fault isolation set numbers and the first ten significant bits in the "fault signature" exhibited by a device with one of the corresponding faults from the YTABLE.

The fault signature from a bad device is compared with the XTABLE. Those entries that match form the "significant bit fault signature." This signature is then compared to the signatures in the ZTABLE. This results in a ranking of the most likely

```
GENERAL ELECTRIC COMPANY

FAULT ISOLATION ROUTINE FOR BOARD # AM2914

DATE: 22 SEP 80      TIME: 16:44:03

NUMBER OUTPUTS    =      20
DYSOGEN LIMIT     =      10
DATA LOGGING KEY  =       3
DATE OF LAST LASAR RUN - 02/04/80

LOG FILE DATE:  29 AUG 80  TIME:  09:27:46
FROM DEVICE TEST  2914 DEVICE TEST

SERIAL NUMBER     1.000

FAULT ISOLATION CROSS REFERENCE TABLE

          FAILURE       REPLACEABLE           ALTERNATE
        DESCRIPTION      PACKAGE          REPLACEABLE PACKAGE
-------------------------------------------------------------------

PERFECT MATCH DETECTED    YTAB =    330

:   15-  9 (1)      QRPK15
;      240  (1)      QIPK8
;      253 (0)      QSPK12  PK15  PK7
;   12-  3 (0)      QSPK12
;   12-   2 (0)      QSPK12
;   15-  8 (0)      QRPK15
;   15-  5* 15-  8 QRPK15
;    8- 61 (0)  QIPK8
;   12-  4 (1)      QSPK12
;   12-  4*253      QSPK12
;      240* 12-  3 QSPK12
;    9-  7 (0)      QLPK9
;    9-  8 (0)      QLPK9
;    9-  9 (0)      QLPK9
;      248 (1)      QLPK9
;      248* 12-  2 QSPK12

-------------------------------------------------------------------

FAULT ISOLATION COMPLETE
```

Figure 5.3. Fault isolation output, showing the LASAR YTABLE nodes that could have caused the failure

fault isolation set numbers. The YTABLE then translates these set numbers into lists of gate level faults. A more detailed description of each of the tables will follow.

5.5.1 XTABLE

The XTABLE contains a list of bits considered to be significant by LASAR, in a coded form. Bits in a particular fault signature are expressed as

R,C,

where

R = vector set row number in which a failure occurred,

C = output column in which a failure occurred.

A number L is calculated according to the formula

$$L = (R-1) * W + C,$$

where

W = width (number) of output columns.

The "bit number" or N is the location of L in the XTABLE.

When using the XTABLE to encode the fault signature, not every resulting L will be found. In that case, they are simply omitted; they are not significant fault signature bits.

5.5.2 YTABLE

The YTABLE is a set of lists of LASAR model nodes and failures. For each of many "fault isolation set numbers," one or several nodes and failures will be listed. With the given input vector set, and a user-specified limitation on the length of fault signatures (to be discussed in section 5.5.3), these faults are indistinguishable from each other.

The fault isolation set number is obtained from the ZTABLE.

5.5.3 ZTABLE

The ZTABLE associates a fault isolation set number with lists of bit numbers, or "Z-entries." Ideally, there would be a Z-entry for each set of indistinguishable faults. These could be truly indistinguishable, as would be certain faults in a counter chain, or it may be that the input vector set was able to detect the faults, but was unable to provide sufficient resolution to discriminate. In practice, memory limitations force a user to set a limit on the length of the Z-entries. A usual option during REDUCE is "FILE 10," setting a limit of 10 bits per Z-entry. A larger limit would increase resolution, but would also increase run time.

Using the ZTABLE requires the matching of the user's fault signature against each of the Z-entries. If a perfect match is found, the user is in luck. However, the user must usually be satisfied with a partial (and hopefully, good) match.

5.5.3.1 The Matching Algorithm

The simple case, in which there was no artificial limitation on the length of the Z-entries, will be considered first, after a discussion of notation.

Referring to Figure 5.4a through 5.4d, it may be seen that there are four cases of interest (noting that 5.4c has an equivalent case, that is, S may be contained in Z). We wish to derive a numerical index, called G, that will allow a ranking according to a "goodness of fit."

The notation that will be used is as follows:

S = The set of bit numbers in the fault signature (recall 5.5.1),

Z = The set of bit numbers in the particular Z-entry being checked at the moment,

SZ = The intersection (AND) of S and Z,

SvZ = The union (inclusive OR) of S and Z,

[S] = The number of bit numbers in S,

[Z] = The number of bit numbers in Z,

[SZ] = The number of bit numbers in SZ,

[SvZ] = The number of bit numbers in SvZ,

+,- = ordinary addition, subtraction.

5.5.3.1.1 The Ideal Case

The simplest measure of the goodness of fit is to let G equal the ratio of the number of bits in the intersection to the number of bits in the union, or

$$G = \frac{[SZ]}{[SvZ]} \qquad (5.1)$$

This would yield G = 0 when there is no overlap (Figure 5.4a), and G = 1 when S and Z are identical (Figure 5.4b). For the cases depicted in Figure 5.4c and Figure 5.4d,

$0 < G < 1.$

When calculating this quantity, there is a desirable shortcut. Using the easily demonstrated identity

[SvZ] = [S] + [Z] - [SZ],

Figure 5.4a. The fault signature (S) and the Z-entry (Z) are disjoint

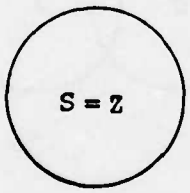


Figure 5.4b. S and Z are identical

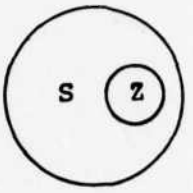


Figure 5.4c. Z is contained in S

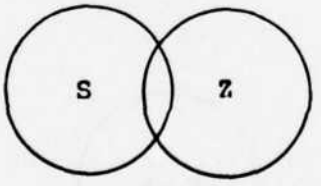


Figure 5.4d. S and Z have some elements in common

Eq. (5.1) becomes

$$G = \frac{[SZ]}{[S] + [Z] - [SZ]} \qquad (5.2)$$

On a computer this quantity is found considerably faster using Eq. (5.2) rather than Eq. (5.1), as [S] and [Z] are usually already known, and [SvZ] will only have to be calculated, rather than found by enumeration.

5.5.3.1.2 The Non-Ideal Case

If there were no limitations on the length of the Z-entries, then Eq. (5.2) would be sufficient to give the relative rankings of each Z-entry. Unfortunately, it is in fact necessary to avoid penalizing Z-entries that have been "unfairly" truncated.

The method used, when [Z] = 10 (maximum size), is to look at the largest N (bit number) in the particular Z-entry under examination. Temporarily, each N in S that is greater than the largest N in Z is discarded, and [S] is adjusted accordingly. Now Eq. (5.2) can be used and G obtained.

Again, this is only done when [Z] = 10 (assuming FILE 10 was used) under the assumption that Z was indeed truncated. When [Z] < 10, S is left untouched.

5.5.3.2 The Significance of G

It must be stressed that G allows only a relative ranking of Z-entries and their corresponding fault isolation set numbers (recall section 5.5.2). One cannot say that G = 0.8 indicates twice as good a match as G = 0.4. Usually, the top ten Z-entries are extracted, and all of their nodes and faults in the YTABLE are considered as possible candidates for failure.

The list of nodes from the YTABLE is not to be taken literally. It is necessary to remember the functions represented by the nodes, and to interpret that information into the necessary physical locations in the DUT. This philosophy is outlined in section 1 of this report.

5.5.4 Other Techniques

The literature is conspicuously barren of practical techniques which can be used for the purpose described in this section. The only one found was a method which was presented at the 1979 Cherry Hill IEEE Test Conference [1].

The algorithm, as described in the paper, uses a scale where a perfect match is given a score of "0" and faulty matches have integral scores ranging upward. The score is obtained by summing the following penalty points, for each Z:

-- Add 1 for each Z-entry not in S,

-- Add 3 for each bit number in S which is not in Z.

This technique was evaluated during this effort by making another entrypoint in the Multics routine "SEARCH" (not described in section 3.1, as it only duplicates the

function of the "FISO" described in sections 3.2.10 and 5.4). Taking advantage of the existing PL/I program's data structures and I/O, it was simple to implement the algorithm described in Reference 1.

The results were interesting. The relative rankings given by this algorithm almost always matched those given by the one described by Eq. (5.2) used in the non-ideal case (i.e., discarding signature bits when the number of Z-entries equals 10). The differences that were observed in the few test cases run for comparison were attributable to the fact that consideration was not given to the truncated fault signatures (due to "FILE 10"). This was particularly evident when the signature and Z-entry would have been a perfect match if Z had not been truncated.

When this algorithm was modified to take advantage of the knowledge of the non-ideal case, results were more consistent with the actual faults inserted. Still, it was not quite as accurate as the Eq. (5.2) implementation.

Relative execution time varied from slightly less to 50% more than that of the Eq. (5.2) implementation.

References:

1. Joseph F. Weiss, "Fault Isolation Speedup For LSI Boards," Digest of Papers 1979 Cherry Hill IEEE Test Conference, pp. 186-188.

6. Fault Insertion

In order to demonstrate the effectiveness of the device models and to achieve confidence in the fault isolation programs, controlled faults have been physically inserted in the test devices. This involved logically locating a suitable site for faulting, tracing the circuitry on the device to identify the corresponding physical location, and physically inserting the fault at this site. A test program generated from a good model of the device type, when applied to a physically faulted device, generated data that were used by the fault isolation software to identify where in the logical model the fault occurred. Model completeness, fault coverage, and fault isolation program effectiveness could thus be assessed. "False Modeling" was also done. The false modeling technique simulates physical fault insertion by developing a model with a known fault. In this case a test program generated from a faulted model of the device type, when applied to a good device, generated the data for fault isolation software.

Five device types underwent the physical faulting procedures: The Advanced Micro Devices 25LS2517 (Arithmetic Logic Unit/Function Generator) and the 2914 (Vectored Priority Interrupt Encoder); Harris Semiconductor 15530 (CMOS Manchester Encoder/Decoder); the Motorola 6821 (NMOS Peripheral Interface Adapter); and the Advanced Micro Devices 2910 (Microprogram Controller). Three faults per device type were inserted except for the 6821, which had two faults inserted.

6.1 Fault Site Selection/Circuit Descriptions

Because of the complexity of the device involved, complete tracing of the physical circuitry of each device would be very time consuming and therefore was undesirable. The first step in faulting a device, then, was analysis of the logic and selection of sites for fault insertion from these analyses. In this way, physical tracing of the device in question was limited to the areas of concern.

6.1.1 Fault Site Selection Considerations

Several considerations were taken into account in selecting fault sites. The following paragraphs discuss these factors.

6.1.1.1 Faults that Affect Mutually Exclusive Outputs

Selecting multiple faults that affected mutually exclusive outputs guarantees that the circuitry affected by each fault was isolated and that more than one fault could be inserted on a single IC. This was only possible on two device types examined here. The complexity of the design of all other device types inhibited multiple faulting unless extremely elementary sites were chosen (e.g., the buffering circuitry to an output). Because they were more challenging it was decided that more complex faults were necessary to adequately test the fault isolation programs. Therefore, several outputs are usually affected by each of the faults that were inserted.

6.1.1.2 Faults that Affect Diverse Logic Areas

By placing faults in different functional areas of a device type, more of the model is tested for accuracy and feasibility. Also, a wider range of situations is provided for exercising the fault isolation program.

6.1.1.3 Faults That Affect Limited Circuitry

A gross fault is one that when introduced into a device produces a fault isolation program signature that indicates possible faults in a wide-spread area (all affected circuitry). The ambiguity of such results masks the functioning of the fault isolation program. Because of this, gross faults were avoided. This problem was encountered on several device types that contained long, chained sequences in the logic or a large amount of interdependence of logic functions.

6.1.1.4 Ease of Physical Site Location

In order to have confidence that the sites identified on the physical device truly correspond to the selected logic sites, it is necessary to choose locations within unique or readily traceable circuitry. This does not exclude imbedded circuitry but directs site selection to areas with distinctive characteristics (e.g., an eight-input NOR gate in the 2914).

6.1.2 Circuit Descriptions

Fault site selection was accomplished using a variety of technical materials containing information and descriptions of the circuits involved. The materials available varied from device to device and from manufacturer to manufacturer, creating an imbalance in the amount of time devoted to faulting each device type. The materials that proved to be most useful are listed below.

6.1.2.1 Published Data Sheets

This source gives a brief description of the operation of the device, block diagrams, electrical characteristics, package specifications, absolute ratings, and pin assignments. It was found that the data sheets often contained errors in the descriptions of the operation of the devices and in the flow of control shown in the block diagrams. They also varied greatly as to the detail of logic design given. Therefore, this material was used only for an initial, basic understanding and as a reference on electrical characteristics.

6.1.2.2 Automaton Diagrams

In some cases, these diagrams were the only logic-flow diagrams available for site selection. The fact that the functional blocks of an Automaton Diagram encompass large areas of circuitry led to some confusion and ambiguity in pinpointing the exact location of a fault site on the physical device. For more detailed explanation of the generation and application of Automaton Diagrams, see Appendix A.

6.1.2.3 RADC Product Evaluation Reports

When available, these reports greatly reduced the amount of time required for fault insertion on a device. The report analyzes the device on both a physical and electrical level. A logic schematic is developed directly from the physical layout of the circuit and the correspondence between the two is given. Therefore, fault site selection was simplified because for each schematic is detailed to the transistor level. There is a corresponding physical structure shown (via micrographs and other figures). There are other disclaimers in the reports as to the accuracy of the logic circuits because of the

complications in identifying and translating physical layouts of LSI devices to transistor level schematics. Verification of fault sites before faulting was done using SEM voltage contrast techniques (see section 6.2).

Unfortunately, these reports are on Restricted Distribution, and so are not directly referenced in this report.

6.1.2.4 Manufacturer's Logic Schematics

Availability of accurate manufacturers' logic schematics would have greatly aided fault site selection and tracing of the physical circuitry to the fault site. Definitive logic diagrams would have eliminated much of the confusion resulting from errors in published data sheets and from insufficient information. The complexity of the devices and the methods used in their fabrication (e.g., Schottky technology, multilayer metallization) led to problems in circuit tracing because of the difficulty in identifying individual logic components. Logic schematics would have solved these problems and would have led to quicker fault site selection and identification as was evidenced by the schematics found in the Product Evaluation Reports that were available.

6.2 Fault Tracing

The major problem in fault insertion is uncertainty that the location of the fault on the physical device corresponds to the logic site selected. When this work was initiated, an attempt was made to trace the physical circuitry of a device to the proposed fault location using optical micrographs of the die enlarged to 400X. This method of identifying device structure and interconnections worked well on the 25LS2517 because of its relatively simple design and its clean and open layout. However, the succeeding devices encountered in this effort were found to be too complex and dense to allow visual inspection of the circuit as an adequate method for fault location.

Voltage contrast techniques using the Scanning Electron Microscope (SEM) proved to be a very effective method of circuit tracing. The SEM operates by scanning across the surface of the sample (in this case an IC) with a finely collimated electron beam that has a variable accelerating voltage potential (200V-39KV) controlled by the operator. Due to the interaction of the beam with the specimen, many complex physical processes occur producing several types of signals that image the specimen. Some of these include secondary electrons (electrons emitted with energies less than 50eV), back scattered electrons (high energy electrons), x-rays, and absorbed electrons. Voltage contrast techniques are applied using secondary electron imaging which presents topographic, voltage and magnetic variations in the specimen. The secondary electrons are sensitive to the effects of locally applied potentials. In other words, when a potential of +5V (the power level for most LSI devices) is applied to the specimen, electrons with energies less than 5eV cannot escape the specimen surface. Conversely, when a ground potential is applied to the specimen, all of the emitted electrons are reflected up to the detector. The resulting image of the areas at ground potential appears brighter than that of the areas at a positive potential. This brightness variation is known as voltage contrast. It lends itself well to tracing physical circuitry of an IC. A pattern of voltage high and low potentials (+5V and ground) can be applied to the DUT in order to exercise an area of interest on the physical device. The voltage contrast mode creates an image of recognizable highs (darker areas) and lows (brighter areas) throughout the circuitry to be traced. Figure 6.1 is an example of voltage contrast effects on a 25LS2517.

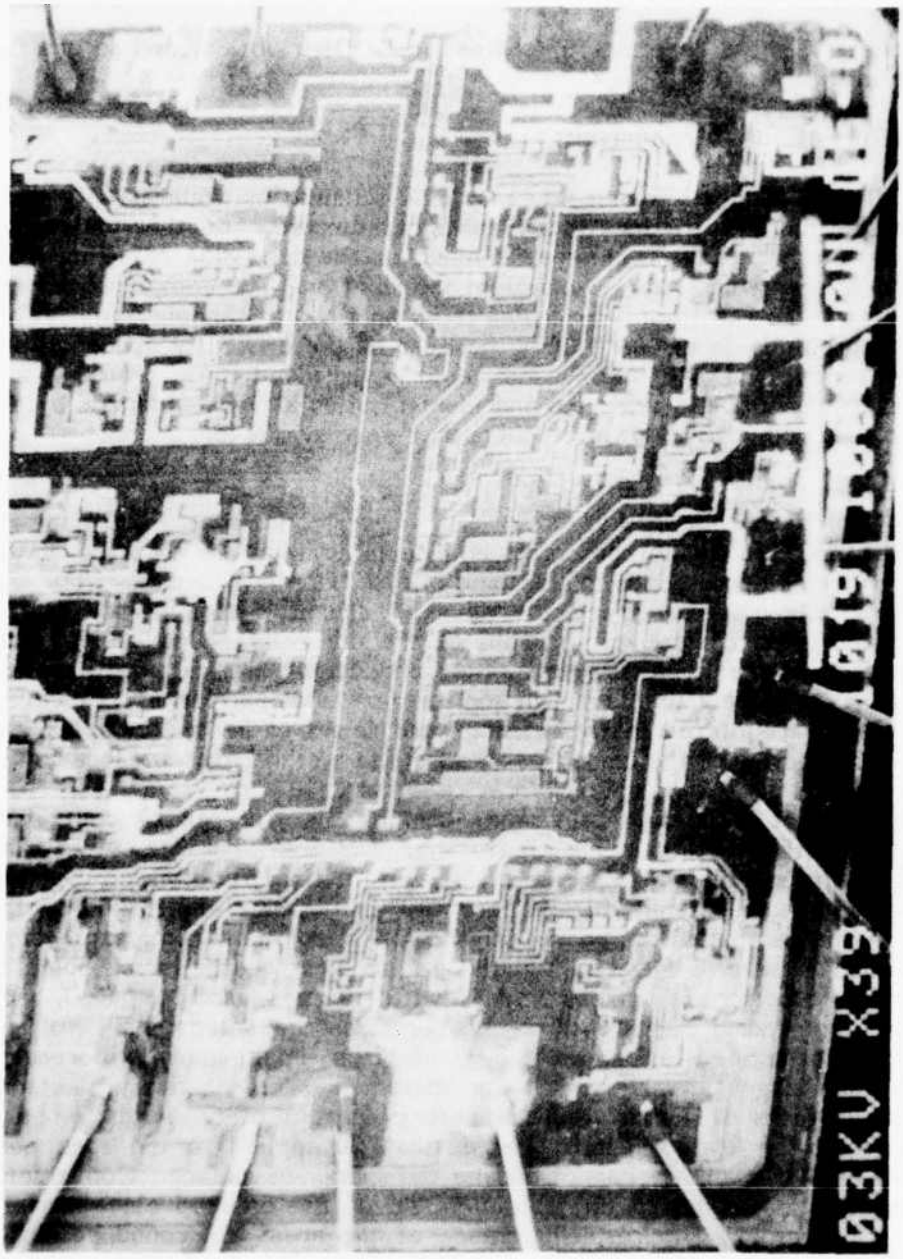


Figure 6.1. Voltage contrast effects on the 25LS2517

6.2.1 Equipment, Support Circuitry, Adapters

Figure 6.2 shows the mainframe SEM and support equipment used for this project. The microscope, a Japan Electron Optics Laboratory JSM-35C, is capable of producing spot sizes of 60 Angstroms. Magnification capabilities range from 10X to 180,000X. The vacuum is maintained at 3E-7 to 5E-7 torr to minimize contamination.

The facility is equipped to perform dynamic voltage contrast studies (applying changing levels or pulses instead of static patterns to the inputs of the device) utilizing a TV scan generator, multiphase clocks, pulse generators, memory exorcisor, and a time lapse video tape recorder. Stimulus to the device under observation can be provided by a variety of signal sources via a 100 pin vacuum feedthrough and support circuitry. To ensure "clean signals" to the clocking inputs of the DUT, four differential line driver receiver pairs were incorporated. In addition, all wiring was done with shielded and terminated coax cables. The drive circuitry includes a four phase clock which accepts as input a master clock of frequency f and provides as output signals of f/2, f/4, f/8 and f/16. The inverted waveforms can be accessed as well. These outputs are then input to the line drivers where they are transmitted over 15 feet of cable to the receivers inside the SEM vacuum chamber. This circuitry is mounted behind a matrix switching array which allows the user to apply any one of ten different input signals to each of up to fifty IC pins. These fifty signals, including power and ground, are connected to the DUT via the cable and feedthrough. The matrix array can be seen in the center of the equipment rack, shown in Figure 6.2, to the right of the SEM mainframe. A memory exerciser (the Macrodata MD-100), power supplies, meter, scope, and function generator are also mounted in this rack to make the entire setup easily accessible to the operator.

The line driver signals are transferred into the vacuum feedthrough and received by support boards designed to incorporate the line receivers directly. These printed circuit boards are adaptable for observing voltage contrast on devices with up to 40 pins and allow direct application of line receiver signals to selected inputs of the IC. An example of a configured support board holding a typical device and residing on the SEM stage is shown in Figure 6.3. The support board limits movement of the stage within the chamber somewhat, notably the rotational movement.

Support services to this facility include equipment for specimen preparation such as a plasma etcher, an ultrasonic cleaner and a diamond saw for sectioning. Also available is a high power metallurgical microscope for immediate visual inspection.

6.2.2 Device Preparation

Because all devices used in this effort were obtained from commercial vendors, device preparation was an important step in the fault insertion process. Specimen preparation is also a key to good scanning electron microscopy. For IC's this preparation can include de-encapsulation (decap), cleaning and etching of the passivation layer.

The devices selected for this effort had ceramic packages. All but one device type that was chosen for fault insertion were sealed with a glass frit. The exception, the 6821, was solder-sealed. Prior to any preparation all devices were tested for logic integrity.

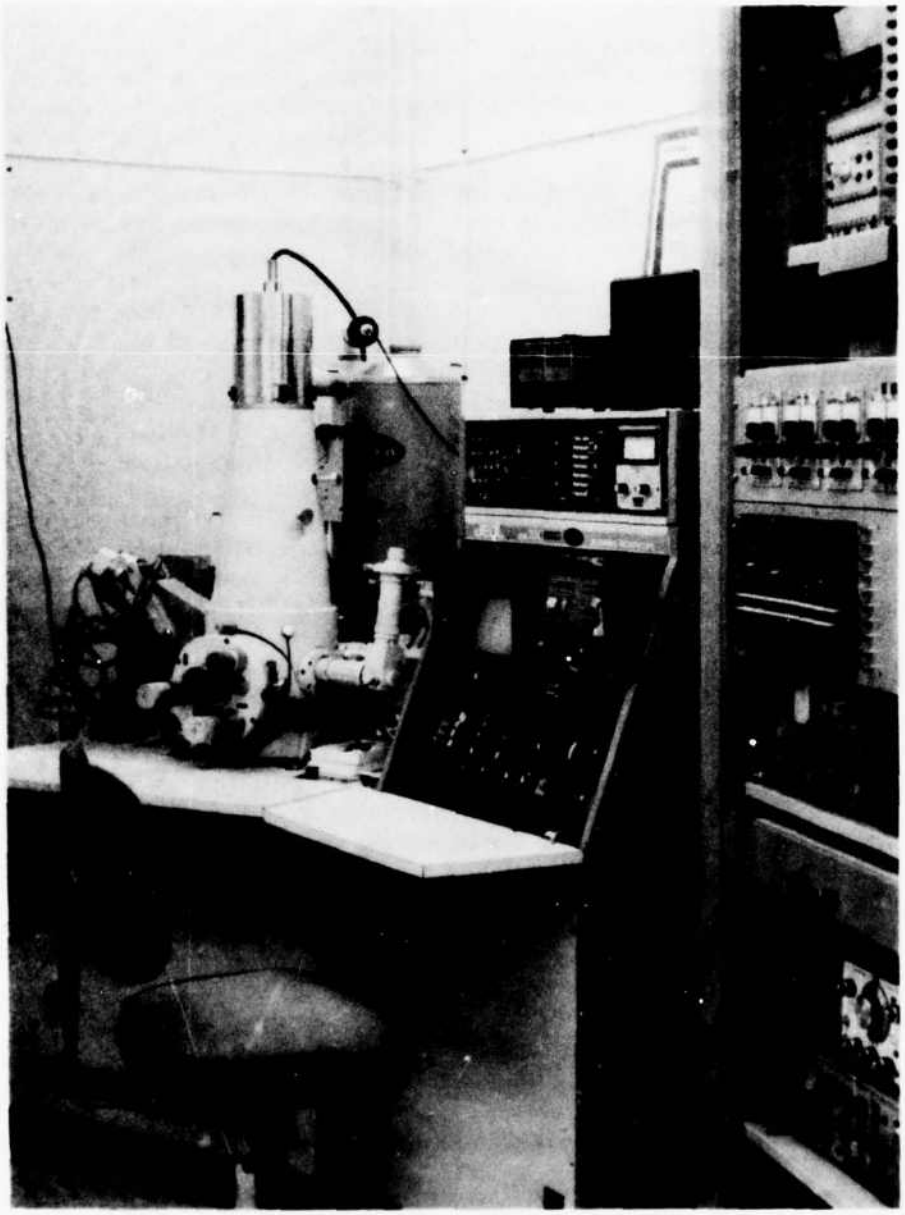


Figure 6.2. Scanning electron microscope and support equipment

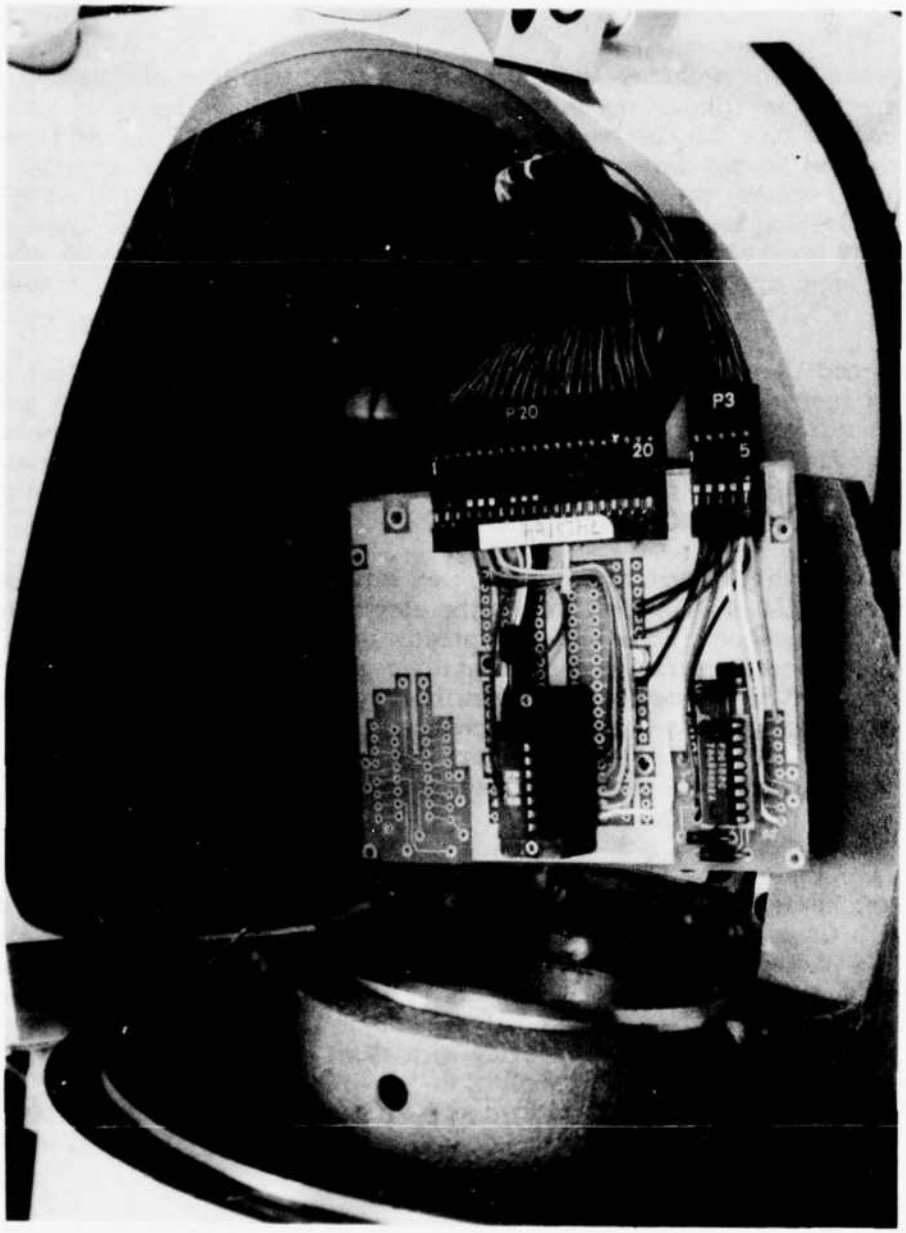


Figure 6.3. Support board

The standard technique for decapping ceramic-packaged, glass-sealed devices consisted of gripping the cap of the device in a vise and applying a sharp blow to the cap until the seal (and usually the cap) cracked. This method had about a 50% success rate with devices of 16 pins or less. A successful decapping is one in which the die is uncovered with no damage to it or to connecting pins or bonds. The 25LS2517, a twenty pin device, had only about a 20% success rate using this method because the cap on this device is thicker than the header and therefore the sharp blow tends to break the header before it breaks the cap.

To overcome this problem, and the even greater difficulties anticipated for the 40 pin devices yet to be tried, a decap method was devised which used a motorized circular diamond saw to slice the cap from the device. This method takes an average of four hours of sawing per 40 pin device but it has had a 100% success rate to date. A complication occurs because the saw blade runs through a lubricating oil bath to eliminate excessive loading on the internal gears. The oil must be cleaned from the device in a freon bath once decap is complete. At this point, the decapped device should again be tested for logic integrity to assess the effect of the decap process on device functionality.

The method used to decap the solder-sealed 6821 was to rapidly heat the device to the solder melting point and carefully lift the lid to expose the die. A small heat sink in this process to together all the pins of the static sensitive device and to heat the device evenly. Difficulty was encountered in lifting the loosened lid manually without damaging any of the wire bonds, which are directly below it. For this reason this method of decap had a 75% success rate.

The manufacturers' protective passivation layer deposited on the IC poses special difficulties for SEM examination. When the electron beam strikes an insulator, such as the silicon dioxide or silicon nitride typically used for passivation, electrons accumulate on the surface since no conductive path to ground exists. The effects of voltage potentials applied to the device are neutralized by the accumulating charge and the image of the IC appears evenly shaded (i.e., contrast is reduced). When a change in potential is applied to the device, the affected circuitry will demonstrate voltage contrast for a short period until the charging effect again neutralizes it.

Some experimentation was done with wet chemical etches to remove passivation layers but these were found to be difficult to regulate, due to variations in passivation layer thickness and composition, and therefore extremely destructive to the device. A plasma etcher using power levels of 15 to 20 watts and a 92% carbon tetraflouride and 8% oxygen plasma medium obtained much better results. However, etching rates were extremely variable because of the inconsistencies in passivation thickness. If etching is continued after the passivation layer is removed, the plasma will combine with the silicon wafer base of the device at a rapid rate and be destructive to the device. This presented problems in handling the devices used in this effort. The etching rates varied greatly from IC to IC as well as from device type to device type. In order to minimize the destruction of devices because of over-etching, this preparation process had to be discarded.

The only apparent alternative to etching was to to develop a technique of viewing necessary deviations under voltage contrast with the passivation layer in place. An advantage was discovered in leaving the passivation layer on the LSI devices. These devices are so complex that applying just power and ground to a sample creates a

complicated pattern of voltage contrast. This makes tracing individual runs and components difficult, as is demonstrated by the 2914 pictured in Figure 6.4a. The charging effects tend to "grey out" all static potentials applied to the device. When a change in voltage potential is then applied to the appropriate inputs, the affected circuitry shows clearly against the evenly shaded image of the device. Figure 6.4b shows this phenomenon on the 2914. The voltage contrast fades in a short while because of the charging but while it lasts it isolates the circuitry and makes it easier to trace.

It is important to note that because of the preparation needed for fault insertion, the devices are extremely susceptible to damage. Care must be taken in handling, transporting and storage of decapped devices to keep all bonds intact and the exposed die free of debris.

6.2.3 Fault Site Identification with Voltage Contrast

Once fault sites were selected with respect to the logic circuitry, and a familiarity with voltage contrast techniques was achieved, fault site identification on the physical device began. A necessary step in this process was a thorough analysis of the operation of the device. This analysis was geared toward developing vectors to apply to the device in order to exercise isolated portions of the circuitry. The fewer inputs needed to exercise the area of interest the easier it was to isolate the circuitry. Therefore, for maximum efficiency, it was necessary to develop patterns of voltage potentials to apply to the device that had the least number of varying inputs and still affected the selected site circuitry. At times it was also beneficial to develop several sets of patterns of voltages to apply to the device for identifying a single fault site. If several unrelated input changes should logically have an effect on the circuitry in question, each change should be applied separately to ensure the definitive location of the fault site. When possible, a known state was established using static input voltage levels. The state was used as a reference state from which circuit tracing could be performed.

The complexity of the device determines the number of vectors needed to initialize it and effectively exercise internal circuitry. When the logic is strictly combinational as in the 25LS2517, a single vector can be used to set up the device operation as needed. To trace circuitry in this device, manually switching between +5V and ground potentials on a single input can produce sufficient change in the operation of the device to allow definitive fault location. The inclusion of a clock in the logic necessitates a more complicated pattern progression. A method of simplifying this progression is to apply a constant pulse to the clock input at a rate faster than is visible on the SEM display (100 KHz and up). As the clock pulse is continuously applied, the rest of the device can be exercised in a manner similar to that used for combinational devices. The logic of some devices dictates that a series of patterns must be applied to a device before a specific site is affected. The MD-100 was used in such cases to automatically apply the needed patterns. The speed with which the patterns are cycled to the device by the Macrodata is controlled by an externally applied clock regulated by the user.

As an example of fault site identification, the procedure for identifying a fault site on the 2914 is given. The fault site chosen was one of the three outputs from the vector hold register. These three outputs combined are the encoded binary value of the highest unmasked interrupt that has just been received. The output of the vector hold register is used as a clear flag to the interrupts as they are serviced. The logic

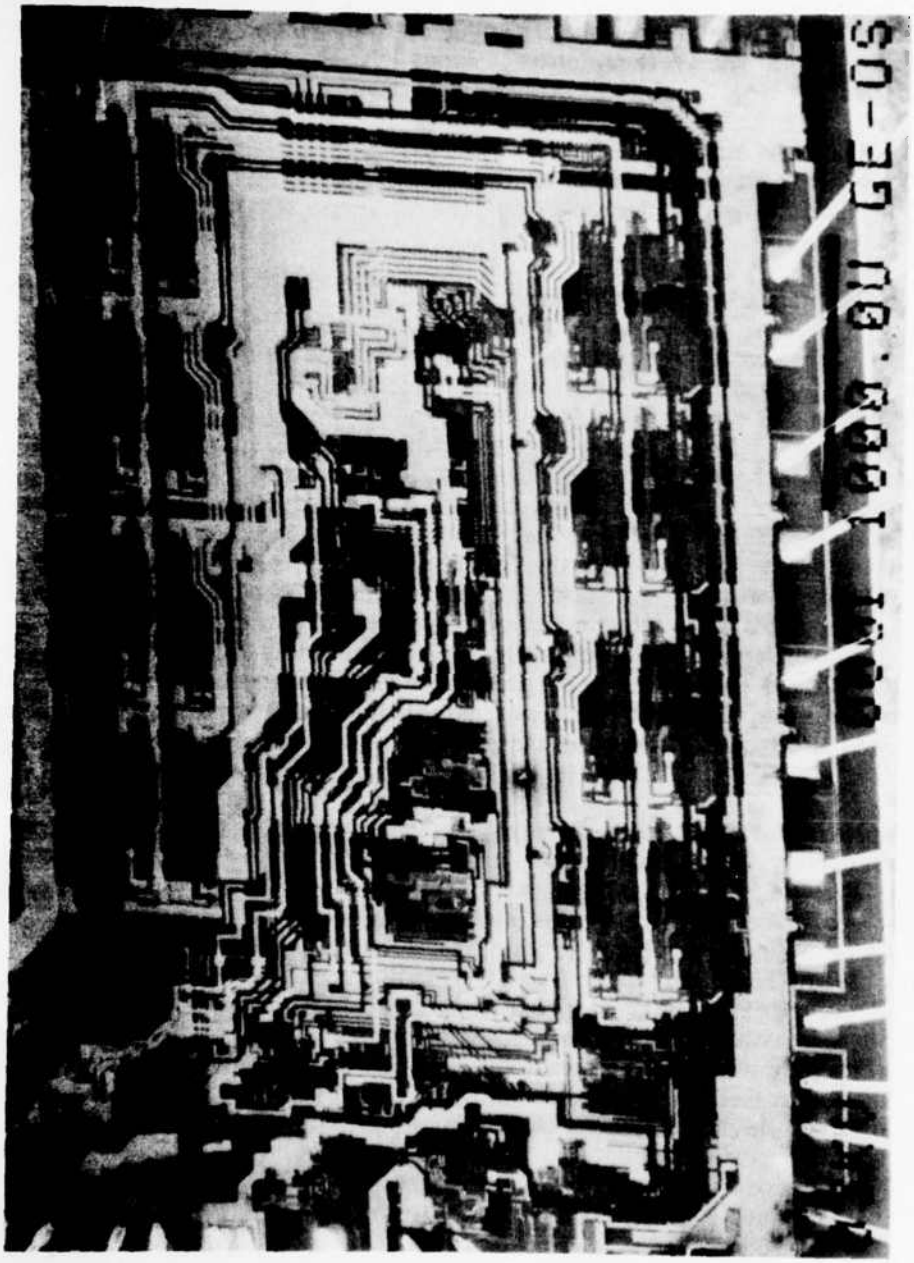


Figure 6.4a. Voltage contrast effects on the 2914

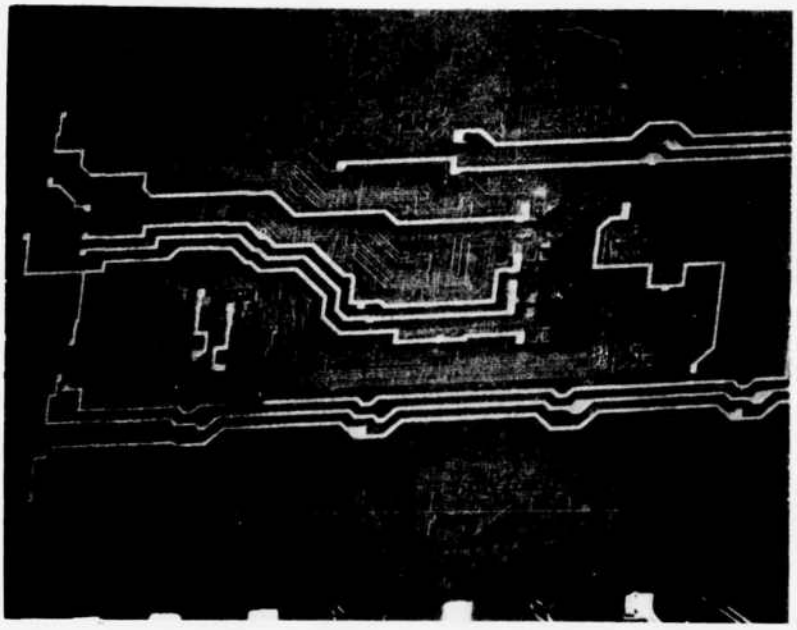


Figure 6.4b. Voltage contrast effects on the 2914, showing the inverted X2, X1, X0 signals, with P7 low (active) and instruction = "5"

location of the fault site is indicated on the portion of the Automaton Diagram in Figure 6.5 as Fault 3. The 2914 has an instruction set that provides proper control of the device. As can be seen from the Automaton Diagram, the instruction code must be set equal to "5" (read vector) for the encoded interrupt value to pass through the vector hold register. Also, the register is controlled by the clock input. Therefore a constant pulse of 100 KHz was applied to the clock input to enable operation of the device as if it were combinational logic and not visibly affect the SEM image.

Initially, the decapped device was checked for logic integrity and placed within the SEM. The patterns applied through the matrix array to the device consisted of +5V (high or "1") and ground (low or "0") potentials. Ground, power and clock were applied first. Then, to initialize the device, the Instruction Enable line was set low to make it active and the Instruction code was set to "0" to master clear the device. The basic pattern used for isolating the circuitry was:

-- P7-P0 (Interrupt Inputs) set high (inactive),

-- M7-M0 (Mask Inputs) set low (inactive),

-- Instruction Enable set low (active),

-- Instruction Code set to "5,"

-- Latch Bypass set high (active).

The outputs of the vector hold register are X2', X1', X0'. Enabling the input P2 will directly affect X1' only, so an alternating signal of ground and +5V was applied manually through the matrix array to this input. The effects can be seen in Figure 6.6. Figure 6.6a shows the circuit when P2 is high (inactive) and Figure 6.6b shows the circuit when P2 is low (active). From this series of events the X1 bit of the vector hold register was located. The same patterns were applied with the SEM focused at the register itself at a much greater magnification. The voltage contrast fades much quicker at greater magnifications because of an increased charging effect from the higher concentration of the electron beam on a smaller area. The voltage contrast faded too quickly to allow photographs to be taken but the tracing of the circuit visually was enhanced. It was discovered that both an X1' and inverted X1' outputs are generated from the register. A physical fault site internal to the register that affected both outputs in a consistent fashion had to be found because the register was modeled using only true outputs. Had only one of the complementary outputs been faulted, fault signature information would not have been accurate. Because of this complication and because of the complexity of the circuitry in this area, the only feasible place to physically insert a fault was the input transistor to the register. This changed the logical fault site from being an output of the vector hold register to being an input of the vector hold register. Figure 6.6c pictures the X1' register and pointers are placed at the X1' output, inverted X1' output, and the X1' input (the final identified fault location).

As noted in this example sometimes it was necessary to alter a selected fault location due to the manufacturer's implementation of the logic.

6.3 Physical Fault Insertion

The best method for accurately inserting faults into an LSI device is to use a

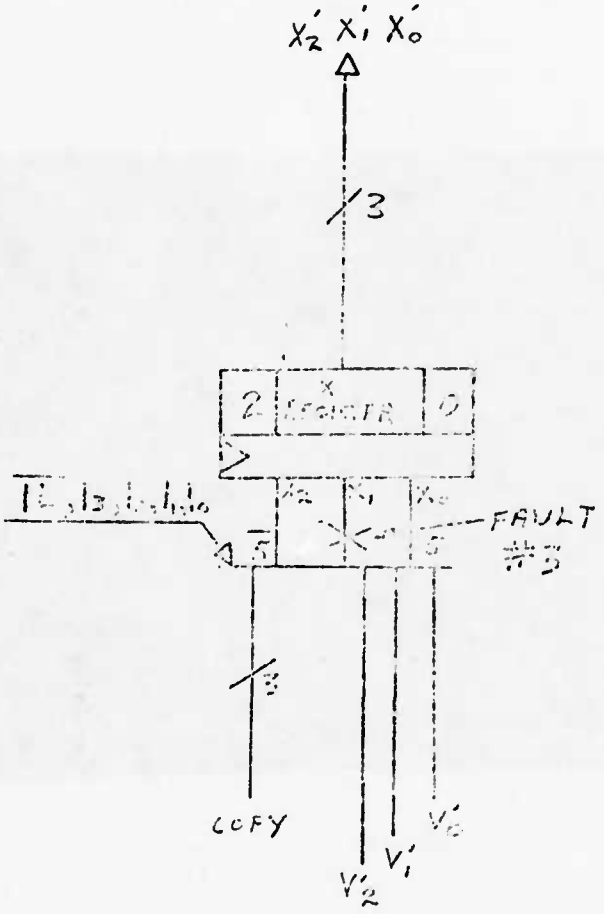


Figure 6.5. Partial Automaton Diagram showing the fault site

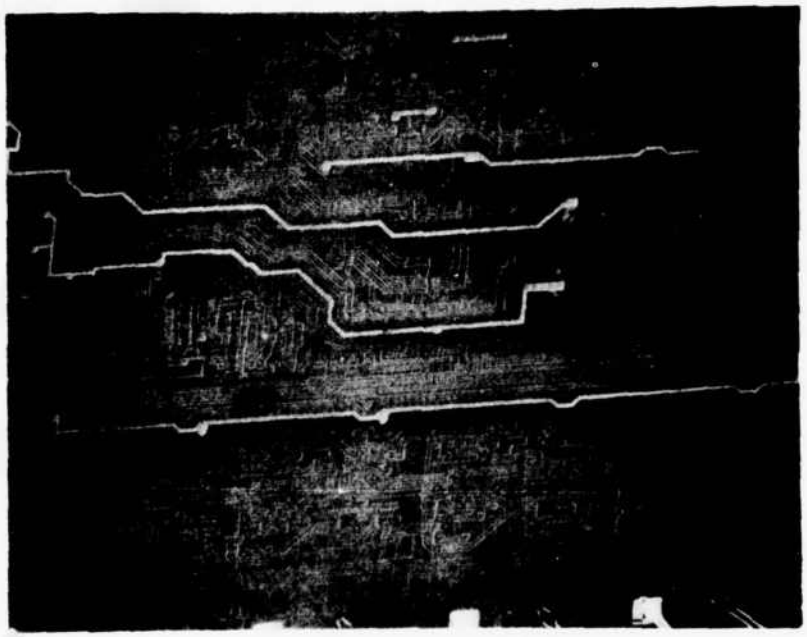


Figure 6.6a. Voltage contrast on the 2914, with P2 high (inactive)

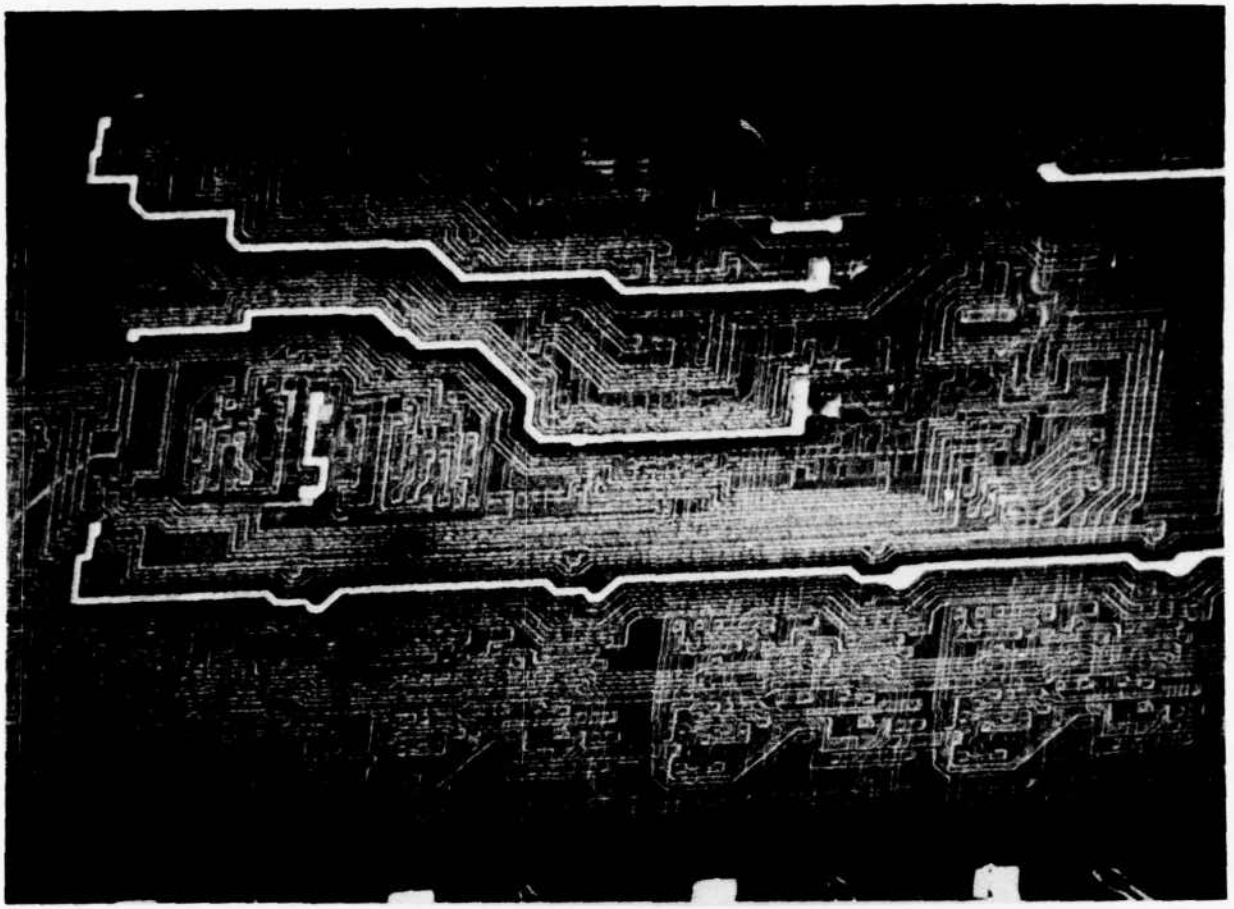


Figure 6.6b. Voltage contrast on the 2914, with P2 low (active)

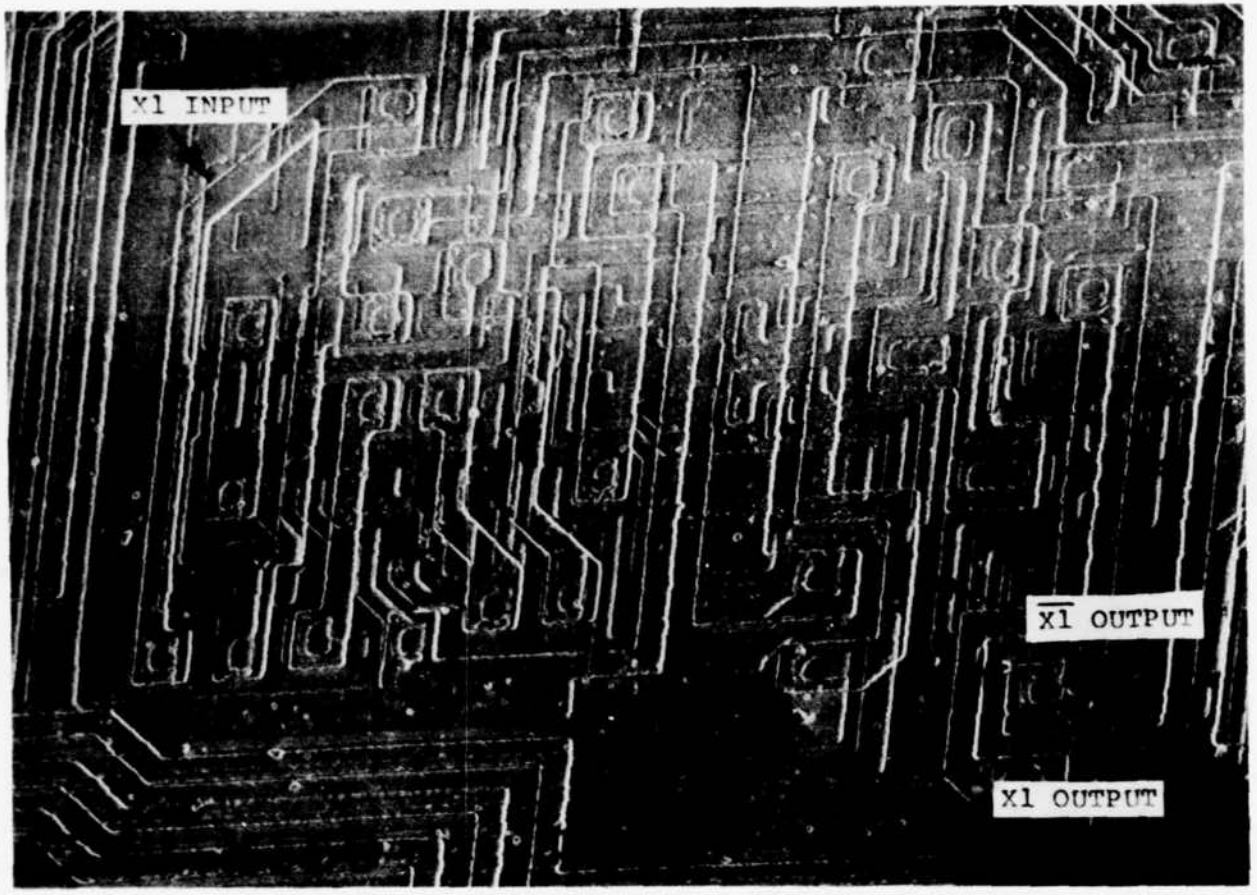


Figure 6.6c. 2914, showing the X1 input, inverted and non-inverted X1 outputs of the X-Register

laser to damage the specified area. Faults were inserted into the devices for this effort by creating discontinuities in runs identified as outputs from, or inputs to, a particular function block in the logic. These runs were typically 10 microns in width with 7 to 15 micron spacing between runs. The geometries involved restricted the type of laser necessary for accurate faulting. The most obvious requirement was the spot size of the laser. Assuming perfect centering of the laser beam on the run to be damaged, the minimum spot size of the laser could be no greater than the run width plus the perpendicular distances from either size of the run to adjacent runs or other circuit structures. This total width averaged about 25 microns (1 mil). Since some margin for error in positioning must be allowed, the laser had to have a minimal spot size considerably less than 25 microns. The positioning capabilities were also a consideration. It was necessary to have manually controlled, high resolution positioning in the X and Y directions in order to enable accurate orientation of the laser beam. Obviously, the resolution of the optical equipment through which the positioning was done needed to be as good as that of the laser beam and aligned with it in order to ensure visual identification of the fault site. Finally, the power of the laser needed to be sufficient to thoroughly cut through the prescribed runs.

The laser used in faulting the devices in this effort was a Union Carbide KORAD Resistor Trimmer. It is a YAG laser with specified minimum spot size of 1 mil. In actuality, the spot size can be narrowed approximately 10 microns. The stage which held the specimen could be positioned in 0.25 mil increments in both the X and Y direction once the course positioned was completed using a "joy stick" mechanism. The optics for positioning consisted of a microscope attached with magnification of approximately 60X. The power of the laser was sufficient for opening runs both on the surface of the devices and on subsurface levels.

The faulting procedure consisted of aligning the proposed fault location on a decapped, logic-tested device with the crosshairs of the laser microscope, applying laser pulses until a change was seen and observing the amount of damage in the affected area under a high magnification optical microscope. This process was repeated until it was determined that the run was damaged enough to result in a complete discontinuity.

Once the faults were inserted, the devices were again tested in the SEM to verify faulting effectiveness. The same patterns used to identify the site locations were applied to the devices to ensure the physical fault inserted created an open (effectively SA1 for most devices, if positive logic is used) condition at the appropriate logic component.

See the figures in section 7 for samples of the fault insertion results.

6.4 False Modeling

This section describes briefly a technique referred to here as "false modeling." This technique involves the deliberate introduction of a fault into a LASAR model of a DUT. This was done in a few cases during this effort in order to see how closely the results of this technique matched the fault signatures obtained from the insertion of actual faults.

It was to be expected, of course, that the fault signature from a LASAR model with an intentional fault in it would always match perfectly an entry in the

ZTABLE (see section 5.5), something that would be expected to happen only rarely in real life.

One use of the false modeling technique would be for circumventing the "FILE 10" option in the LASAR REDUCE package (the utility that produces the fault dictionary). As explained in section 5.5, this option limits the length of ZTABLE entries to 10, to reduce the size of the memory required for execution. This causes the fault isolation to suffer, as the Z-entries may not be long possible failed enough to discriminate among possible failed nodes in the model, since the fault signatures may not differ until later in the vector set. This false modeling would allow the user to create the complete fault signatures for each fault listed in the YTABLE, that resulted from the matching algorithm.

Another application would be to interactively home in on faults. The fault dictionary would identify a few regions that are candidates for failure, and the user could start inserting non-single-stuck-at faults, until he has matched the observed fault signature.

7. Applications

This section discusses details peculiar to each of the devices which were studied during this effort. These devices were:

1) 25LS2517 -- Arithmetic Logic Unit,

2) 2914 -- Priority Interrupt Encoder,

3) 15530 -- Manchester Encoder/Decoder,

4) 6821 -- Peripheral Interface Adaptor,

5) 2903 -- Bit Slice Operation Unit,

6) 2910 -- Microprogram Control Unit.

Each of these devices presented certain difficulties in modeling and testing. The 6821 handshake logic portion of the LASAR model has not been completely debugged at the time of this writing, but the response of the model differs from the response of the real device in only a few bit positions, so it is thought at this time that the vector set used to debug the model may have a subtle error in it, and may be exercising the device in an incorrect manner. The 2903 and 2910 LASAR models were not debugged, but the Automaton Models have been completed. The 25LS2517, 2914, and 15530 have had fault isolation performed on them, and the latter two devices had slash sheets prepared for them.

7.1 25LS2517

The 25LS2517 is a 4-bit ALU. This device is similar to the 54LS381, differing only in the functions of two outputs. The 25LS2517 provides outputs for use in ripple carry applications, while the 54LS381 provides the outputs used in carry look-ahead applications.

The devices used in this effort were the Advanced Micro Devices, Inc. Am25LS2517, which had a date code of 7834. The data sheets used were from AMD [1].

7.1.1 Learning the 25LS2517

Becoming familiar with the device, or "learning" the DUT, was relatively simple in this case. Being a strictly combinational device the test vector set for it was short. These test vectors were supplied entirely by LASAR, using the STIMGN facility, the only case where this was done during this effort. The fault dictionary was also prepared by LASAR.

7.1.2 Test Generation for the 25LS2517

Nothing of note here, as this was done automatically.

7.1.3 Test Implementation for the 25LS2517

The test was originally written as a TEKTEST program dedicated to this device. Later, when the SIMTEK program (see section 3.2.3) was developed, a new test using SIMPL was written. These tests and the faulted 25LS2517 were used to debug the S3260/3270 utilities discussed in sections 3.2.8 through 3.2.14.

7.1.4 Tracing and Inserting Faults in the 25LS2517

This device consists of strictly combinational logic in an open layout that allows most logic tracing to be done by visual inspection. Because of the simplicity of the device, the prime consideration in choosing fault locations was the possibility of putting multiple faults on a single physical device. All three faults were put on one device, each affecting an output or outputs exclusive of those affected by the other faults. Figure 7.1 gives the logic diagram for the 25LS2517 with the logic sites of the inserted faults identified. Figure 7.2 gives optical photographs of the actual faults inserted in the device. Fault 1 affects only output F0, Fault 2 affects only output F1, and Fault 3 affects outputs F2, F3, Cn+4, and OVR.

7.1.5 Isolating Faults in the 25LS2517

The faulted 25LS2517 was tested using four SIMPL files (see section 3.2.1), one for each fault plus one for an unfaulted device. The latter file verified that the test program, test system and ancillary equipment were operating correctly. Then the other three SIMPL files were used to log the errors from each fault separately. The fault isolation routines (see sections 3.2.10, 3.2.11, and 3.2.12) then evaluated each log file as though only one fault existed at a time.

The printouts from fault isolation routines pinpointed the faulted gate for each of the three faults. Thus the validity of the fault isolation algorithm described in section 5.5 was demonstrated.

7.2 2914

The 2914 is a Priority Interrupt Encoder intended for use in systems built with members of the 2900 bit-slice family. The 2914 has eight input lines for the signaling of interrupts, and has the capability of either sampling or latching the interrupts, masking any combination of interrupts, rejecting any interrupts below a certain level, and clearing interrupts in several modes. Several 2914s may be used in a parallel configuration to accept more than eight interrupts. This device has a total of 17 instructions, including a disabled state which could be called "Wait For Interrupt." The instruction bits are usually stored in the same microprogram store that serves the rest of the system. The approximate maximum useful clock rate (taking worst case conditions) is 9MHz, which rivals the test rate of most ATE.

The devices used in this effort for the characterization were the Advanced Micro Devices Am2914 which had date codes of 7727 and 7906. Those used for microcircuit tracing and fault insertion had the latter date code. The data sheets used were from AMD [2].

7.2.1 Learning the 2914

Bench testing of this device was done before looking at the logic diagrams. Working from only the prose descriptions and function tables, and what could be observed

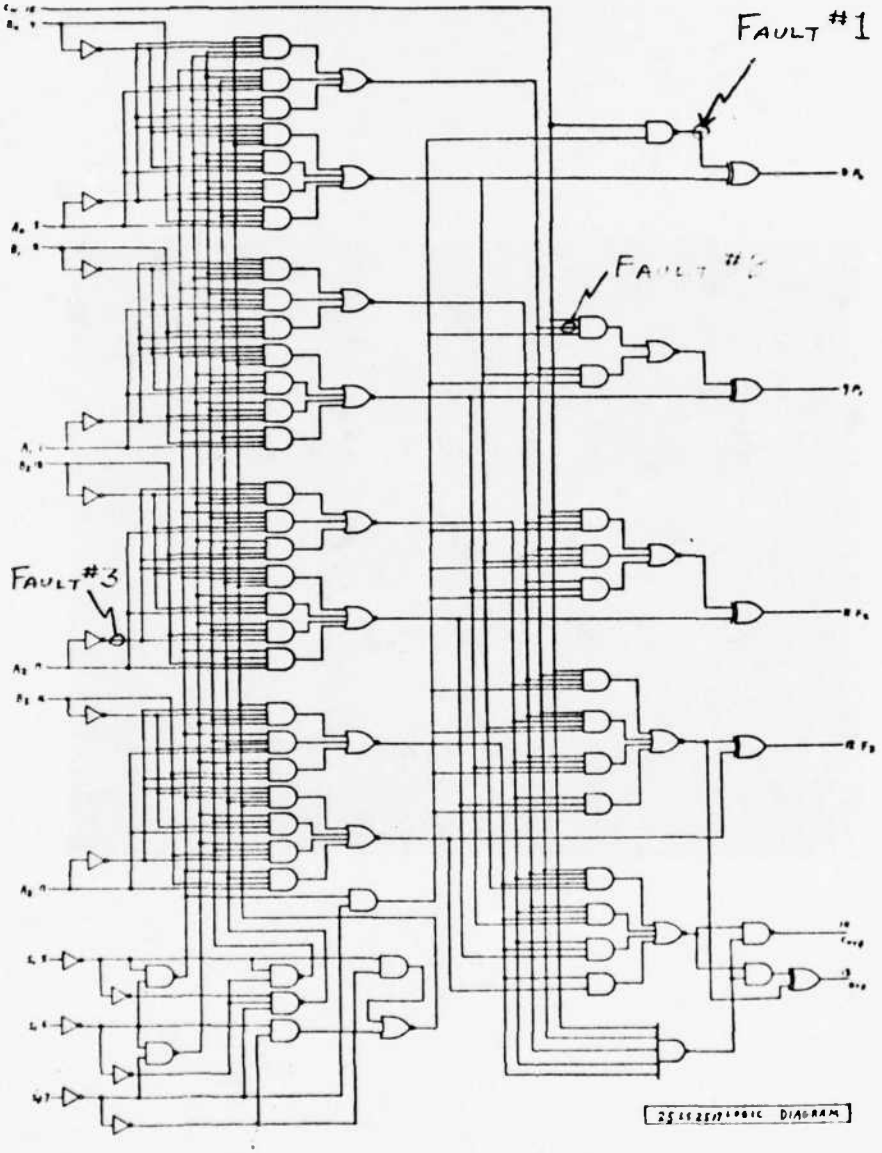


Figure 7.1. Logic diagram for 25LS2517, showing faulted locations

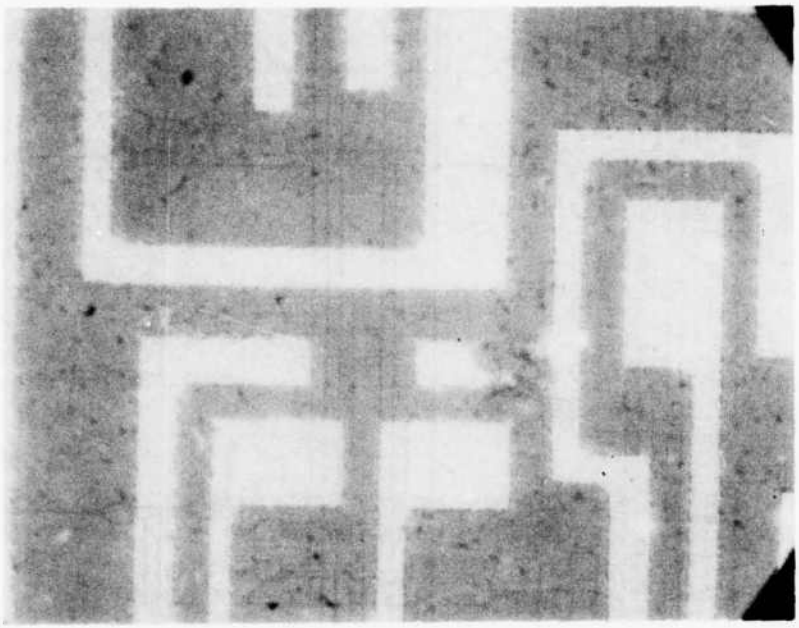


Figure 7.2a. Optical photograph of the 25LS2517, showing the location of Fault #1

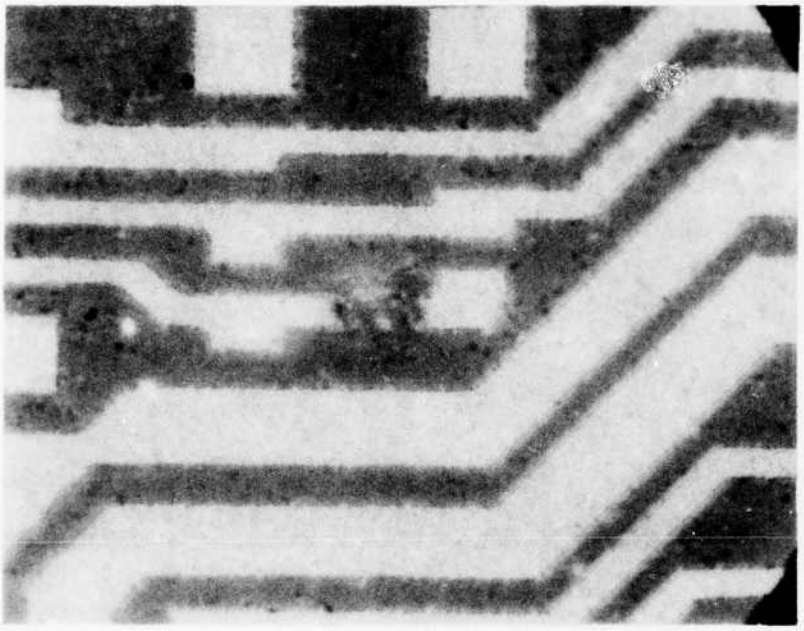


Figure 7.2b. 25LS2517, Fault #2

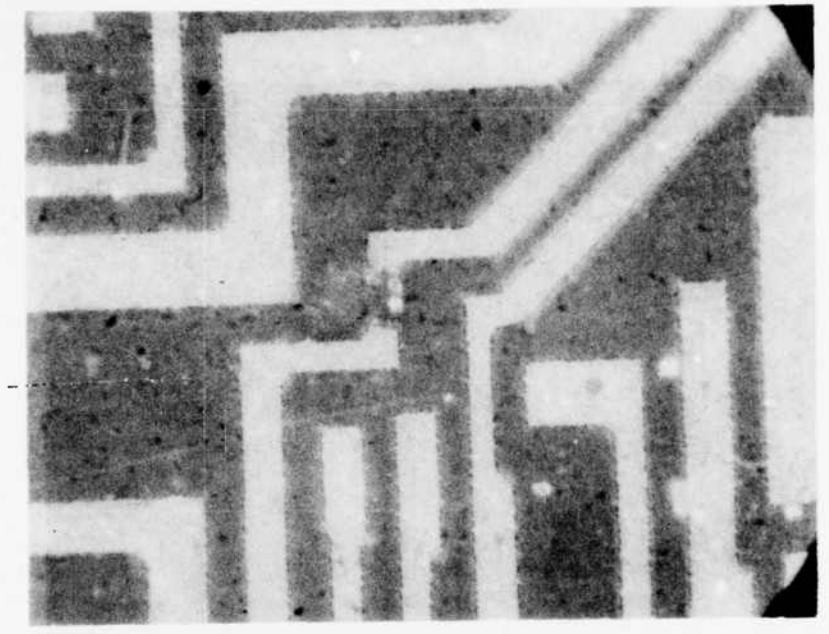


Figure 7.2c. 25LS2517, Fault #3

using a switch panel and lights, it was attempted to completely model the device. This failed, as the operation of the device was incompletely specified in the data sheets, and no information was available as to the operation during active portions of the clock cycle.

Although this lack was partially satisfied by the bench testing, the number of pins (effectively 29 inputs and 20 outputs, counting bidirectional pins twice) and the switching of input/output modes made this extremely difficult. Eventually, cheating (looking at the logic diagrams included with the data sheets) was necessary to complete the Automaton Diagram. It must be said, however, that this modeling was done very early in the effort (in fact, much was done in-house at RADC before the effort started) and that the sources of information have improved, along with the sophistication of the researchers.

The Automaton Diagram is shown in Figure 7.3.

7.2.2 Test Generation for the 2914

Test vector generation for the ATE was hampered by the same limitations as those that plagued bench testing, and ALICE was developed as a result.

The recipe approach to testing also started with the 2914. Although it is similar in effect to the testing of "hardcore" segments of a device [3] the philosophy is a bit different. Blocks are assumed to be completely accessible to the outside world, and tests are developed for them. The test writer then tries to apply those through the connecting logic, and sensitize the result to the output (see sections 1 and 4.2).

Most of the LIT was generated by manual techniques (through ALICE), but the last few vectors were generated by the LASAR STIMGN, to see if LASAR could take care of the small number of remaining faults that were extremely difficult to detect manually. The final vector set, it is believed, detects every single-stuck-at fault that can possibly be caught. The remaining faults are in logic that, because of the design, cannot be tested.

Unique problems arose during the generation of the AC parametric testing. Due to the small propagation delay times, and minute setup and hold times, the S-3260's capability was taxed. Small differences in table skew, which cumulatively were nearly the same as some of the times being measured, accounted for a great deal of perturbation in the results when debugging these tests.

The DC parametric tests were relatively simple to complete, as any output can be put into any desired state with a very short preconditioning vector set. In addition, GO/NO-GO tests for VOH and VOL were done entirely by running the LIT with the output pins loaded.

With this device there were no tests included specifically to verify the fact that the bidirectional pins and three-state pins actually go into a high impedance state within some specified propagation delay time, during the AC parametric testing. Although this is a very important parameter, it is extremely difficult to test without complicated switching of loads, and multiple passes during the testing.

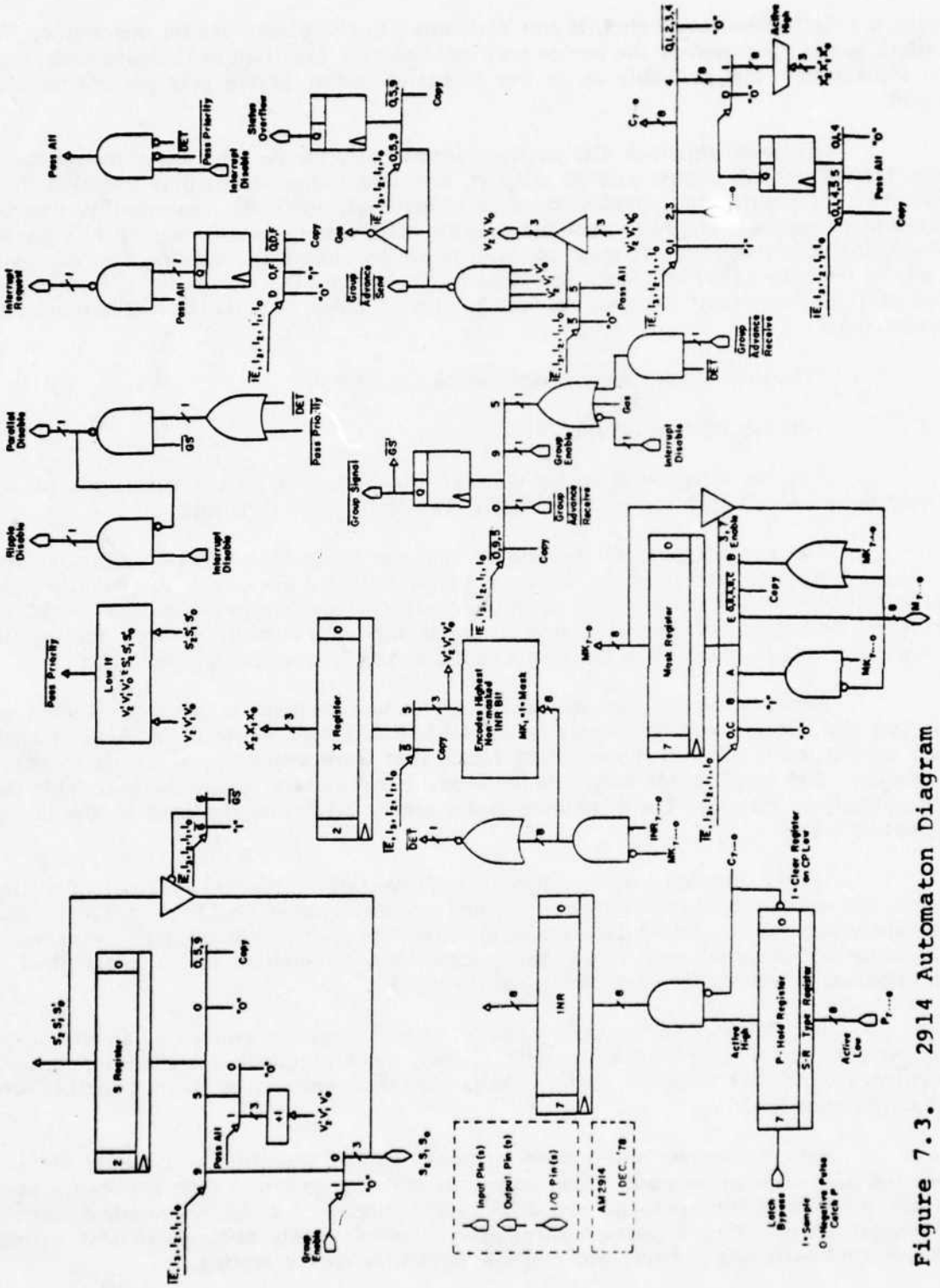


Figure 7.3. 2914 Automaton Diagram

A complete draft slash sheet was prepared for the 2914 (see sections 2.1 and 8.1).

7.2.3 Test Implementation for the 2914

Because of the large number of AC parameters, this test was originally written with the setup, limits and pattern file row numbers stored in an array in the TEKTEST program. The program then replaced variables in hardware control statements with numbers from the array and executed the statements in a large loop. This approach led directly to the SIMTEK program with the array stored separately as the SIMPL file (see sections 3.2.3 and 3.2.1).

The AC parametric test problems mentioned above were partially circumvented by delaying the application of the signals under test until the ATE cycle was well along, with the other inputs applied at the start of the cycle. This allows the programmed signal application times to be adjusted (both positive and negative) relative to the reference signal (usually clock) to correct for measured errors in the ATE. The obvious problem with this approach is the large number of correction factors that are needed since each pin pair under test would have a unique error. The next step would be to automate the measurement and use of the correction factors. This approach was not developed further because of the lack of time. Tektronix, Inc. has partially implemented the above concept for use with their High Performance Option (HPO) and their T.I.M.E. Option.

7.2.4 Tracing and Inserting Faults in the 2914

The 2914 is fabricated using multilayer metallization. Circuit tracing was attempted using enlarged micrographs and visually inspecting the physical device but the device proved to be too complex. Voltage contrast techniques were initiated with this device, as a method of fault site identification. The location of Fault 1, an output to a gate with eight inputs, was selected because each input could be readily exercised and viewed on the SEM using the voltage contrast mode. In exercising this gate, it was discovered that it had two outputs, a DET signal and its inverse. This required finding a fault location internal to the gate in order for these outputs to be faulted in a consistent manner (stuck at opposite levels). Fault 2 was chosen because of its effect on a single output. Fault 3 was chosen because of its effect on the Vector Hold circuitry. The fault site was initially chosen as the output of the X Register but the logic was again implemented in such a way that each X and its inverse were output separately from the X Register. The circuitry of the register itself was too dense to allow laser faulting so the fault was inserted at the input to the X Register, bit X1.

Figure 7.4 shows portions of Figure 7.3, the Automaton Diagram of the 2914, with additional markings indicating the fault sites used. Figure 7.5 shows optical photographs of faults that have been inserted on physical devices (one fault per device).

7.2.5 Isolating Faults in the 2914

The faulted 2914s were tested on two S-3260/3270s with somewhat less than total success. Fault 1 was located satisfactorily by the fault isolation routines, but Faults 2 and 3 were not located very well.

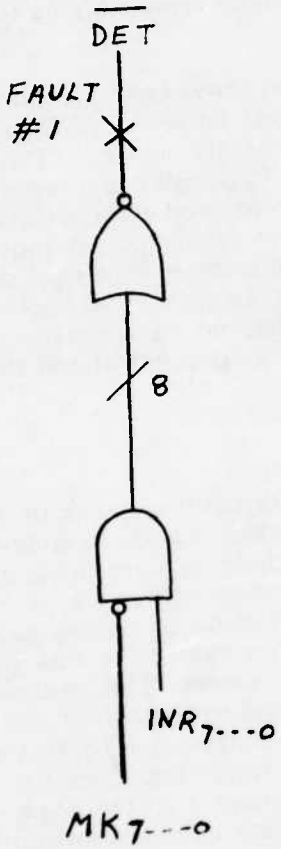


Figure 7.4a

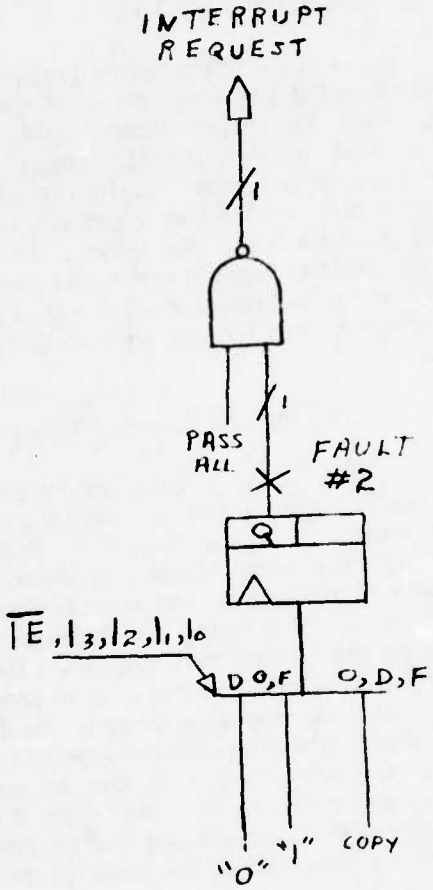


Figure 7.4b

Figure 7.4. Portions of the 2914 Automaton Diagram, showing the fault locations

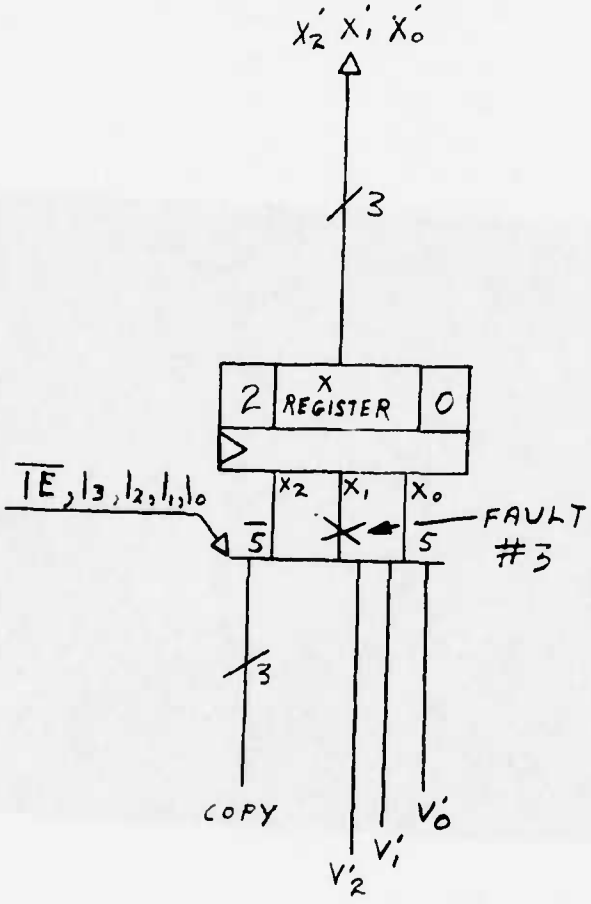

X2' X1' X0'
3
2
X
REGISTER
0
IE, I3, I2, I1, I0
X2
X1
X0
5
5
FAULT
#3
3
COPY
V2'
V1'
V0'

Figure 7.4c

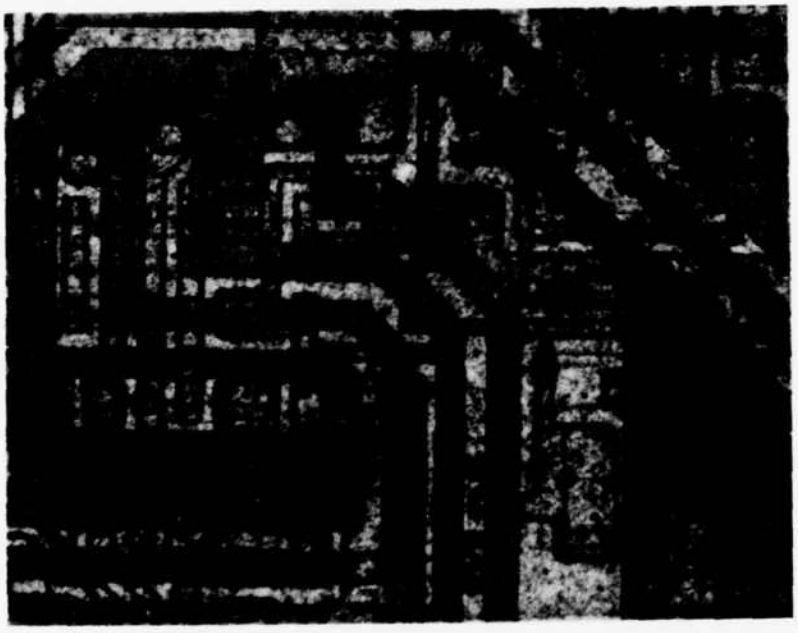


Figure 7.5a. Optical photograph of the 2914, showing the location of Fault #1

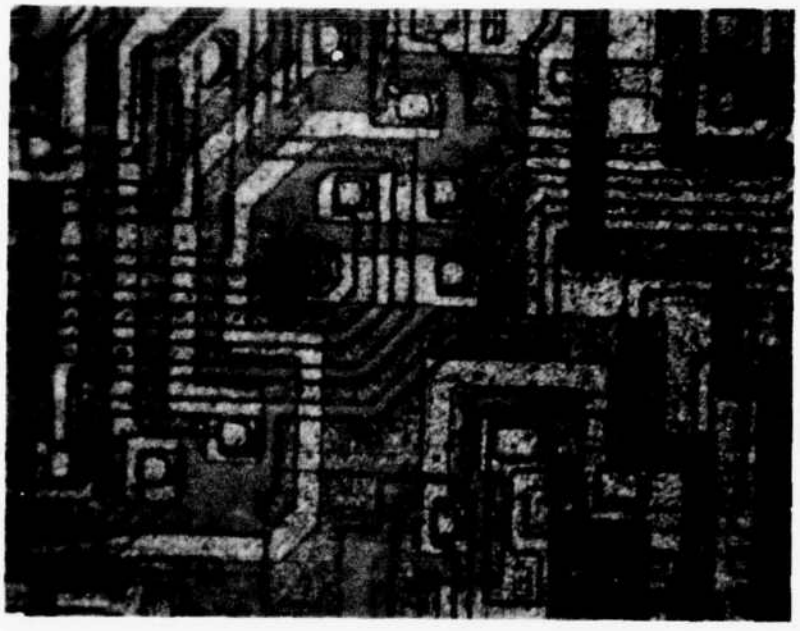


Figure 7.5b. 2914, Fault #2

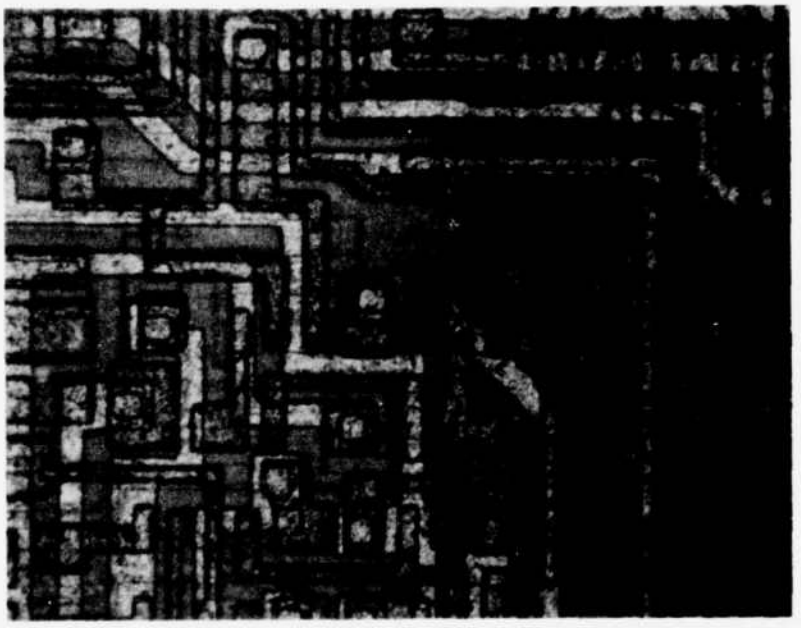


Figure 7.5c. 2914, Fault #3

The printout for Fault 2 included the actual fault site as the second best option, and the printout for Fault 3 included the actual fault site stuck-at-zero in place of the expected stuck-at-one. Both of these discrepancies may have been caused by having the actual implementation at the chosen fault sites being inverted from the logic level indicated on the available schematics. This emphasizes one of the hazards of developing device models without complete and detailed knowledge of the actual device implementation.

The concept of false modeling was verified using the 2914. The approach was to replace a connection in the model with a stuck-at-one or stuck-at-zero, and generate new expected outputs with LASAR, using the same inputs that were used for the good model. The differences between the new outputs and the good outputs were input to the FISO routines, which then isolated the fault. Using this concept verifies the models and dictionaries without tracing a device and inserting a fault.

7.3 15530

The 15530 Manchester Encoder/Decoder is intended for use in Avionics Systems. This device, through line drivers and receivers, provides the means of encoding and decoding NRZ data using the Manchester II Bi-Phase Level. This directly supports the protocol outlined in MIL-STD-1553 [4].

The devices used in this effort for the tracing and fault insertion were the Harris Semiconductor HD-15530-2, with datecodes of 7815 and 7840. The data sheet was from Harris Corporation [5].

7.3.1 Learning the 15530

The 15530 has 12 inputs and 10 outputs, with no bidirectional or three-state pins. However, due to the great number of clock cycles needed to cause changes in the outputs, it is a very difficult device to characterize entirely on the bench. In fact, this was one of the few cases where a static device was easier studied at a high clock rate than when single-stepping. In addition, the 15530 contains two completely separate functions, which share nothing but a common master reset (encoder and decoder functions).

In spite of the name "master reset" this signal does very little to the internal state. There is an additional "decoder reset" for which the same is noted.

Harris Corporation supplied the schematics to RADC for the purpose of evaluating the draft slash sheet submitted by Harris. This schematic is considered proprietary by Harris, and so no notes or models for the device are included in this report. The machine-readable LASAR model for the 15530 and fault dictionary for the device have been treated in the same way.

In any case, an Automaton Diagram for this device could not be developed that was any simpler than the gate and flip-flop level, due to the single-bit-wide data path and a reliance on random logic for state decoding.

7.3.2 Test Generation for the 15530

The test generation for this device relied less on the recipe approach than did that for any other device tested to date. As the largest functional element was a flip-flop, and there were none of the usual functional blocks, test generation was done on an ad hoc basis. The tests consisted of a set of valid and invalid words, graded by the LASAR DYSOGN. These "words" are 16-bit data words, as defined in MIL-STD-1553. The Encoder was tested by using NRZ data, and the Decoder was fed Manchester II Bi-Phase. The feedback from LASAR in the form of undetected nodes provided guidance during the writing of the test. Unfortunately, there were a great number of nodes and failures which are completely untestable.

The fault isolation provided by the fault dictionary is not very good. There are very few separate paths for the data, causing most failures to be vaguely isolated to a region somewhere in a counter chain.

The AC tests developed for the 15530 were very complicated. There are several parameters which are simple propagation delay times, but many are delays with respect to another output, rather than an input. This required the use of the Delta-T subsystem on the S-3260/3270 (see section 2.4).

In addition, the measurement of the Divide-By-Six output at its maximum speed required this device to be tested using the Mode 1 of the S-3260/3270, instead of the usual Mode 3.

7.3.3 Test Implementation for the 15530

The test for the 15530 was implemented on the S-3260/3270 using the SIMTEK program. This test was used to verify the output from LASAR and thus the model. Also, the time measurement and unique portions of SIMTEK were evaluated.

This device was tested on all four of the S-3260/3270 ATE's mentioned in section 4.3.4.3. The only problem encountered in changing testers was caused by one tester not having a full set of sector cards.

The LIT pattern for this device includes several hundred vectors needed to drive the internal states to a known condition, and to cause all of the outputs to become known. This technique of initialization is compatible with most ATE's. Using special features, such as the ability of the Sentry to execute a group of vectors repeatedly until the desired condition is obtained, is a technique that would make the pattern incompatible with another ATE family.

7.3.4 Tracing and Inserting Faults in the 15530

Much of the fault identification/insertion work for this device was completed using the RADC Product Evaluation Report. Computer-Aided Design techniques were used in developing this device. This method of design uses standard cell designs, making the logic gates and other basic structures easily identifiable. Because CMOS technology was used less laser power was required to deliberately damage the polysilicon runs. The regularity of the design also aided in locating fault sites and in inserting the faults.

The logic circuit diagrams and schematics used in identifying suitable fault sites are found in the RADC Product Evaluation Report of the Harris 15530, June 1979. This was on restricted distribution and cannot be reproduced in this report. Basically,

the faults were placed in the following locations: the output of the ENCODER ENABLE flip-flops in the Encoder circuitry; the output of the Parity check flip-flop in the Decoder circuitry; and the output of the flip-flop of a single register bit in the Bit Counter in the Decoder circuitry.

7.3.5 Isolating Faults in the 15530

The fault isolation printouts for the 15530 included each of the inserted faults. But each printout also listed a large number of other possible fault sites. This was caused by the highly sequential implementation and the extensive use of feedback loops in the logic. This illustrates the difficulty in isolating faults in counters and feedback loops.

7.4 6821

The 6821 is a Peripheral Interface Adapter (PIA). This device attaches to the data bus of a microprocessor system (usually one built around the 6800) and in its usual configuration is addressed as part of RAM. There are two ports, A and B, each of which have eight lines which go to the outside world. Each bit of the A and B ports may be programmed individually and independently to be an input or an output. There is handshaking logic supplied for both the A and B side that can act in several different modes. The 6821 can cause an interrupt in the microprocessor system when an external device requests service or responds on the handshaking lines.

The devices used for tracing were the Motorola Semiconductors, Inc. MC6821 which had a datecode of 7934. The data sheets were from Motorola Inc. [6, 7].

7.4.1 Learning the 6821

The 6821 consists of two major sections: the registers and buffers, which were simple to understand; the handshake logic, which is still not completely understood or debugged in the LASAR model.

Part of the learning process was done on the bench with a set of switches and lights, and part was done on the S-3260. Although the data sheet provides an excellent description for the 6800 system programmer, it fell short of that needed by a test writer. A Product Evaluation report was available, but the handshake logic had proved to be too difficult to trace meaningfully. However, valuable information about the control logic was obtained from the report.

The Automaton Diagram for the 6821 is shown in Figure 7.6. Bear in mind that the sections covering the handshake logic are only postulations, and have been fudged over and over to try to copy the behavior of the real device. This section is made up of asynchronous logic and the behavior under a few odd conditions is very confusing.

7.4.2 Testing the 6821

The tests generated for the 6821 are limited because they were originally intended to help debug the model, not provide for fault detection or isolation. That would have come later. The portions of the planned test which would cover the registers and control logic would be a straightforward application of the recipe approach. The

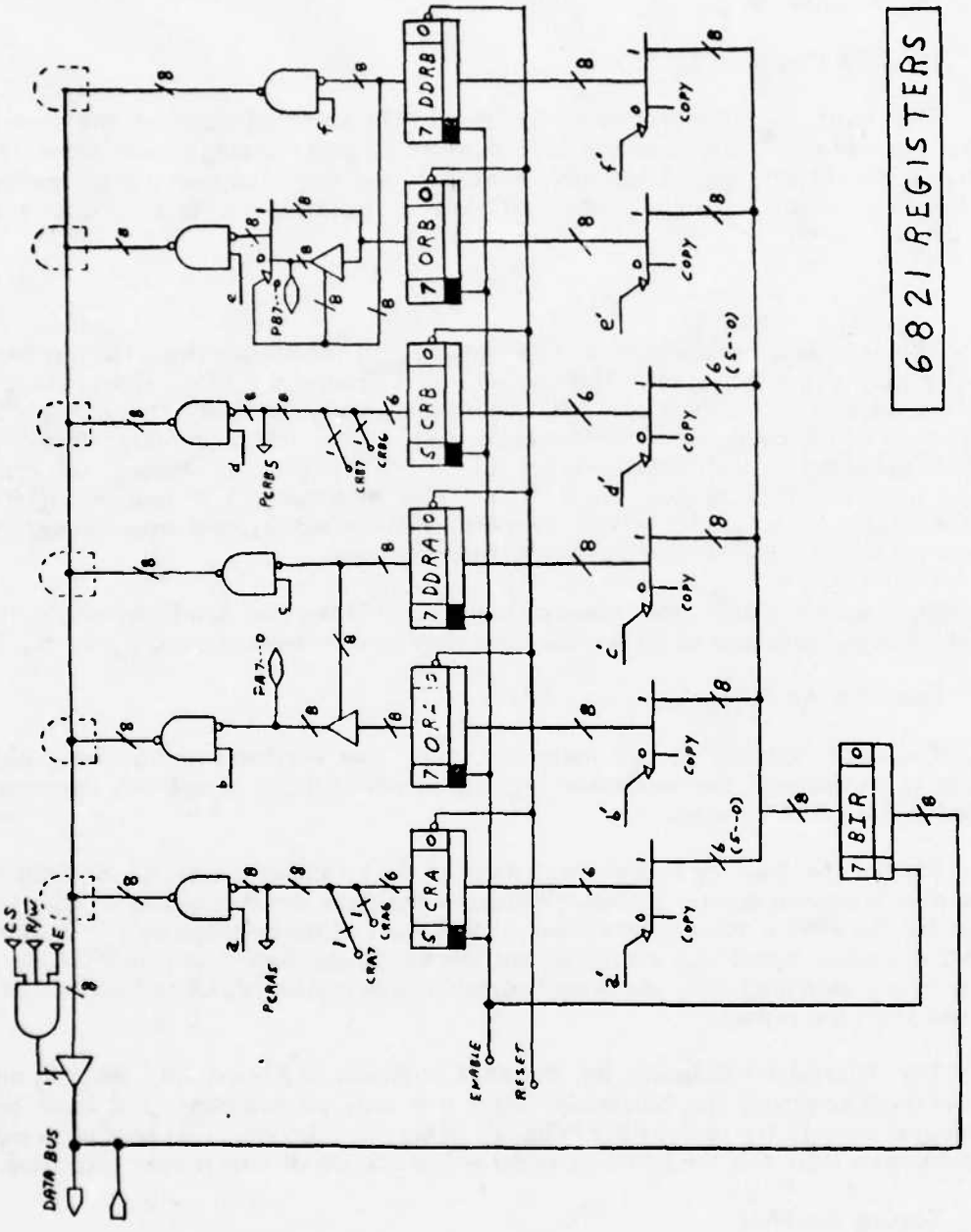


Figure 7.6. 6821 Automaton Diagram

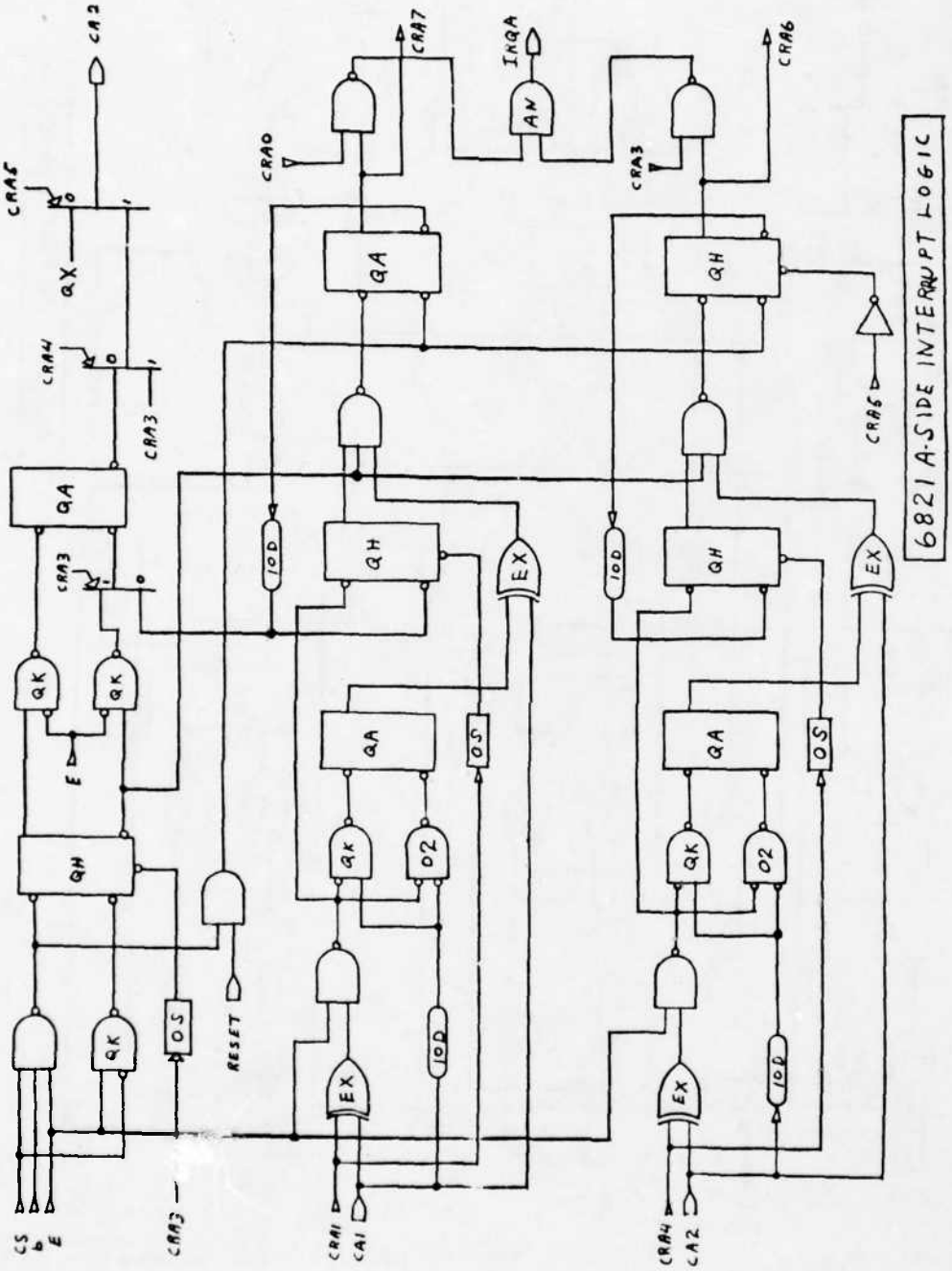


Figure 7.6. 6821 Automaton Diagram, continued

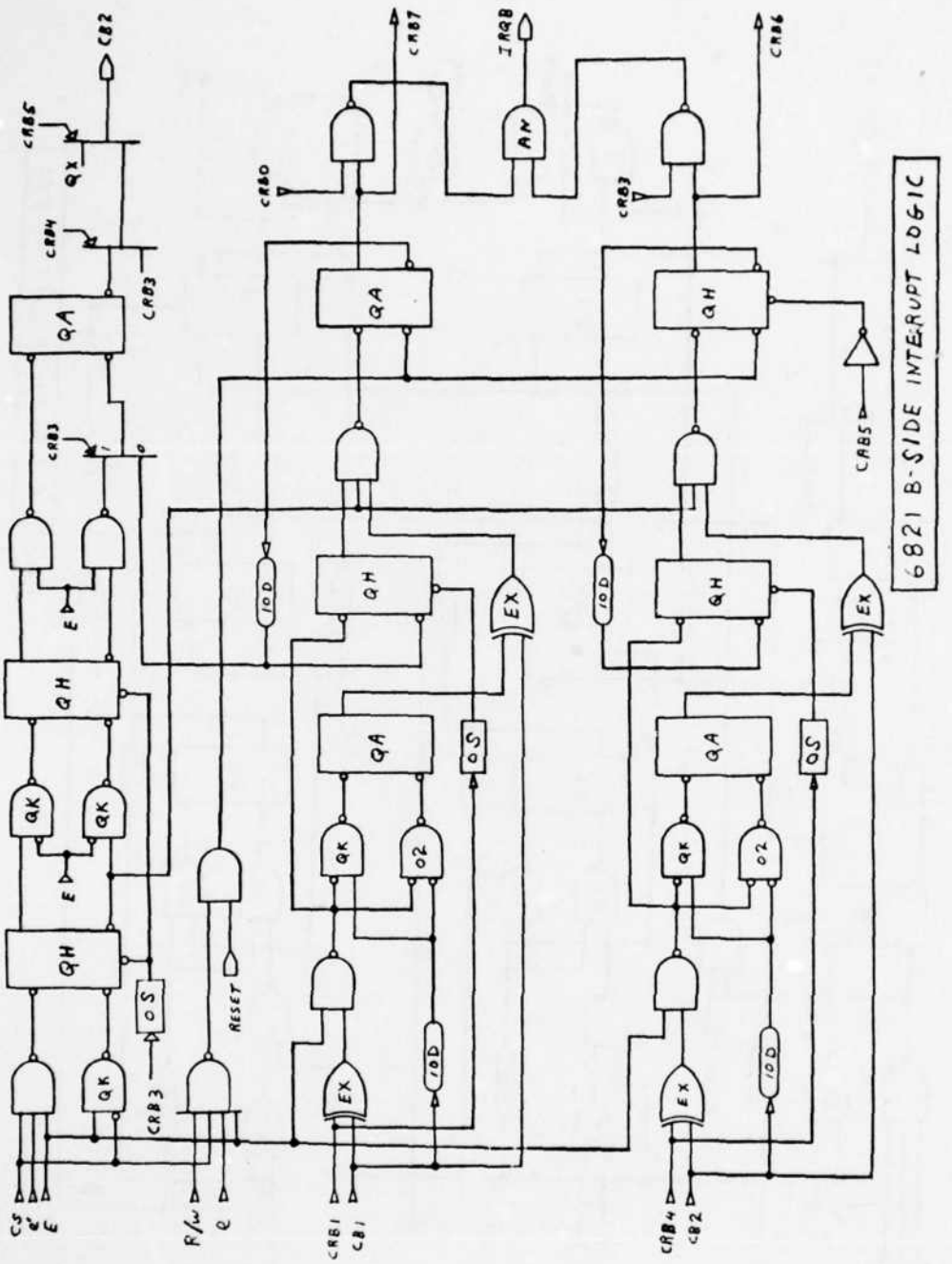


Figure 7.6. 6821 Automaton Diagram, continued

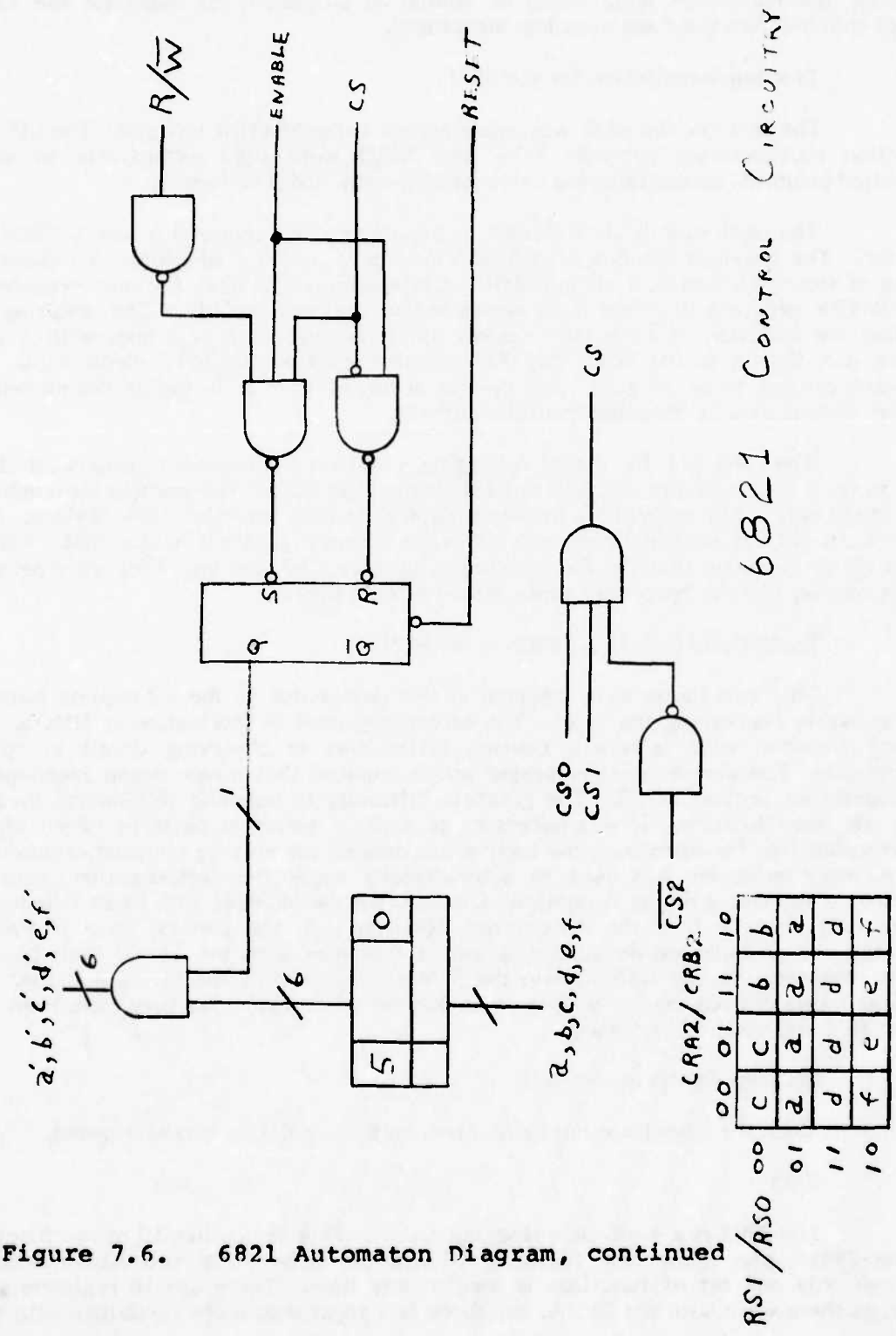


Figure 7.6. 6821 Automaton Diagram, continued

tests for the handshake logic would be similar in philosophy to those for the 15530, except that the functions are even less structured.

7.4.3 Test Implementation for the 6821

The test for the 6821 was implemented using SIMTEK routines. The LIT data reduction routines (see sections 3.2.6 and 3.2.7) were used extensively to obtain simplified printouts of the failures, thus assisting with model debugging.

The 6821 was the first device in this effort that required a new S-3260/3270 adaptor. The previous devices used modifications of existing adaptors. To check the wiring of the new adaptor, a simple SIMPL file was written. This file was executed by the SIMTEK program that had been generated to perform the LIT. The resulting test verified the isolation of each DUT socket pin from all other pins and, with a little manual aid, the continuity from the DUT pin to the associated ATE electronics. This approach proved to be so easy, useful, and accurate that it is highly recommended. Further details may be obtained from the authors.

When the LIT for model debugging was first implemented, nearly all of the pattern rows failed on nine of the ten 6821 devices on hand. The one (serial number 7) that failed only a few rows had a different date code than the other nine devices. Also, the devices did not exhibit the expected voltage contrast patterns in the SEM. Further investigation revealed that the 6821 packages contained 6800 chips. They were returned to the vendor, who promptly sent replacement 6821 devices.

7.4.4 Tracing and Inserting Faults in the 6821

Only two faults were inserted in this device due to the incomplete information available concerning the logic. The technology used in fabrication is NMOS. The spacing between runs is small, causing difficulties in observing detail in optical photographs. The device is solder sealed which required that a new decap technique be developed (see section 6.2.2). The greatest difficulty in handling this device involved fault site identification. It was necessary to apply a series of patterns to set up the proper conditions for exercising the logic areas desired for voltage contrast observation. The memory exerciser was used to automatically apply this initialization sequence. Figure 7.7a is sheet 2 of the Automaton Diagram for the MC6821 with Fault 1 indicated. Figure 7.7b is sheet 1 of the Automaton Diagram with the general area of Fault 2 indicated and an enlarged detailed diagram of this area with the actual fault location shown. Figures 7.8a and 7.8b contain the SEM photographs of the inserted faults. The physical faults are not clearly visible on an optical microscope, yet they have been seen on the SEM and verified by testing.

7.4.5 Isolating Faults in the 6821

Since the model was not completed, no fault isolation was attempted.

7.5 2903

The 2903 is a 4-bit-slice Operation Unit. This device has all of the functions of the 2901A plus many new features. There are effectively two ALU's onboard, although only one set of functions is used at any time. There are 16 registers in an array, as there were with the 2901A, but there is a great deal more flexibility with their

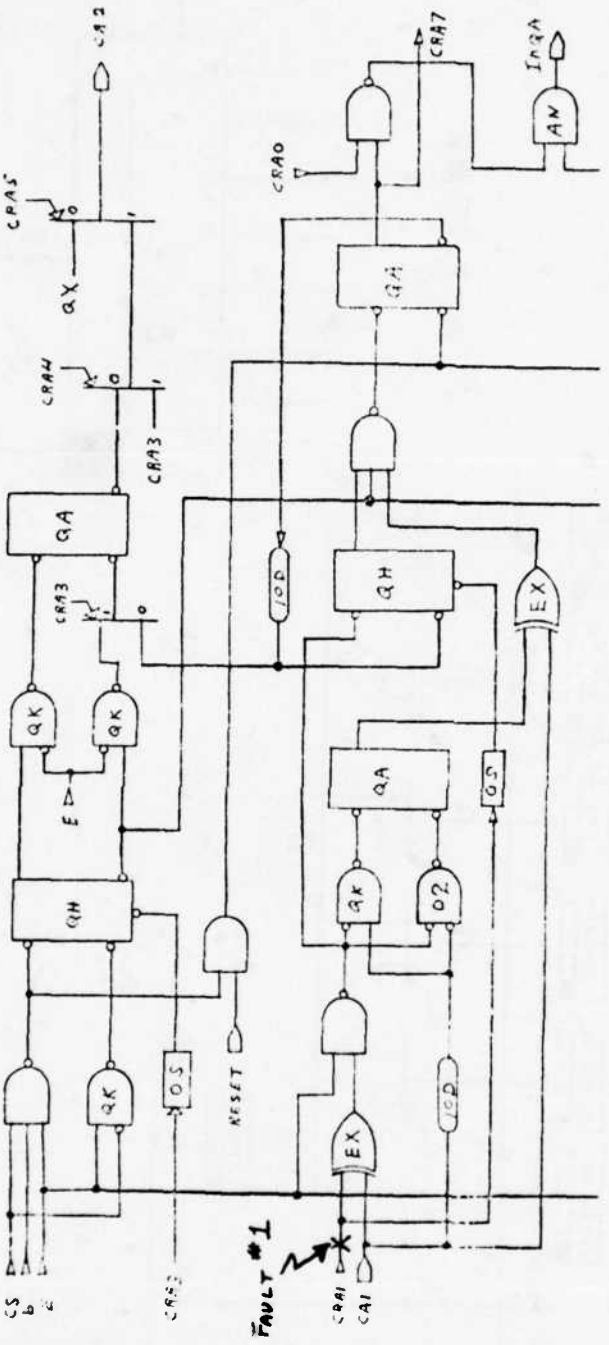


Figure 7.7a. A portion of the 6821 Automaton Diagram, showing the site of Fault #1

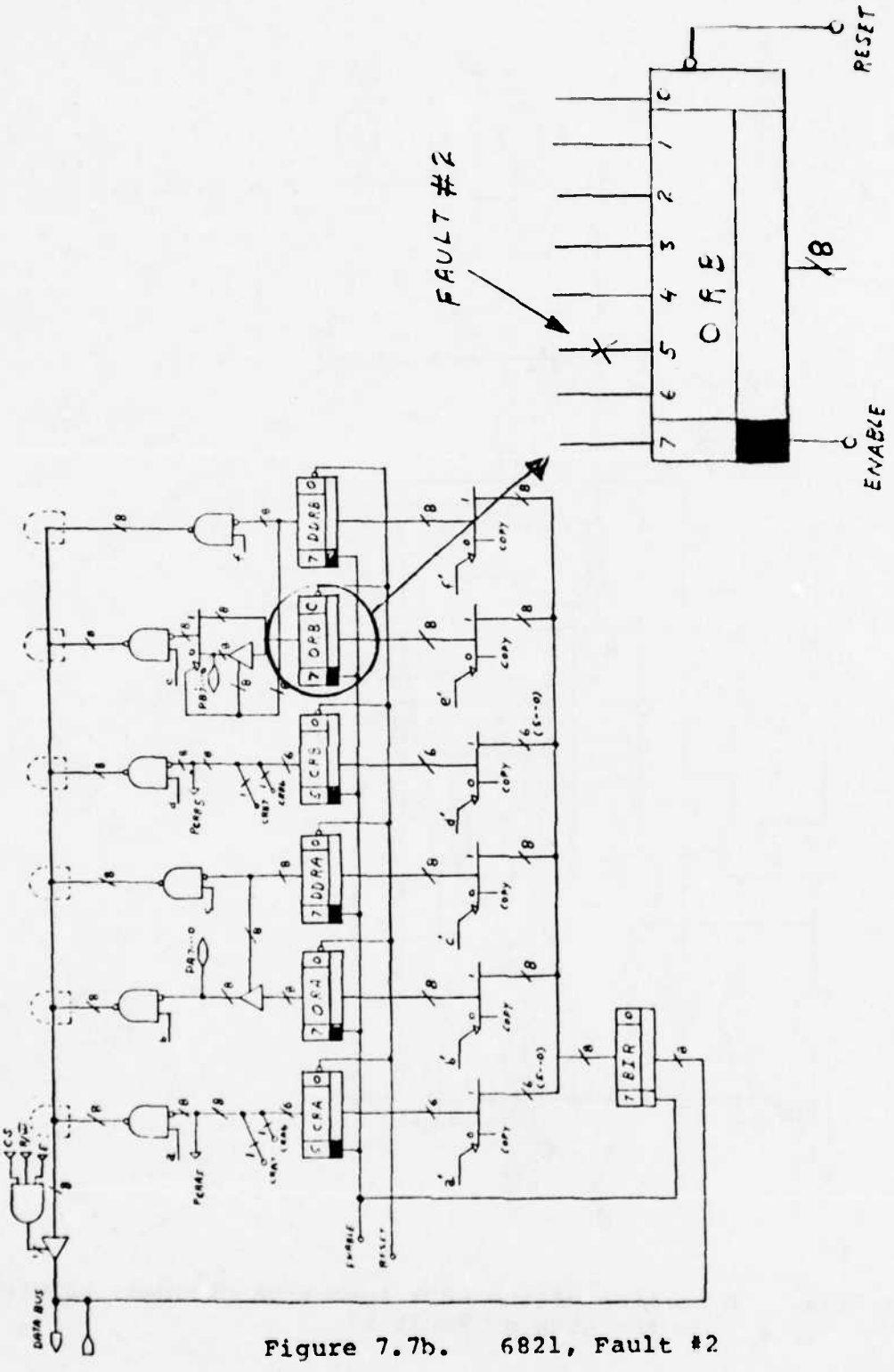


Figure 7.7b. 6821, Fault #2

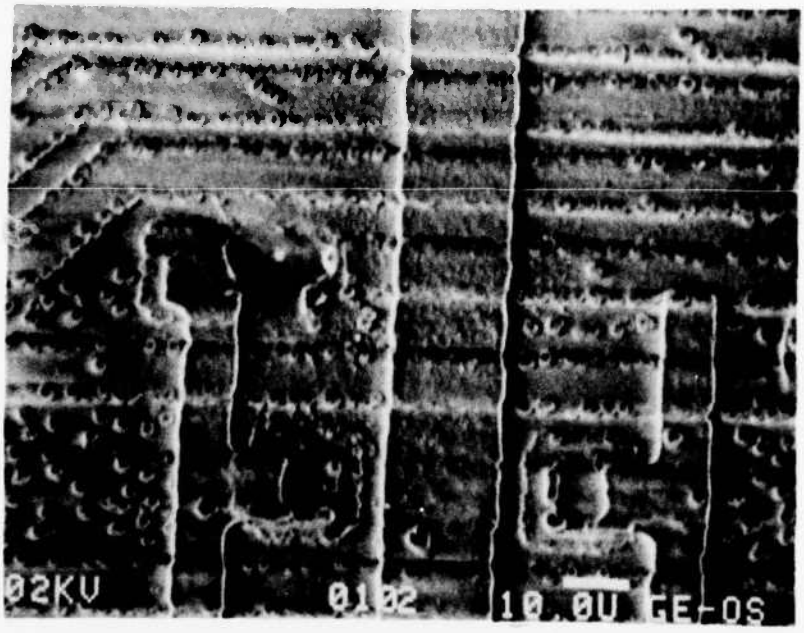


Figure 7.8a. Scanning electron microscope photograph of the 6821, showing the site of Fault #1

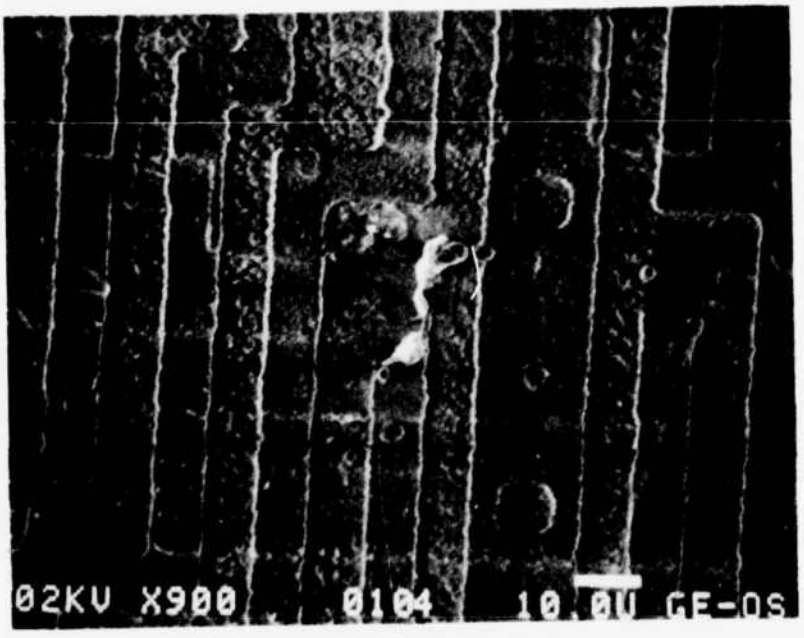


Figure 7.8b. 6821, Fault #2

use, such as being able to disable all writes to the registers; the 2901A required one register always to be loaded on every cycle.

The devices used in this effort were the Advanced Micro Devices Am2903 which had a datecode of 7942. The data sheets used were from AMD [2]. The Automaton Diagram for this device is shown in Figure 7.9.

7.5.1 Learning the 2903

The logic diagrams for this device were not available, but a Product Evaluation report had been prepared and supplied necessary information. Sufficient detail to reconstruct the ALU was not included, and so a reasonable implementation was chosen and written into the LASAR model.

This device was somewhat unique among those studied during this effort in that extensive bench testing was not necessary. Due to the investigators' familiarity with the 2901A there were no surprises, only a great deal of functional data to be consumed.

The LASAR model for this device was not completely debugged by the close of this effort, but is expected to be completed as part of an RADC in-house effort to characterize the 2903.

7.5.2 Testing the 2903

Test generation for fault isolation was not done for this device because the LASAR model had not been completed. However, if done as an RADC in-house effort, no great problems are expected as the 2903 will lend itself to testing by means of the recipe approach better than any large device encountered to date.

7.5.3 Test Implementation for the 2903

Since the LIT was not generated, no device test was implemented. However, an S-3260/3270 adaptor was built and verified using the technique described in section 7.4.3.

7.5.4 Tracing and Inserting Faults in the 2903

This was not done with the 2903.

7.6 2910

The 2910 is a Microprogram Controller capable of directly addressing 4096 words of control store (via a 12-bit address). The predecessors to this device, the 2909 and 2911, were only 4 bits in width but were expandable. The 2910 is not immediately expandable, as there is no carry out from the microprogram counter register incrementer, but more than 4K words of microprogram address space are seldom needed. In addition, the 2910 provides a great deal more flexibility in addressing modes.

The devices studied in this effort were the Advanced Micro Devices, Inc. Am2910 which had a datecode of 8009. The data sheets used were from AMD [2].

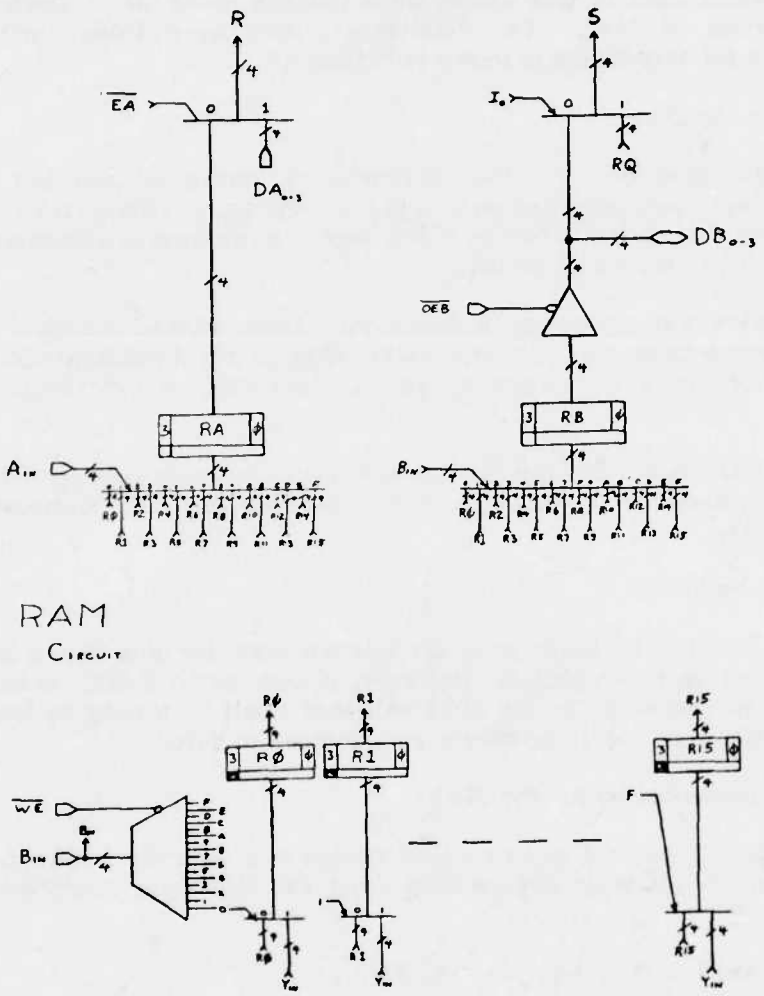


Figure 7.9. 2903 Automaton Diagram

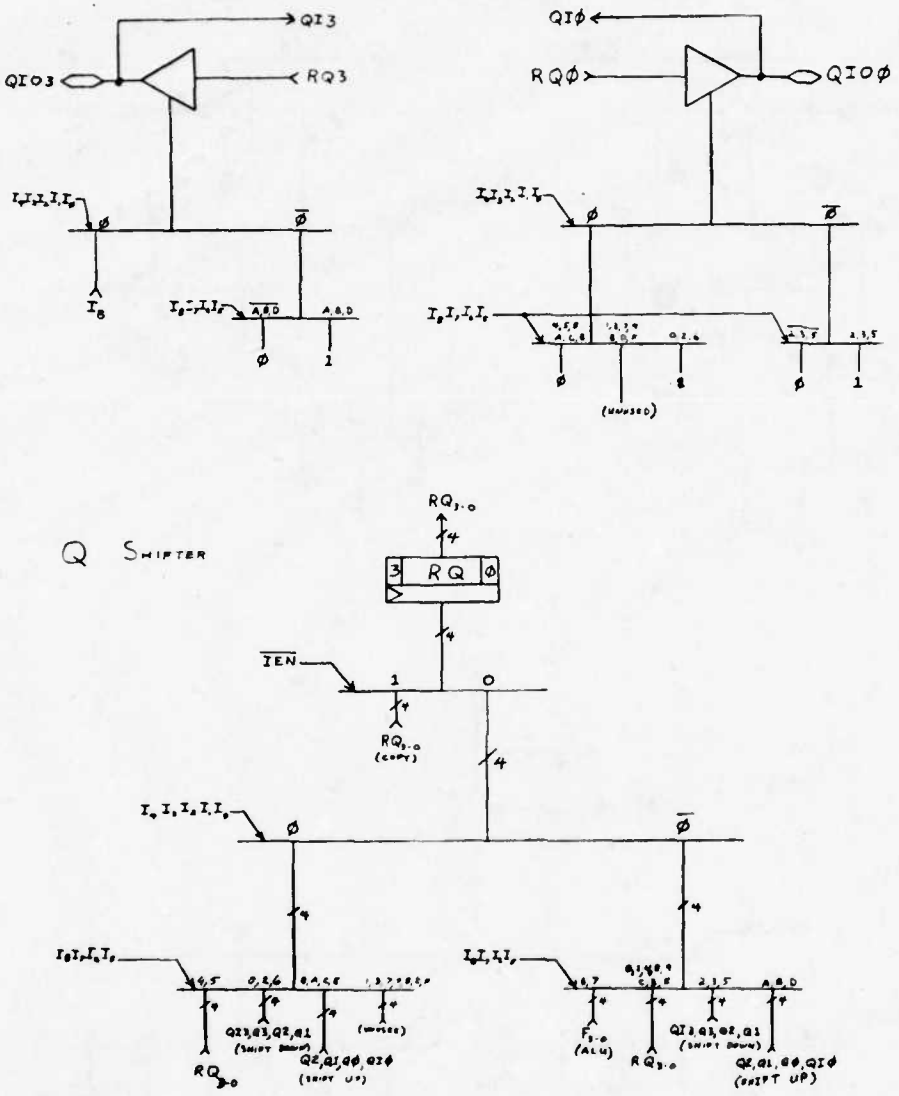


Figure 7.9. 2903 Automaton Diagram, continued

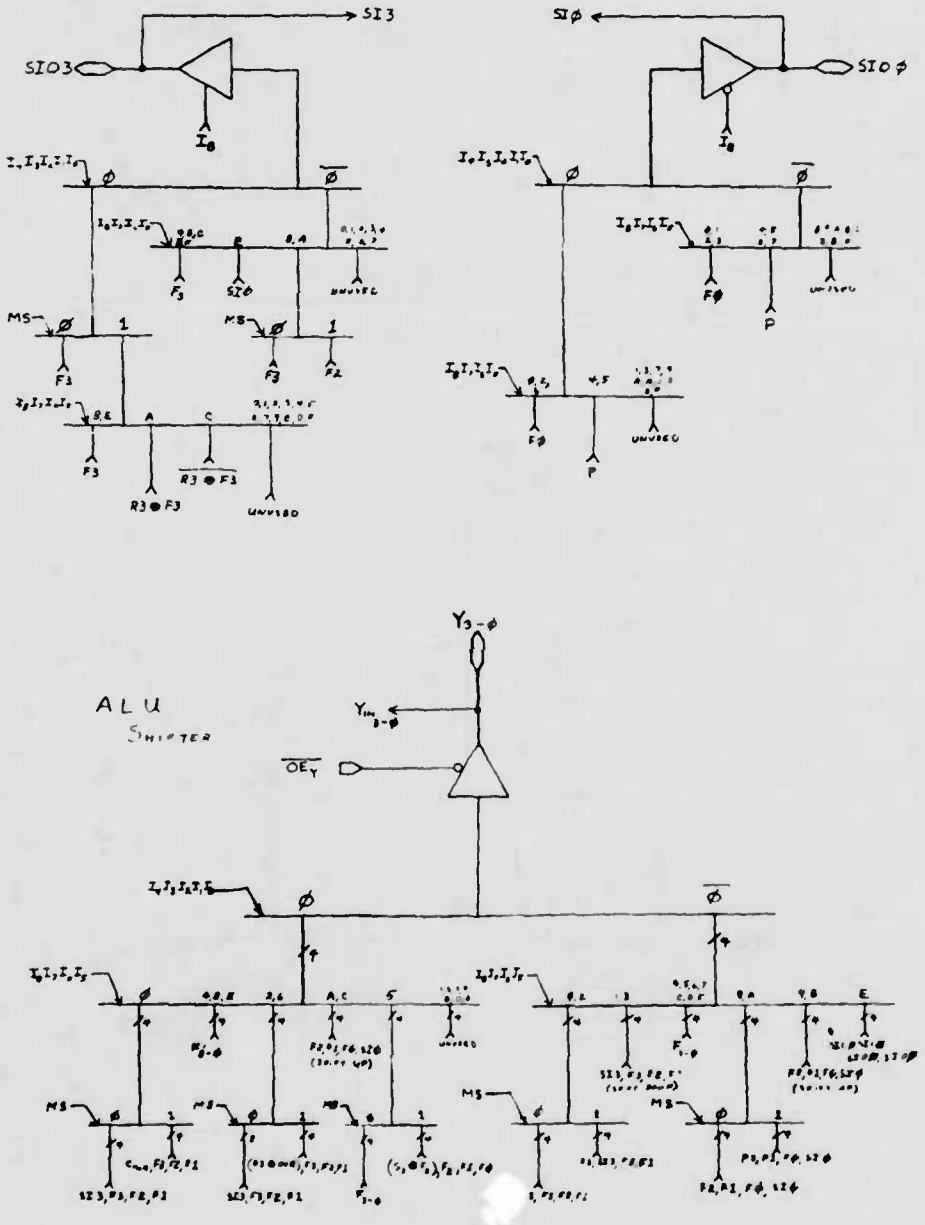


Figure 7.9. 2903 Automaton Diagram, continued

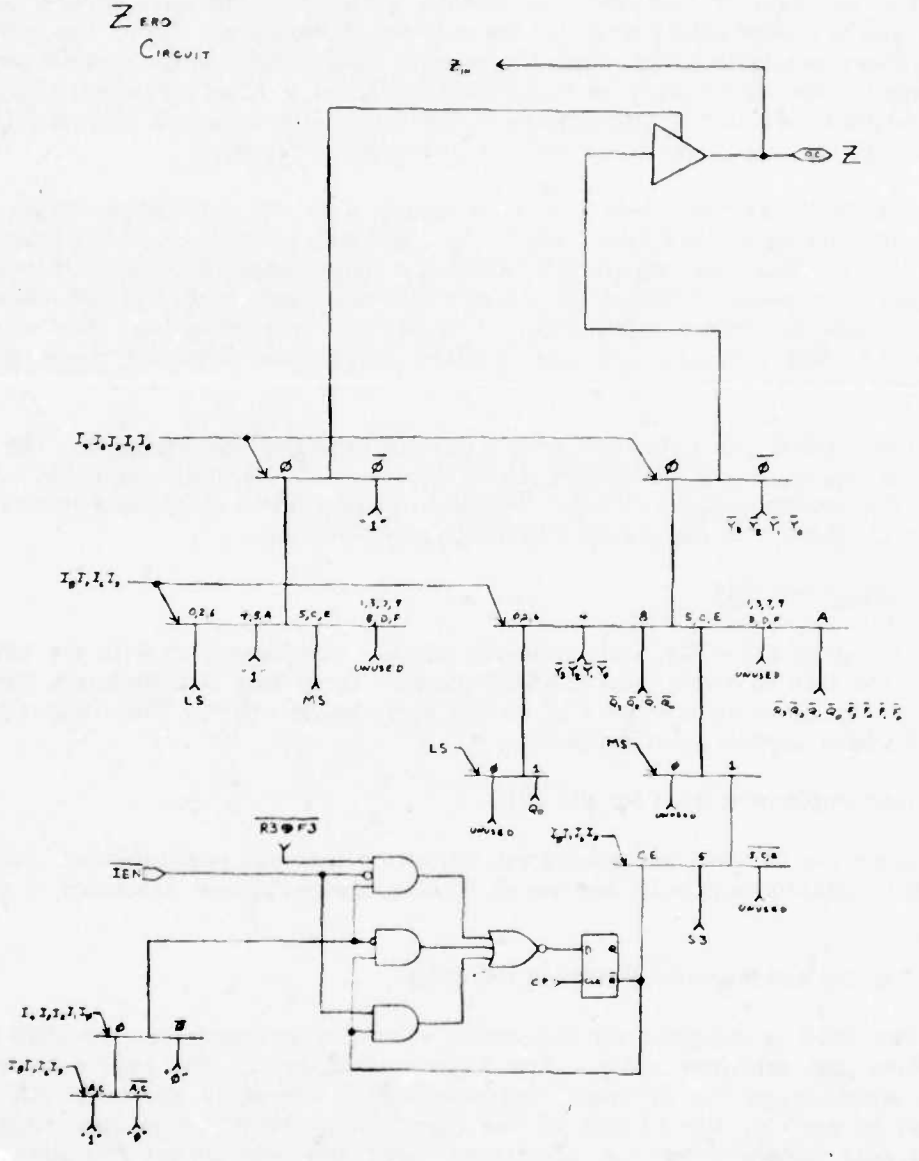


Figure 7.9. 2903 Automaton Diagram, continued

7.6.1 Learning the 2910

The function of the 2910 was clearly outlined in the data sheets, but this information was of course insufficient for the purpose of modeling. There was a Product Evaluation report available which gave the needed information about certain sections. AMD had supplied the schematics for the 2910 earlier, for a different project, but these were not consulted. All of the information in the Automaton Diagram (Figure 7.10) was garnered through the use of the PE report and through bench testing.

The 2910 was extremely simple to model, with the notable exception of the 5-word x 12-bit microprogram counter stack. It is actually a 4-word x 12-bit stack, with an intricate pair of "stack-in" registers to buffer the input/output (PUSH/POP) functions. A simpler implementation for this in the model could have been used, but the Automaton Diagram was made to reflect fairly closely the actual implementation. This was done for the planned fault isolation, although a more programmer-oriented model may be developed later.

The LASAR model for this device has not been debugged entirely. The major problem is, of course, in the PUSH/POP/HOLD sections of the stack circuitry. As the write signal is asynchronous, one problem is how to force LASAR to produce pulses (using one-shots, etc.) that do not cause race conditions in the model.

7.6.2 Testing the 2910

Test generation for fault isolation was not completed, as with the 6821 and 2903, due to the lack of a complete LASAR model. Once this is completed, the fault isolation test may be written as part of an RADC in-house effort. The recipes for the various blocks have already been defined.

7.6.3 Test Implementation for the 2910

Since the LIT was not generated, no device test was implemented. However, an S-3260/3270 adaptor was built and verified using the technique described in section 7.4.3.

7.6.4 Tracing and Inserting Faults in the 2910

The 2910 is designed and fabricated in the same manner as the 2914 so the same problems and solutions apply. The logic is laid out on the device in distinct identifiable areas, (e.g., 4 x 12 stack, instruction PLA and stack pointer). All of the logic related to each of the 12 bits of the input/output structure is concentrated in repeated layouts bordered by the associated input (D) and output (Y) pins. This organization simplified fault identification on this device. It also affected the choice of fault sites giving definite divisions of logic which would be useful to test. The fault areas chosen were an output of the instruction PLA, carry-in to the incrementer logic of one of the repeated bits and the FULL function output from the stack pointer. The control line issuing from the PLA was chosen because although it functionally affects only one operation of the device, this operation cascades through all 12 bits. The control line chosen regulates the R register decrement function and was faulted in such a way that the register is always decremented unless a load command is being executed. Figure 7.11a is a portion of the Automaton Diagram for the 2910 showing the effect of the above fault.

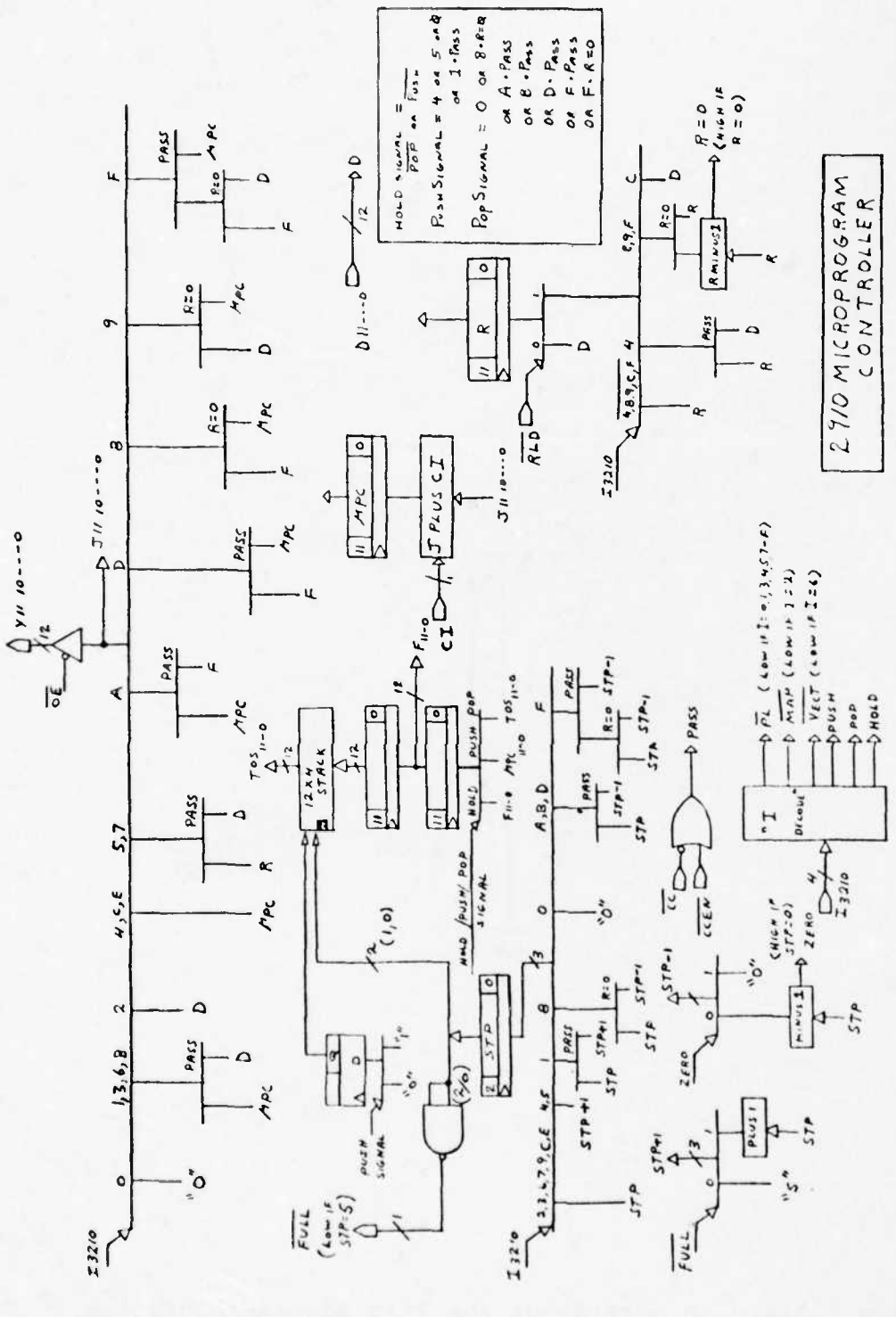


Figure 7.10. 2910 Automaton Diagram

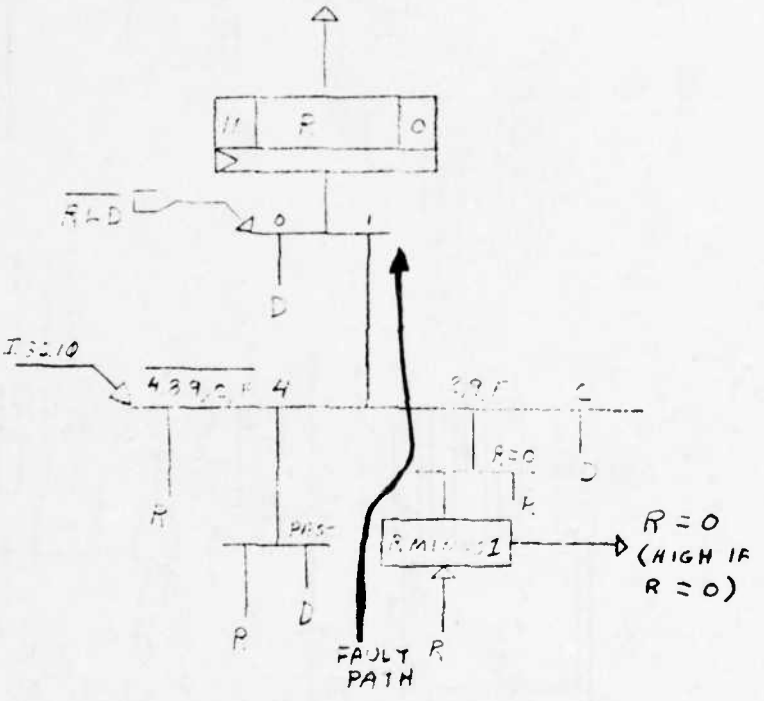


Figure 7.11a. A portion of the 2910 Automaton Diagram, showing the site of Fault #1

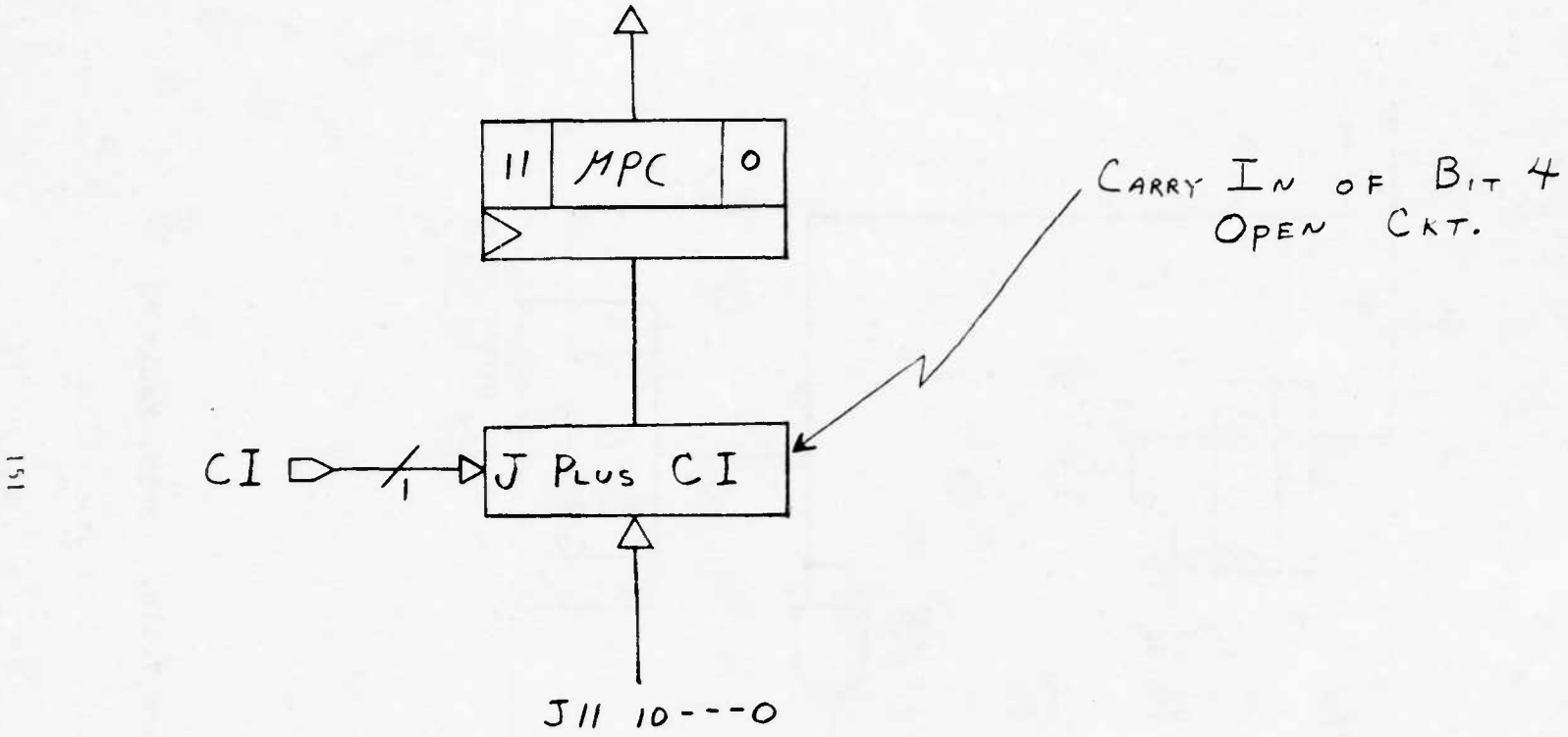


Figure 7.11h. 2910, Fault #2

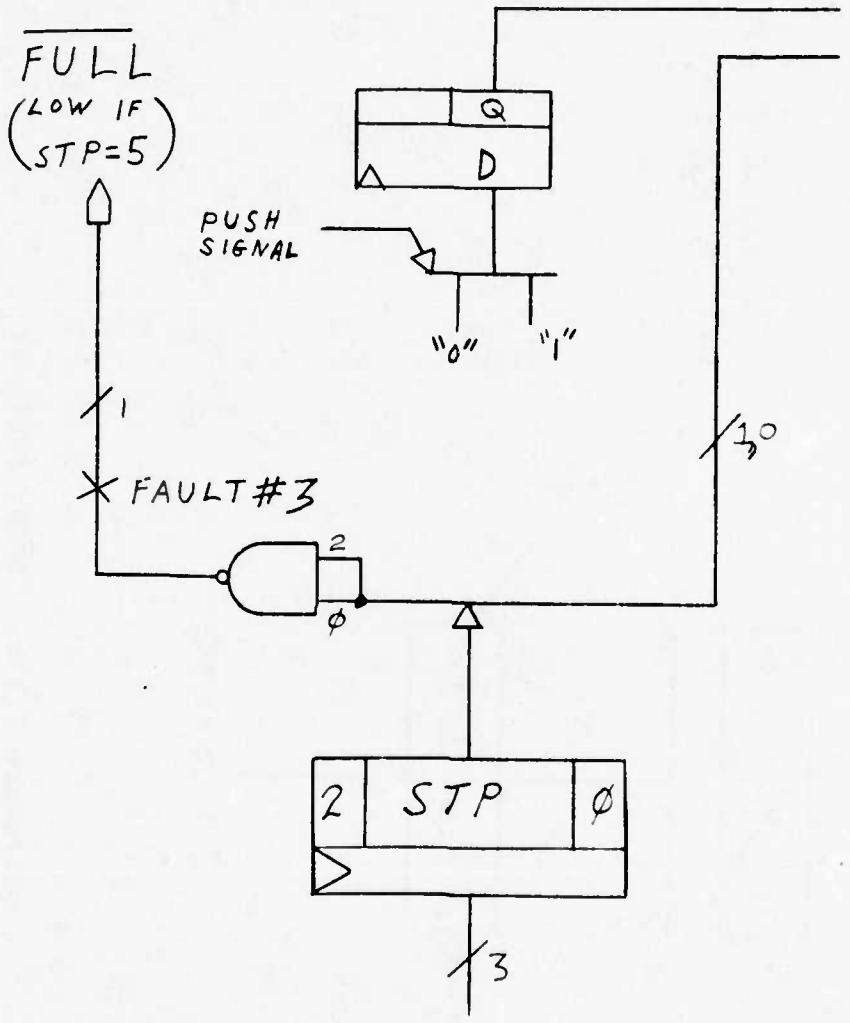


Figure 7.11c. 2910, Fault #3

Fault 2 prevents bit 4 of the uPC incrementer from recognizing the active level of the CI input. The other uPC bits are not directly affected. Figure 7.11b shows the area of the fault.

Fault 3 prevents the full signal from the stack pointer from going active. Figure 7.11c shows the fault site on the Automaton Diagram.

Figure 7.11a is a detail of the logic for the R register and its decrement function showing more accurately the fault inserted in the control line. This logic diagram comes from the RADC Product Evaluation of the Advanced Micro Devices 2910. Figure 7.11b details the positioning of Fault 2 showing the logic for the bit slice in question. Figures 7.12a, 7.12b, and 7.12c give optical photographs of the faults inserted.

References:

1. Schottky and Low-Power Schottky Data Book, Advanced Micro Devices, Inc.

2. The Am2900 Family Data Book, Advanced Micro Devices, Inc.

Note that the data covering AC and DC parameters for the 2903, 2910 is incomplete or lacking. Other data sheets were found around the laboratory which did have the information. The authors are unsure why these data were omitted from this book.

3. A Survey of LSI Test Methodology, RADC-TR-79-85 (AO 70733), pp. 12-13.

4. MIL-STD-1553B, Military Standard, Aircraft Internal Time Division Command/Response Multiplex Data Bus, 21 September 1978.

5. HD-15530 CMOS Manchester Encoder/Decoder Data Sheet, Harris Corporation.

6. MC6821 Peripheral Interface Adaptor (PIA) Data Sheet, Motorola Semiconductor Products, Inc.

7. M6800 Microprocessor Applications Manual, Motorola Semiconductor Products Inc.

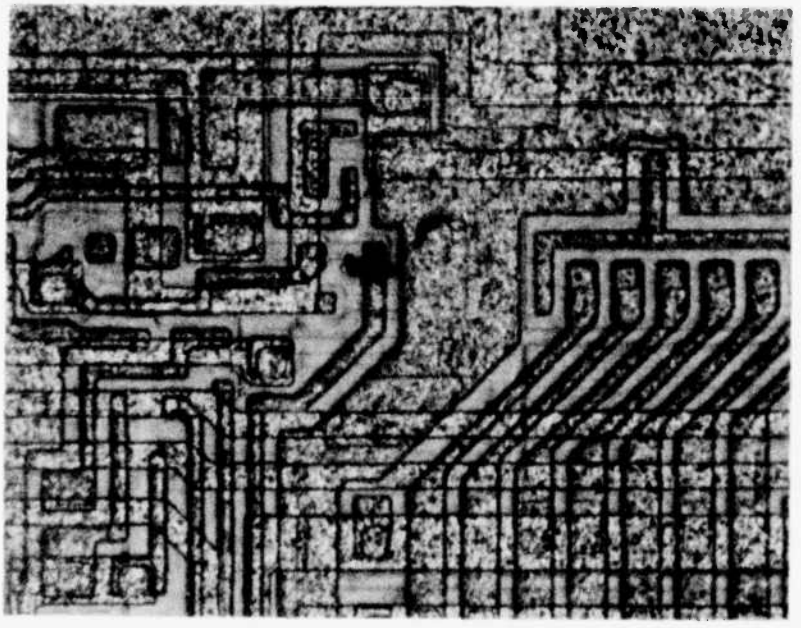


Figure 7.12a. Optical photograph of the 2910, showing the site of Fault #1

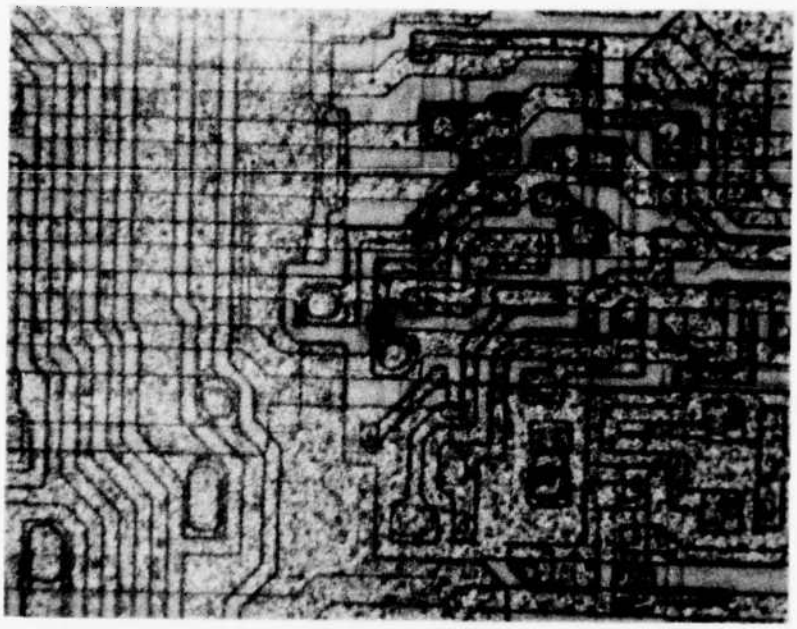


Figure 7.12b. 2910, Fault #2

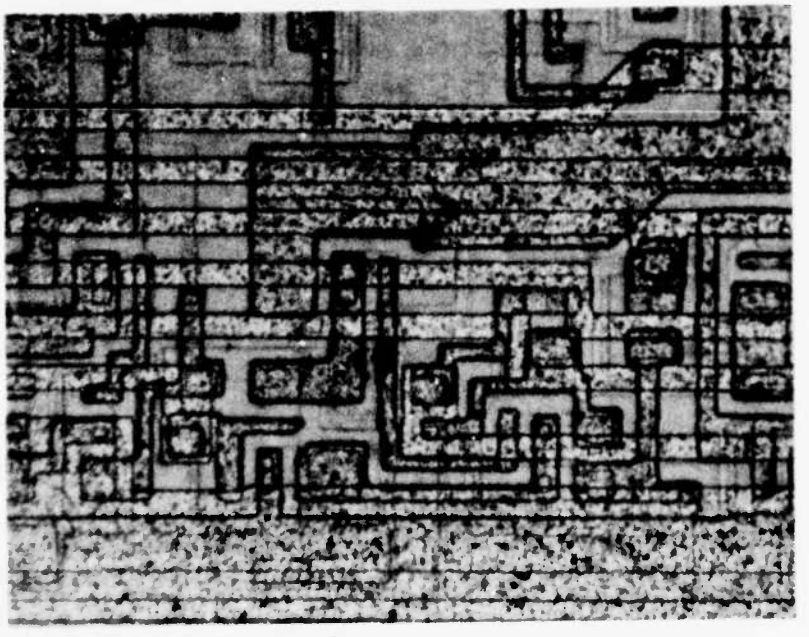


Figure 7.12c. 2910, Fault #3

8. Status

This effort provided the vehicle for investigations into several areas. Test generation, fault isolation, techniques for the tracing of internal circuitry of devices, slash sheet generation, and software development were a few of these areas.

In-house expertise was developed and improved at RADC with the intention of becoming better able to review work done by others, and to perform these tasks as required for quick reaction projects or the handling of proprietary material.

The following will be a summary of the status of important areas which were studied. These areas are Slash Sheets, Fault Isolation, and Software. These discussions will be followed by some recommendations.

8.1 Status of Slash Sheets

Two slash sheets were completed for review under this effort. The first was for the 2914, and was assigned the number M38510/444. The Table III LIT was developed using the techniques described in section 3 of this report, and was in fact the reason that these techniques were developed. ALICE was designed and implemented specifically for this LIT. Some of this was done in-house at RADC before the start of this effort.

The remainder of Table III and the balance of the slash sheet were done entirely under this effort.

This slash sheet has been sent out for review by DESC and the industry.

The second slash sheet was for the 15530, and was assigned the number M38510/501. This slash sheet was originally a draft sent in by the manufacturer of this device, and provided the basis for the work done with the 15530 under this effort.

Although this slash sheet has been sent out for review, comments have not yet been received by RADC.

The Table III format for these slash sheets differs from the usual format. It was attempted to make the Table III readable by humans through the use of vectors of pins and comments. The test vectors are available in both the Sentry and S-3260/3270 formats, on magnetic tape. It is hoped that anyone who needs to test these devices will be able to find a usable machine-readable version of the vectors, and will be able to implement the same timing modes.

8.2 Status of Fault Isolation

The fault isolation routines have demonstrated the ability to locate the failing element within the limitations set by the model and the LIT. Most of the deficiencies are caused by differences between the model and the actual device.

There are several improvements that could be made to the routines in operating time and operator interface. In particular, reducing the time required to perform the search through the data files would improve the operating speed directly.

8.3 Status of Software

The various software packages developed for use in test generation and test implementation were an integral portion of this effort. These packages and utilities, described in section 3, allowed the more mechanical tasks to be automated, leaving more time for the important jobs.

It is unfortunate that more effort could not be devoted to their development. As is usual in these cases, a particular problem arose, and a utility was written to solve it. Later, a slightly different problem required a modification of the software. Fortunately, the software generated was well-structured and commented sufficiently to permit an orderly set of "patches."

On the S-3260/3270, the well-structured software patches did not always work. Simple modifications often put a program outside the core limitations of the computer, and forced a roundabout approach. Lack of modularity of the TEKTEST language also hampered progress.

There are several programs for the RADC Multics and S-3260/3270 that have not been completed, or are not sufficiently general to satisfy the authors. These programs will be fixed or completed either by RADC, working in-house, or by the contractor's personnel when they wish to use these utilities for other efforts. These packages are not explicitly listed here as they are expected to be completed by the time this report has been printed.

Following is a brief summary, in outline form, of the status and projected changes for some of the software packages developed during this effort. The list is divided into separate sections for Multics and S-3260/3270 utilities.

8.3.1 Multics Utilities:

ALICE, TECO, etc. -- These are complete as they are, but several improvements are planned. ALICE needs a more powerful expression evaluation capability, which will allow it to perform more complicated pattern generation algorithms.

TEKTRONIX, SENTRY, SENPASS, SENPASS2 -- These utilities, which convert patterns to/from ATE languages, work only for a subset of these languages. For the Sentry Utilities, this is an extremely limited subset. It is hoped that these can be augmented in the near future so that they more fully implement these languages. A great deal of experience was garnered during this effort, and further refinements of these utilities will probably implement Virtual Machines that will be more faithful to the actual ATE.

NQUINE, SWAPCOL, PLAGEN -- These are the basis of Computer-Aided Design packages which will make the modeling of the DUT much easier. Although these techniques work at the present time, the cost of regularly running these on very large networks is prohibitive. Vast reductions in processing time and storage requirements will be necessary.

ATE_TAPE_UTIL -- These programs, which allow compatibility between Multics and ATE on a magnetic tape level, require considerable improvement. At the present time, file transfers between the Multics facility and the S-3260/3270 take place on a retail basis, rather than a wholesale basis. In other

words, it should be possible to send or receive files in bulk, by causing Multics to simulate the file structure (and storage system) of the S-3260/3270. A comprehensive emulation of the whole system is needed (see section 8.4.4).

8.3.2 S-3260/3270 Utilities

PATPLO, WAVE, etc. -- At present these programs use the device pattern file and pinlist for all plotting information. Additional capability of plotting LIT errors information against the pattern information would be useful in analyzing failure information. Other additions to the program should include plotting the pattern file, with the appropriate phases, to aid in the analysis of a device test.

BATRED -- The BATRED file is presently stored in each device UID and is used only to run DATRED (see section 3.2.7). The user must enter RUN (name):SIM to run the other programs in sections 3.2.8 through 3.2.16. These programs should be transparent to the user, since they are not DUT-dependent. Changing BATRED and adding a user-interface program would simplify the operation of the S-3260/3270 utilities by eliminating the need for the user to enter program names and automatically assigning all the required devices and files.

MATSIM -- A first generation of this program was developed and is presently operating. This version assists the operator in the specification of each argument and outputs a ".ASC" file for the device test. No special system checks or opcode checks have presently been incorporated in this program. Future revisions to this program should include these checks as well as the capability of verifying that existing SIMPL.ASC files will execute on a particular piece of ATE. Other possibilities include the capability of reading a computer version of a device slash sheet and automatically developing the SIMPL, PIN, PINLIST, and SIMTEK files for a particular device. Developing this capability was beyond the scope of this effort.

SIMPL -- The SIMPL opcodes for all DC tests have been developed. Standard AC tests such as input setup and hold times as well as propagation and transition times can also be executed using the SIMPL opcodes. Special tests such as a Galloping RAM test require special hardware pattern generators and have not been implemented. However, future updates to SIMPL should include the implementation of these tests on the various test systems.

SIMTEK -- The SIMTEK program will presently execute all existing SIMPL opcodes, but as it requires approximately 13K of S-3260/3270 memory for testing. Some reduction in memory usage should be made. For example, the routines in SIMTEK that check the validity of the SIMPL parameters against the system limits could be moved to the MATSIM program. A further reduction in memory would result from having MATSIM compile a SIMTEK that contains only the sections needed by a given SIMPL file. The final objective could be to rewrite SIMTEK to develop an interpretive program that would implement SIMPL statements directly.

8.4 Recommendations

The following sections briefly discuss some suggestions for simplifying device specification development, interpretation, and usage.

8.4.1 Slash Sheet Format

The slash sheet format should be standardized in a format that is both human and machine readable. This would allow development of programs to convert the Table III tests to the format required by the user's ATE and to present the LIT and AC tests as waveforms (or other format) that a user can readily understand. Also, the slash sheets could be stored on magnetic tape for easy transmission among interested parties.

8.4.2 Device Modeling

This effort has demonstrated repeatedly that the development or evaluation of an LIT can be no better than the knowledge of the actual device implementation. This knowledge leads directly to accurate models, which then allow a test generation program to properly generate or evaluate an LIT, resulting in a realistic TCL and fault dictionary. Thus, the amount of device implementation details provided by a manufacturer has a direct impact on the accuracy, usefulness, and cost of a slash sheet.

8.4.3 Specialized ATE Capabilities

As noted in section 2.4, each ATE has special features that facilitate implementation of some test types while restricting others. These capabilities also can make transferring a test from one ATE to another very difficult. For this reason, the slash sheet tests should always be written in a simple and standardized format that does not use the special capability of any ATE. This simple format has the additional advantages of improving an user's understanding of the device, simplifying slash sheet storage and manipulation by computer and allowing test implementation on an ATE that is not familar to the slash sheet writer.

8.4.4 Desired CAT Tools/Techniques

This effort relied heavily on the use of Computer-Aided Testing. A great many utilities were created which appear to be unique in this field. However, there are still several areas which could bear improvement.

One useful item would be a hardware simulator which could take an Automaton Diagram directly as input. It is quite possible that fault simulation would prove difficult on that level, but this utility would make the original model debugging much faster.

Along similar lines, a tool for synthesizing asynchronous networks of (hopefully) arbitrary complexity would be valuable. Such a tool could have been used when developing the handshake logic for the 6821 model during this effort. Other Computer-Aided Design tools would also have been welcome.

Complete simulators or interpreters for the S-3260/3270 TEKTEST and Sentry FACTOR languages would also have been useful. If the simulation also were able to allow debugging of hardware control statements, off-line implementation of tests would be much easier. Usually, the bottleneck in implementation is the availability of test table time. In production environments this availability is severely limited. In

addition, minor test table problems can cause a correct program to fail, causing engineers to waste time fixing something that is not broken. A reliable software emulation of the table response, providing trace capability, would be invaluable.

A utility which would unwind complicated clocking schemes (see section 3.1.12) will be implemented eventually. This will be a large step toward 100% compatibility among different types of ATE.

Appendix A. Automaton Diagrams

It was attempted, in this effort, to model each DUT in two forms, both the LASAR representation and Automaton Diagram. In one case, the modeling of the 15530 Manchester Encoder/Decoder, it became evident that this was difficult or impossible, as the Automaton Diagram was no simpler than the original schematic. This was due to the lack of parallelism in the data paths. With other devices, the Automaton Diagram was an invaluable tool for understanding the operation of the device (as in test generation) and for guiding the design of the LASAR model. In the latter case, heavy use was made of Product Evaluation Reports (generated by RADC and contractors) containing information about the specific implementations used for certain functions. Without this information, LASAR modeling would still have been possible, but fault detection/isolation results would not have been as accurate or faithful to the actual devices.

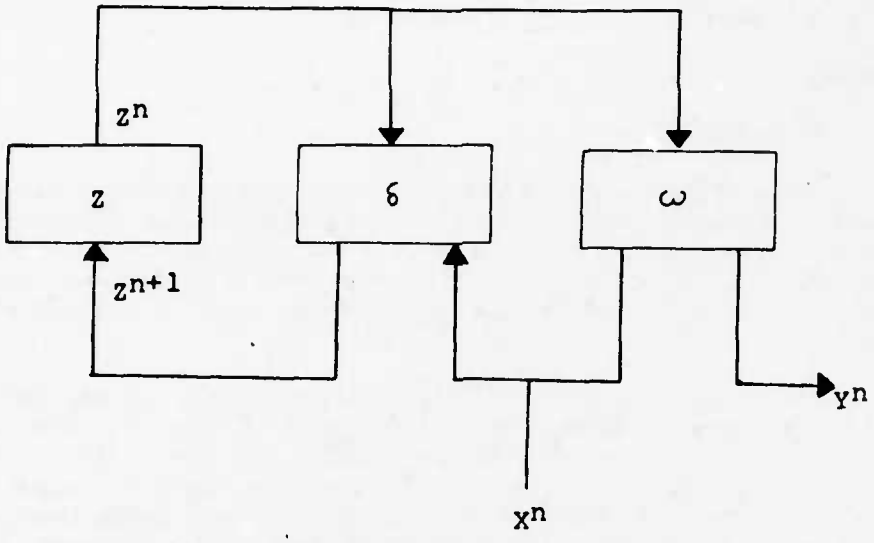
The Automaton Diagram is a tool for both the design and diagnosis of sequential state machines. Essentially, an Automaton Diagram describes the behavior of the machine in terms of memory elements (delay elements) and combinatorial logic. The memory element of choice is the edge-triggered master/slave D-Flip-Flop, used in an array called a "register," with data flow controlled by means of data selectors. However, most existing designs require the use of other types of memory elements, and the data flow is seldom implemented neatly with only data selectors. Occasionally, one-shots, SR-Flip-Flops, and discrete gates must be introduced. Examples will be provided shortly.

The generalized sequential state machine is shown in Figure A.1. As the equations in the figure show, the next state is derived from the present input and the present state by the use of "delta," the next state mapping function. Delta is usually implemented as combinatorial logic. In a similar manner, the present output is derived from the present input and present state using "omega," the present output mapping function.

The astute reader will have noted that there is no clock signal shown in Figure A.1. Although most current designs will employ such a signal for synchronization, the memory elements denoted by "Z" are included solely for delay, and could for example be implemented as acoustic delay lines. Their only purpose is to allow the system to settle (become determinant) before the next cycle, or change, occurs. In practice of course, the synchronization is most easily accomplished by the use of the external clock signal, which may be implemented in many ways, including: two- or multi-phase clocks; on-board timing generators; or one-shots. When drawing the Automaton Diagram, if there is a single clock signal, it is usually omitted entirely. The notation for the triggering mode of the memory elements will be discussed shortly.

The total number of elements in Z is the total number of feedback paths in the automaton. The delta and omega blocks shown in Figure A.1 are memoryless, meaning that their outputs depend solely on their current inputs (ignoring propagation delay time). The automata considered here differ from "asynchronous" machines that may operate in the "fundamental mode" or "pulse mode" [1].

The Mealy Machine shown in Figure A.1 is the most general case, but it is necessary at this point to restrict the definition. In that figure, the equation



$z^{n+1} = \delta(x^n, z^n)$

$y^n = \omega(x^n, z^n)$

z = Memory Elements

δ = Next State Mapping

ω = Present Output Mapping

z^n = Present State

z^{n+1} = Next State

x^n = Present Input

y^n = Present Output

Figure A.1. Mealy Machine

$$Y^n = \omega(X^n, Z^n) \tag{A.1}$$

is now assumed to be indeed a function of both of its arguments, not only one,

$$Y^n = \omega(X^n), \tag{A.2}$$

or the other,

$$Y^n = \omega(Z^n). \tag{A.3}$$

The condition described by Eq. (A.2) is of course trivial, reducing the machine effectively to a single combinational block. The special case described by Eq. (A.3), however, is a very powerful one, and is called a Moore Machine. This is shown explicitly in Figure A.2. The distinction between the behavior of a Mealy and a Moore Machine is so great that Eq. (A.1) must always be considered to be a function of both of its arguments.

The output of a Moore Machine changes only at discrete time intervals, in contrast to the Mealy Machine where any change of an input may potentially cause an output transition. This implies that, in general, two Mealy Machines may not be connected directly together in a loop. This is shown in Figure A.3, along with a much-simplified schematic of the unintended, and illegal, feedback path. This may be avoided by making one of the Machines into a Moore Machine. Now, no uncontrolled feedback path can occur.

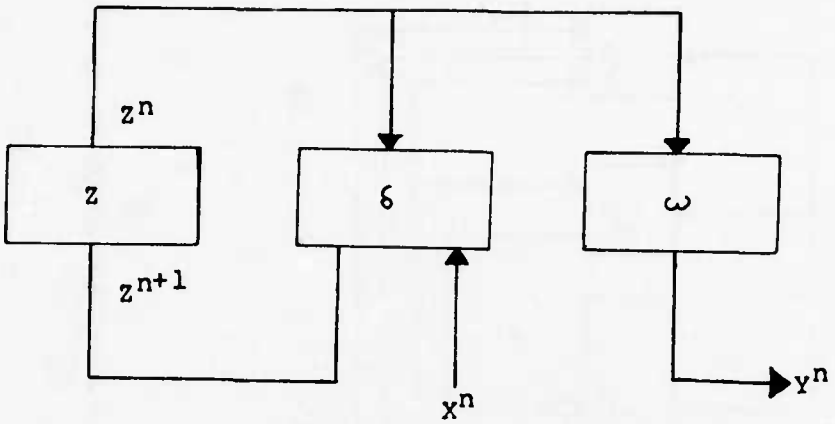
One may take Eq. (A.3) a step further, and cause the omega function (present output mapping) merely pass its argument unchanged, yielding

$$Y^n = Z^n. \tag{A.4}$$

This is called a Medvedev Machine, and is shown in Figure A.4. When one uses a 2910 Microprogram Sequencer to address a ROM, with a pipeline register to buffer the output of the ROM, one has built a Medvedev Machine. The 2901 or 2903 Operation Unit in the same system is a Mealy Machine.

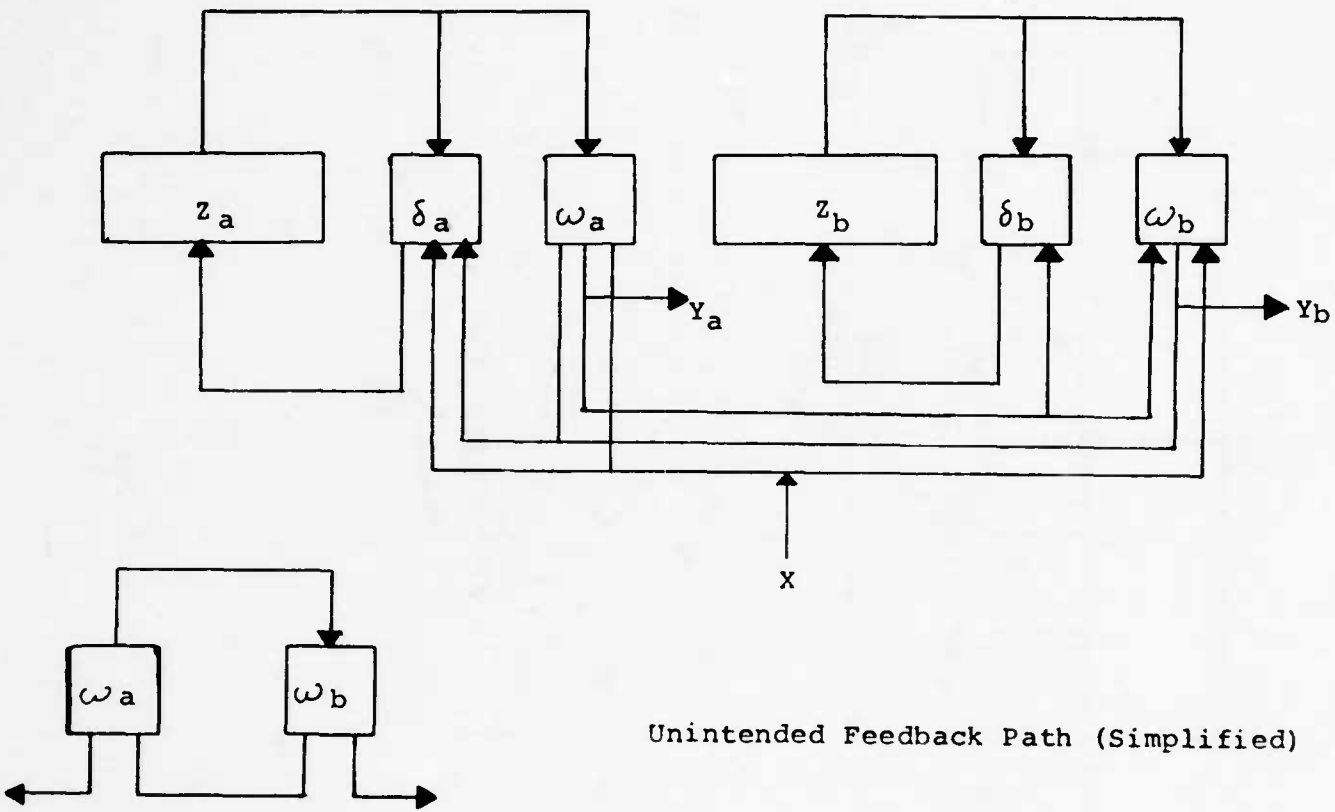
The symbols used in the construction of an Automaton Diagram have not as yet been completely standardized, as it appears that each new DUT requires at least one new element. However, the main functional blocks are relatively constant, and are generally drawn in the manner indicated by the following description.

Registers and Flip-Flops form the core of the Automaton Diagram. An eight-bit, positive edge-triggered register is shown in Figure A.5. For convenience, the name of the output signal is usually taken to be the same as the name within the block. In addition, (a) through (h) on the same figure show the various clocking schemes usually encountered. Note that (c) and (d) are more properly called "latches" as opposed to "registers," and are usually not sufficient singly to satisfy the requirements of a memory element "Z." This is because they are transparent for a portion of the clock cycle, allowing *uncontrolled* feedback. Of course, tricks may be used, such as restricting the width of the "transparent" portion of the clock cycle to a duration safely shorter than the shortest propagation delay time, or transition time, through delta, but that is beyond



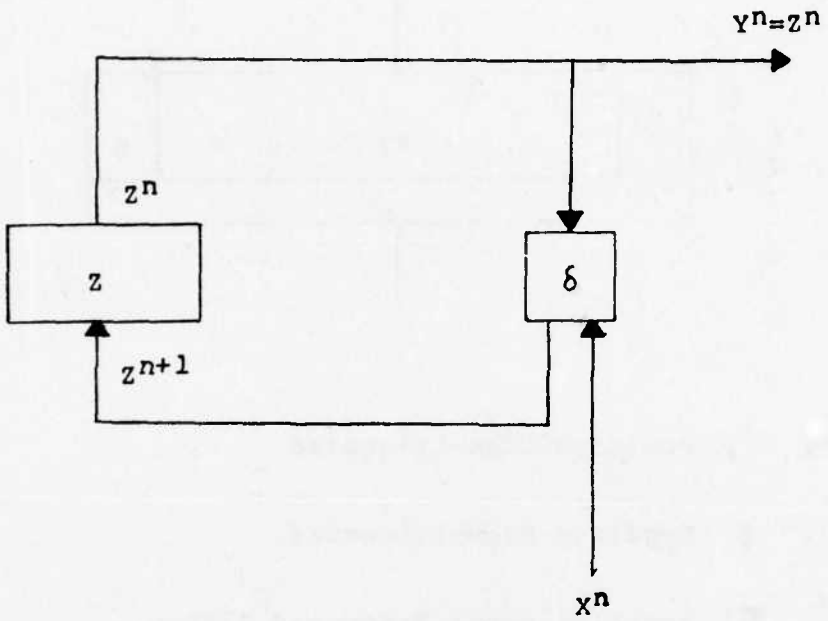
$$z^{n+1} = \delta (x^n, z^n)$$
$$y^n = \omega(z^n)$$

Figure A.2. Moore Machine



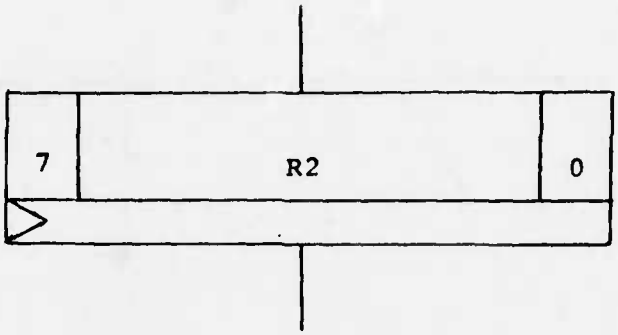
Unintended Feedback Path (Simplified)

Figure A.3. Mealy/Mealy Machine



$$Z^{n+1} = \delta(X^n, Z^n)$$
$$Y^n = Z^n$$

Figure A.4. Medvedev Machine



A. ▷ Positive Edge-triggered

B. ◗ Negative Edge-triggered

C. □ Positive Level Triggered (latch)

D. ■ Negative Level Triggered (latch)

E. ▬ Neg. Level Master, Pos. Level Slave

F. ▬ Pos. Level Master, Neg. Level Slave

G. ⋈ Pos. Edge Master, Neg. Edge Slave

H. ⋈ Neg. Edge Master, Pos. Edge Slave

Figure A.5. Register (or latch)

the scope of this discussion. Their usual purpose is to discretely and explicitly form a master/slave register, such as in case (e) or (f).

One may consider (e) and (f) to be functionally equivalent in some cases to (a) and (b), respectively, and that is indeed how (a) and (b) are often implemented. These are not true edge-triggered devices however, and may be inappropriate for use with the scheme discussed in the next paragraph.

The terms "setup" and "hold" time, while familiar and adequate when describing memory elements by themselves, are unfortunately insufficient to describe the dynamic properties of automata, particularly those with deltas and omegas possessing very short propagation delay times. The concepts of "decision" and "transition" intervals are more appropriate, and are discussed in detail in another report [2]. It is sufficient to say at this time that the cases described in (g) and (h) provide the only absolutely reliable operation in a large, high speed system, where an arbitrarily high degree of stability may be achieved by adjusting the width of the high portion of the clock pulse, for (g), or the low portion of the clock pulse, for (h). The edge-triggered flip-flops used here should be truly edge-triggered, not implemented as master-slave latches. Due to cost considerations, however, no one has yet implemented this scheme in LSI or VLSI. It is quite possible that the implementation of VHSIC will find this necessary.

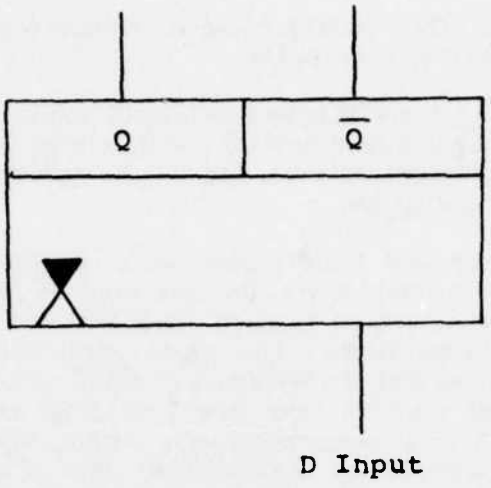
As in the case of registers, D-Flip-Flops may enjoy the same wide range of clocking options. An example is shown in Figure A.6.

Although the D-Flip-Flop is the easiest memory element to work with in design of automata, it is necessary on occasion to include SR- or JK-Flip-Flops, and other memory elements which tend to muddy the dividing lines between combinatorial logic and feedback paths [3]. They may, however, simplify the implementation of a complex function.

The data selector is a heavily-used element in Automaton Diagrams. The data selector (also called a multiplexer) has a single output (which may be a vector of "bits"), and many inputs (also vectors, each with the same bit width as the output vector). A control vector selects which input is fed to the output, with disregard for the state of the other inputs. An example is shown in Figure A.7.

While schematically convenient for humans to understand, the data selector is customarily implemented as in Figure A.8. Whenever possible, gate-level modeling (LASAR models) should reflect this. Here, decoding logic (which may serve the entire device) produces the necessary control signals. Note that a common failure mode would be failure to select any operand, which would default to a logical zero output if implemented as in Figure A.8.

A useful element for modeling a register array address selector is the demultiplexer, where an input vector is decoded into many bits, one and only one of which is active at any time. This is shown in Figure A.9. Occasionally there may be an enable signal to the demultiplexer, which can cause no signal to be active, thus no register to be selected.



Positive Edge-triggered Master,
Negative Edge-triggered Slave

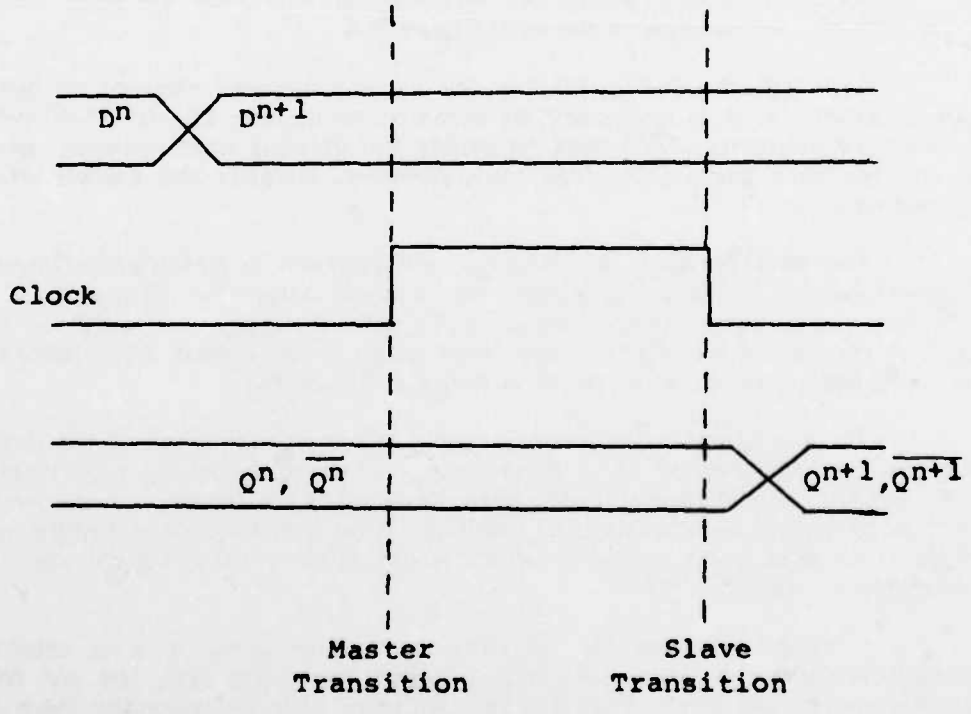
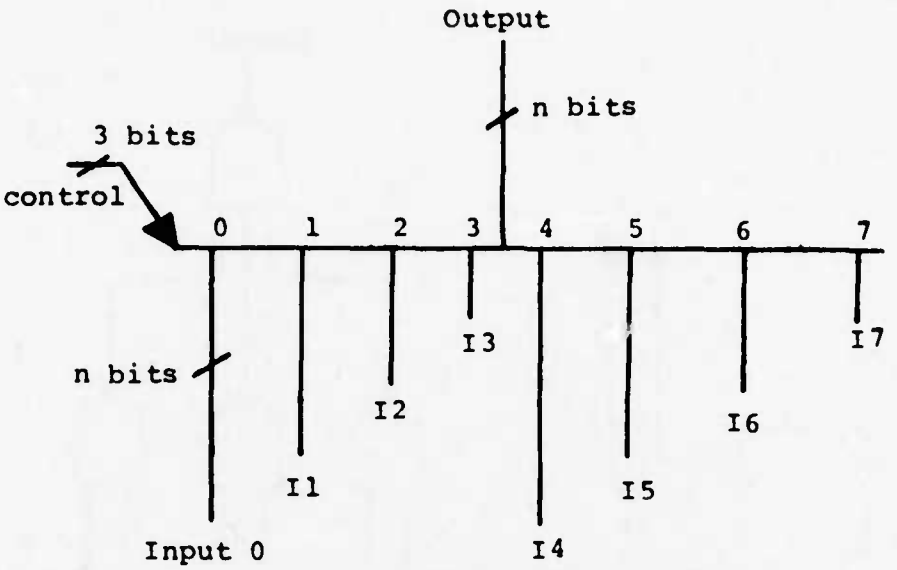


Figure A.6. D-Flip-Flop



8 inputs

n bit-wide data path

with K inputs, control path must be $\lceil \log K \rceil$ bits wide

Figure A.7. Data Selector

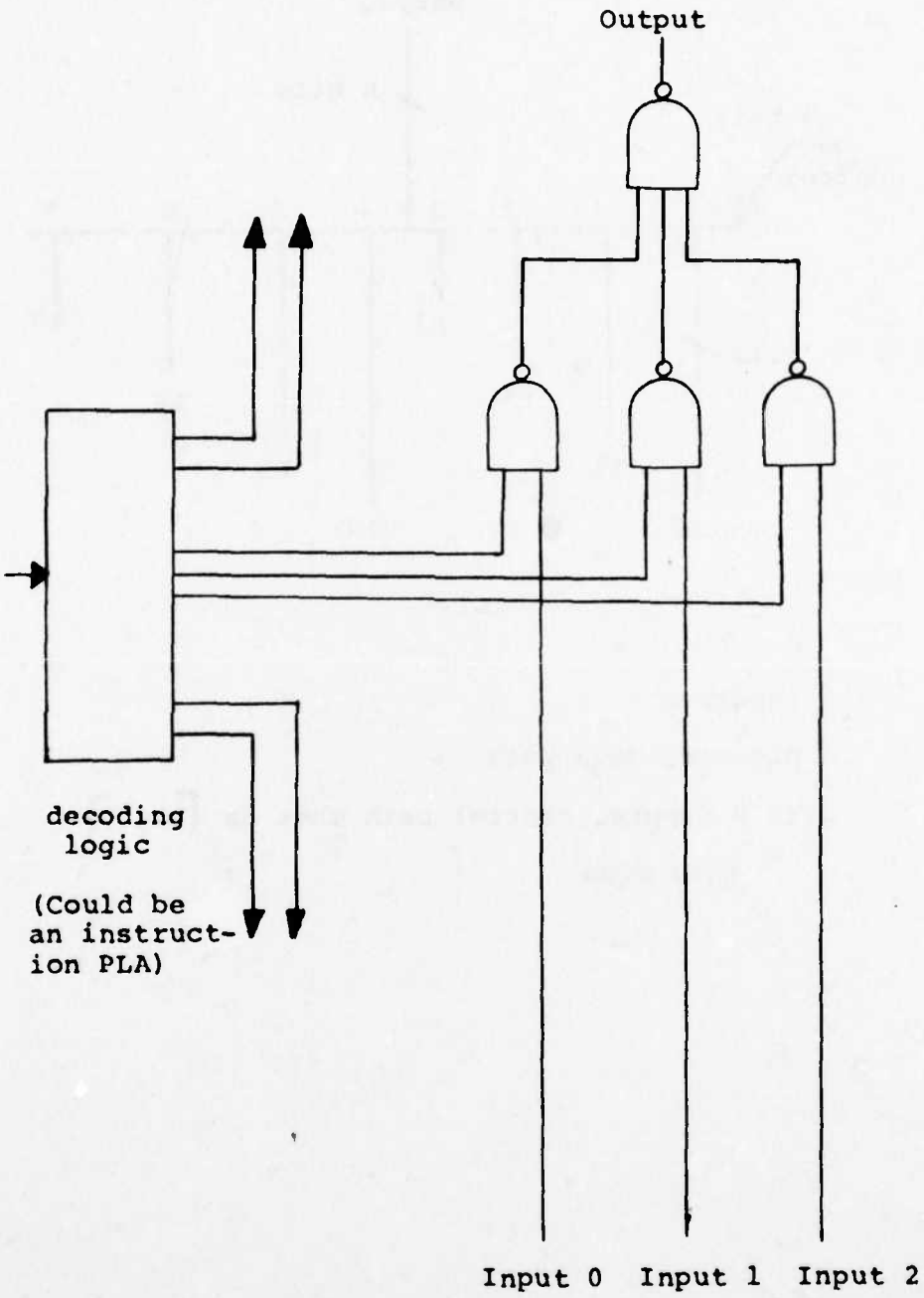


Figure A.8. A typical implementation of a data selector

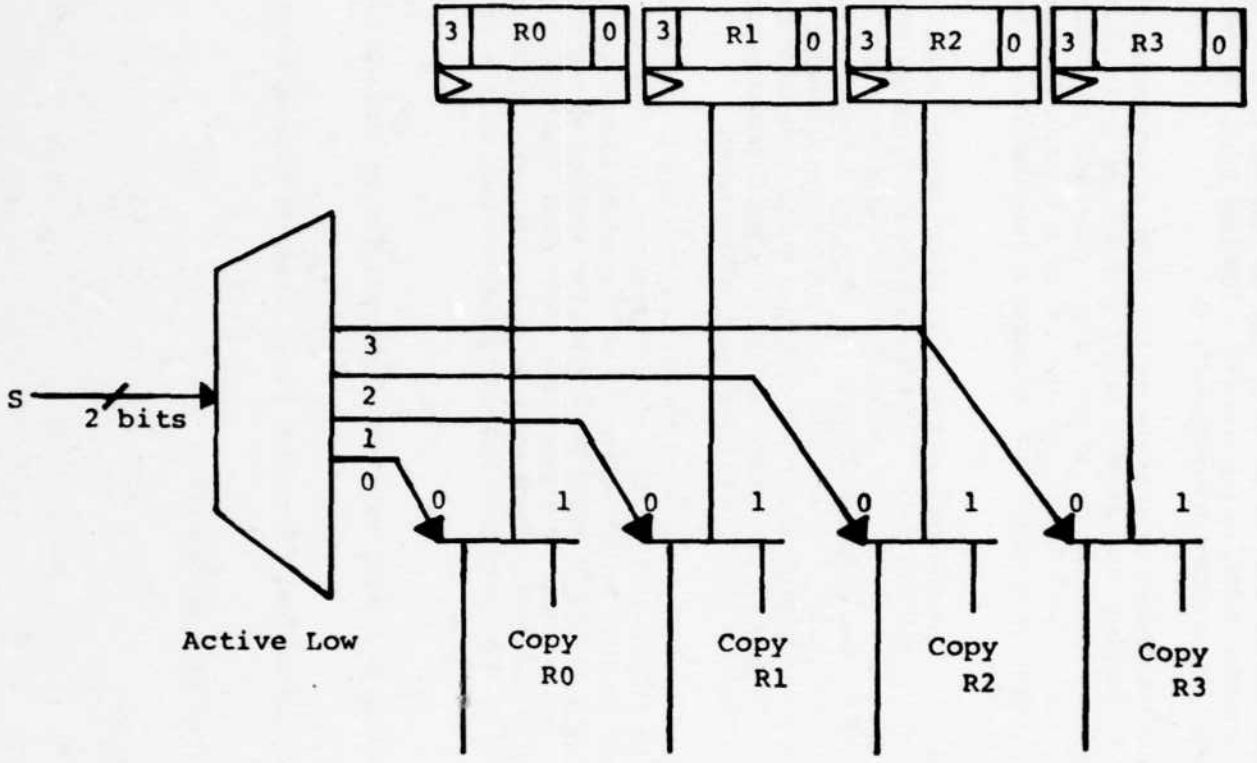


Figure A.9. Demultiplexer (acting as address selector)

Function blocks, such as an ALU, may be denoted as a rectangle with a boolean equation, function table, or reference to a function table. In rare cases, logic gates may be drawn. This is shown in Figure A.10.

A few miscellaneous notes are necessary. First, asynchronous resets, presets, and clears should be avoided, both as good design practice and as an existing machine description. The only exception is in the latter case and only when the exact implementation is known (such as through the use of a Product Evaluation Report). Ideally, a machine should be designed so that complete initialization is a matter of only one or a few instructions.

Second, the synchronizing signal (the clock) should never be gated or modified. Gating of the clock is one of the easiest ways to encourage race conditions, either in the real device, or in a LASAR model of a DUT where there is in fact no actual race. In most devices, the copy operation is not done as is Figure A.9, but is done instead by stopping the clock. This is a case where it may be justifiable to implement this as a true copy in the LASAR model (and of course in the Automaton Diagram) even when the actual device gates the clock. Again, it is good design practice to avoid techniques which change the characteristics of the feedback paths.

It is not known at the time of this writing how widely Automaton Diagrams will be used. The MIL-M-38510 detail specification for the 2914 will include the Automaton Diagram for that device, and it will be interesting to note the reaction from industry. A great deal of benefit would result from the adoption of any common design language: logic designers need be trained only once; good design practices would be encouraged and easily monitored; a consistent set of logic blocks would evolve; and most pertinent to this effort, test generation and fault isolation would become almost trivial.

References:

1. Zvi Kohavi, Switching and Finite Automata Theory, McGraw-Hill, 1978, pp. 356-375.

2. Microprocessor Test Generation Using Automata Models, RADC-TR-?-?, In preparation.

3. Kohavi, op. cit., pp. 283-305.

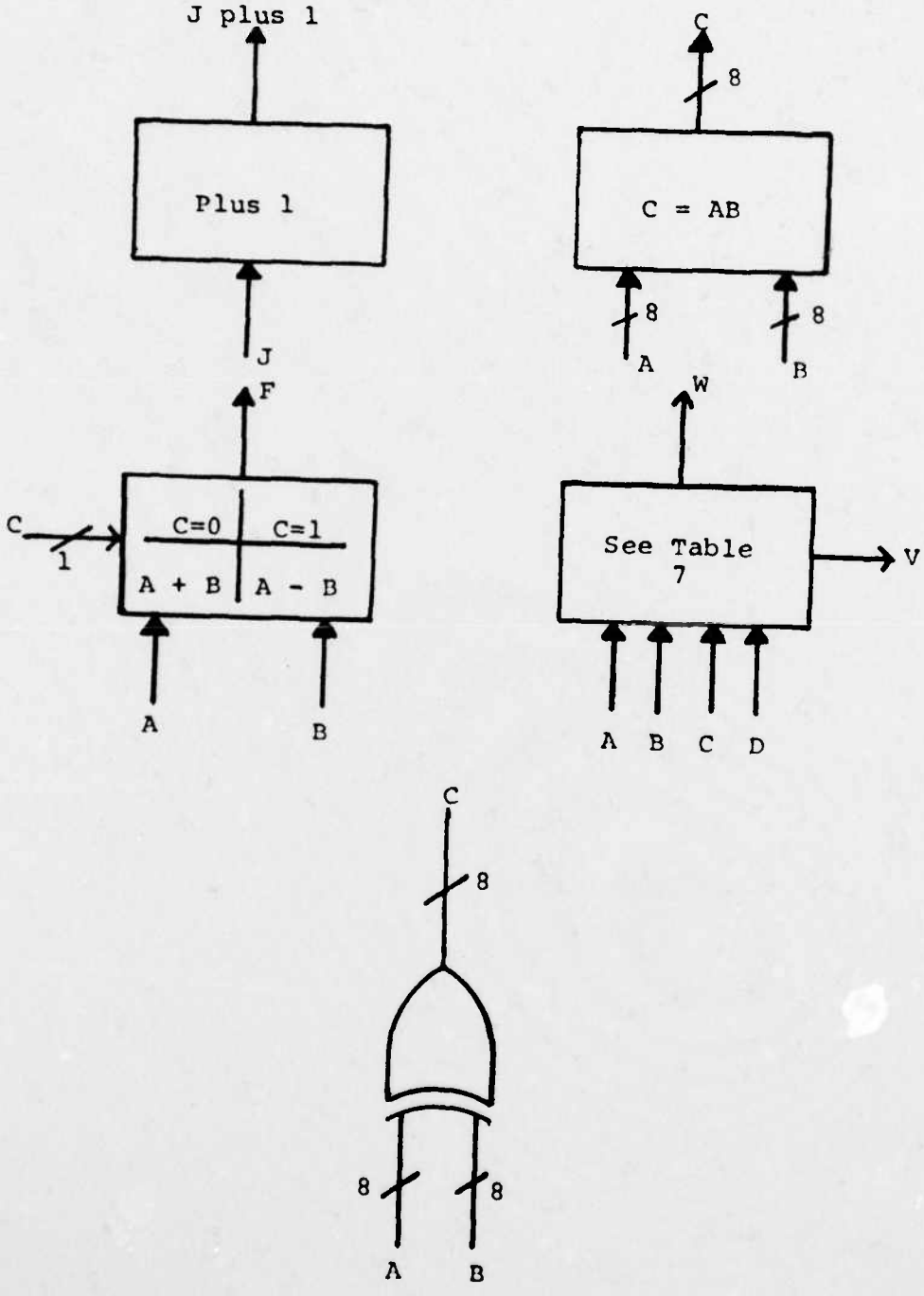
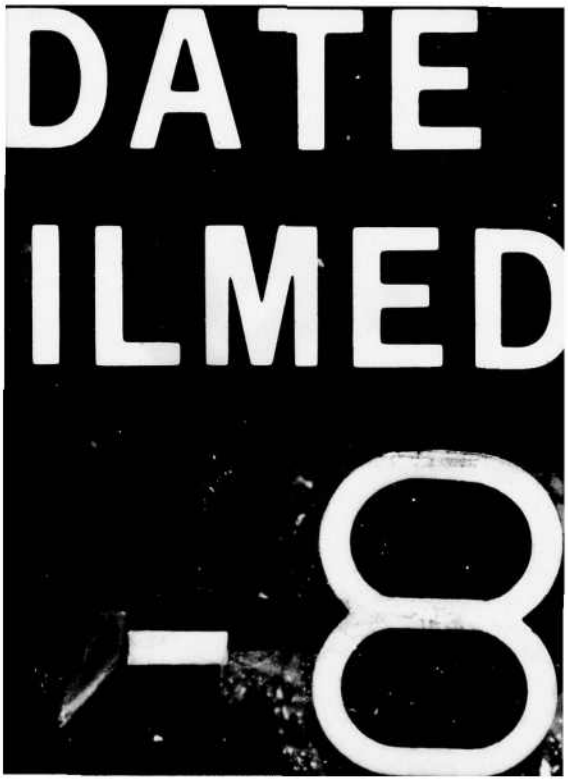


Figure A.10. Function blocks

☆U.S. GOVERNMENT PRINTING OFFICE: 1981-714-025/90

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


DATE
ILMED
-8