6-30-1989

Optimal algorithmic testing of re-programmable logic arrays

Arvind Chopra
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

OPTIMAL ALGORITHMIC TESTING OF RE-PROGRAMMABLE LOGIC ARRAYS

(May 1989)

Arvind Chopra, M.S.E.E., New Jersey Institute of Technology

Thesis Advisor: Dr. William N. Carr

This research delineates a test strategy and detailed sequence of test vectors for a new class of programmable logic devices (PLDs). This new class of devices is based on architectures that contain programmable logic arrays (PLAs) programmed by on chip integrated static RAMs or EEPROMS. The new PLD architecture poses unique testing problems which require 100% nodal test coverage. Previous PLAs/PLDs did not require that 100% of the nodes be tested since their circuits utilized fuse link crosspoints and were not dynamically reprogrammable. Such devices could be tested by a unique personality pattern that combined the application and the test circuit. The new PLA/PLD architecture requires testing of arbitrary personality patterns. The dynamic reprogrammability of these devices presents a more challenging 100% test requirement. The same PLA/PLD device can be reconfigured for many different personality patterns permitting a 100% internal node coverage using externally applied test stimulus-response sequences at the packaged device's pins.

This work details the testing of a specific PLA/PLD circuit but attempts to provide a scheme of general applicability. All stuck at, bridge and crosspoint faults are tested. The unified testing scheme minimizes the costs for both the production and application environments.

OPTIMAL ALGORITHMIC TESTING OF RE-PROGRAMMABLE LOGIC ARRAYS

by
Arvind Chopra

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1989

APROVAL SHEET

Title of Thesis: Optimal Algorithmic Testing of ReProgrammable Logic Arrays.

Name of Candidate: Arvind Chopra
Master of Science in Electrical Engineering, 1989

Thesis and Abstract Approved:

Dr. William N. Carr
Professor
Electrical Engineering — Date: June 22, 1989

Dr. Durga Misra
Assistant Professor
Electrical Engineering — Date: June 23, 1989

Dr. N. M. Ravindra
Assistant Professor
Electrical Engineering — Date: 6. 23. 89

VITA

Name: Arvind Chopra

Present address:

Degree and date to be conferred: M.S.E.E., 1989.

Date of birth:

Place of birth:

Secondary education: DAV College, Chandigarh 1982.

Collegiate institutions attended	Dates	Degree	Date of Degree
Panjab University	8/81-5/85	B.E.(Hons.)	July,1986
New Jersey Institute of Technology	01/88-5/89	M.S.E.E.	Oct,1989

Major: Electrical Engineering.

Positions held:

Assistant Engineer (R&D), Hindustan Computers Ltd., New Delhi, India.

Research Assistant; Microelectronics Center, NJIT, Newark, NJ.

Teaching Assistant; Physics Dept., NJIT, Newark, NJ.

ACKNOWLEDGEMENT

I wish to express my grateful indebtness to Prof. William N. Carr at the EE Department, NJIT for his continuous guidance and encouragement throughout the course of this thesis work. I am beholden to Prof. Durga Misra and Prof. N. M. Ravindra for sparing their valuable time to go over and help in the consolidation of this work. My appreciation is due to my colleagues Jayesh, Praveen, Elie, Alexis and Ravi Prakash, for their suggestions, help and for creating the stimulating atmosphere we all worked in. I am also thankful to Dr. Zheng Teng and Ruyan Wang for making available the Microelectronics Laboratory resources. I am eternally grateful to my mentor and grandfather Prof. R. S. Chopra and my parents for their guidance and support. Finally my special thanks to Ines for her constant encouragement and untiring support.

Arvind Chopra

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last few years have seen an increasing use of programmable logic devices such as PROMs, PALs and PLAs. PLAs are being increasingly used to replace random glue logic in large circuit designs. The PLAs offer a smaller turn around time than full custom design and yet a proprietary circuit customization. The use of PLAs poses unique testing problems. The PLAs are usually one time programmable and hence cannot be checked for production faults by the conventional methods . The production tests must completely check the chip for all possible faults. The one time programmability of the PLAs limits the section of the circuitry that can be checked from the external pins. Therefore production testing is done before packaging by wafer probing which requires special equipment and is time consuming. Furthermore after packaging the chip cannot be completely tested and the user must design some test scheme and circuitry to be implemented along with his actual circuit.

A new approach to the design and testing of PLAs is to employ a static RAM instead of one time fusable metal links to enable and disable the crosspoint links in the Input and Output arrays. The reprogrammability allows new circuit implementations at will but also neccessiates a complete diagnostic check to be per-

formed, preferably continuously but at least periodically, say at power on. The reprogrammability would make the chip completely fault testable from the external pins and would also obviate the need to program redundant test circuitry along with the actual circuit.

The NJIT PLD is a SRAM programmable array of four PLA Cells and is used as a model for test derivation. The design and layout of the initial SRAM cell has been done by RAVI VADIYANATHAN in his master's thesis *CMOS SRAM Design for initializing Programmable Logic Devices.* The design and layout of the PLD chip has been implemented by PRAVEEN K. PARVATHALA for his master's thesis *A Dynamic CMOS Programmable Device.* This thesis highlights the design for testability aspects of the chip and its testing scheme in production runs. The production tests may also be executed by a user to check the correct functioning of the chip.

Chapter 2 describes the NJIT PLD as it appears to a user. This chapter describes the various blocks in the chip. It also provides other necessary application information such as the different Pin functions, signal timings, the programmer's model and the the SRAM writing/reading for circuit implementations.

Chapter 3 describes the testing approach to the NJIT PLD. It explains the testing of all the sections except PLA Cells on the chip.

Chapter 4 describes the faults that arise in PLAs. It then discusses the detailed testing of the PLA cells. It details the various test patterns and vectors for detection of different faults in the PLAs.

The test equipment used for actual testing of the NJIT PLD is INTEGRATED MEASUREMENT SYSTEM's HS1000 test station which consists of an IMS1000 Main-

frame and VS1000 Verification System. Appendix A gives an overview of the equipment and its use.

Chapter 2

The NJIT PLD

2.1 The PLD Alternatives

Today a logic designer has a plethora of alternatives available for his circuit implementations. These range from standard products to fully customized integration. In between these two extremes are the semi-custom products that include gate arrays, programmable logic devices (PLDs) and standard cell libraries. The major advantages of the PLDs over other types are that they offer a significant chip count reduction from the standard circuit design and have a faster design and development cycle compared with fully customized integration.

The PLD category includes a number of competing alternatives, all based on a programmable AND-OR plane architecture. Bipolar PLDs are programmed by burning or opening fuse links. CMOS PLAs can be one-time programmable, electrically programmable (EPLDs), or electrically erasable (EEPLDs). These PLDs are often used in place of five to ten SSI/MSI devices.

Programmable Read-Only Memories (PROMs), Programmable Array Logic (PALs) devices and Programmable Logic Arrays (PLAs) constitute the three most popu-

lar forms of Programmable Logic Devices today. All these PLDs have the same basic two level AND-OR architecture, but vary in their logic features and programmability aspects. The basic AND-OR structure of PLDs makes them ideal for implementing logic equations in Boolean Sum-of-Product form.

PROM

The basic logic structure of the PROM consists of a fixed AND array whose outputs feed a programmable OR array (Fig 2.1a). The PROM inputs are fully decoded by the fixed-AND array. This means that every combination of inputs is represented by a separate AND gate. Since the OR array is programmable, the outputs can be programmed individually corresponding to every possible input combination. A limitation of the PROM is that it is difficult to accommodate a large number of inputs. Each additional input doubles the size of the matrix.

PAL

The basic logic structure of the PAL consists of a programmable AND array whose outputs feed a fixed OR array (Fig 2.1b). Unlike the PROM ,the AND array of a PAL device is programmable therefore all the inputs need not be fully decoded. PAL devices may contain additional architectural features such as programmable I/O pins, registered or combinatorial outputs with internal feedback etc. that make them well suited for implementing logic functions.

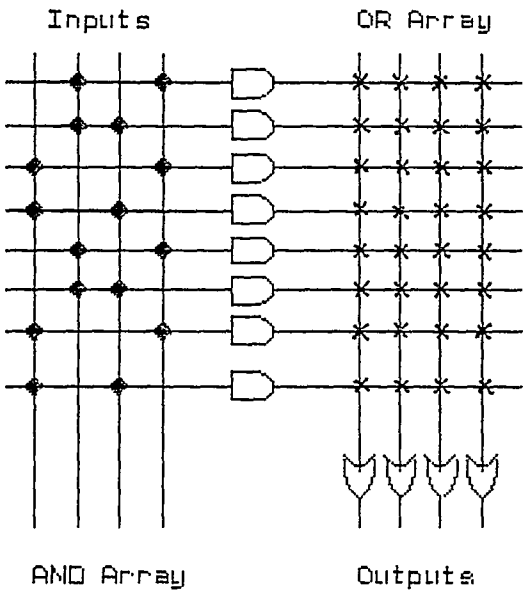
PLA

The basic logic structure of the PLA consists of a programmable AND array and a programmable OR array (Fig 2.1c). The designer has complete control over all inputs and outputs. PLAs allow a great flexibility for implementing logic functions. A limitation of PLAs is that due to the extra programmable-OR plane, a given signal has to pass through two programmable arrays ; as a result, PLAs are inherently slower than PAL devices and PROMs.

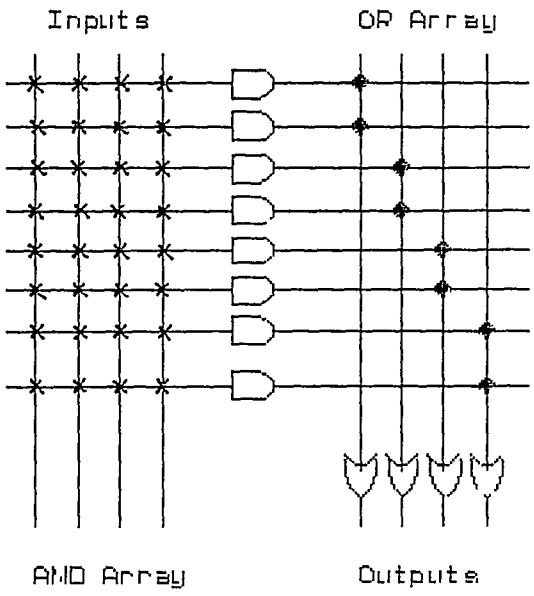
2.2 NJIT PLD Overview

The NJIT PLD is an advanced LSI design. It consists of four PLA cells linked by an 8-bit internal data bus. Figure 2.2 shows the basic block diagram. Each of the four PLA cells is individually programmable. There is an I/O block that interfaces the internal bus to an external 8 bit data bus. Each of the four PLA cells has an address to be selectively loaded or read via the data bus. This architecture allows for concurrent operations in the four cells. The PLD is usually connected to a microprocessor or another intelligent unit that issues appropriate commands and data.

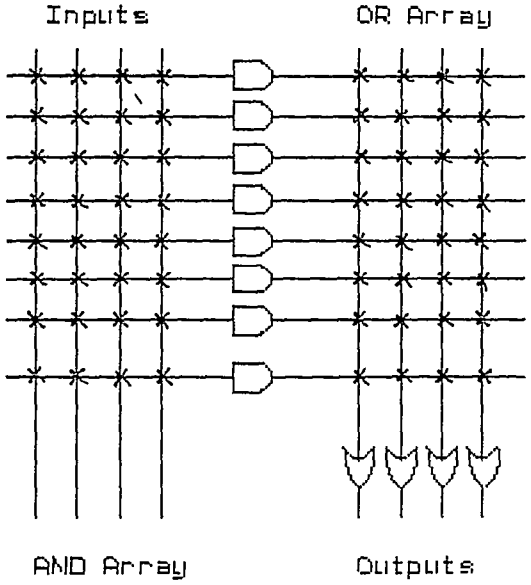
To allow *dynamic re-programmability* each of the PLA cell derives its personality pattern from a SRAM embedded in the background. This SRAM is arranged as 2960 bit (74row X 40column) matrix. Each bit has one to one correspondence with a node in the AND or the OR matrix in the PLA cells. Thus by reprogramming the SRAM the function of the PLA cells may be changed even while powered on. The SRAM is programmed with a serial bit stream and may also be serially readout anytime. There is a daisy chain feature that allows sequential programming of



(a) PROM



(b) PAL



(c) PLA

* Indicates Fixed Connection

X Indicates Programmable Connection

Figure 2.1 Different forms of Programmable Logic Devices

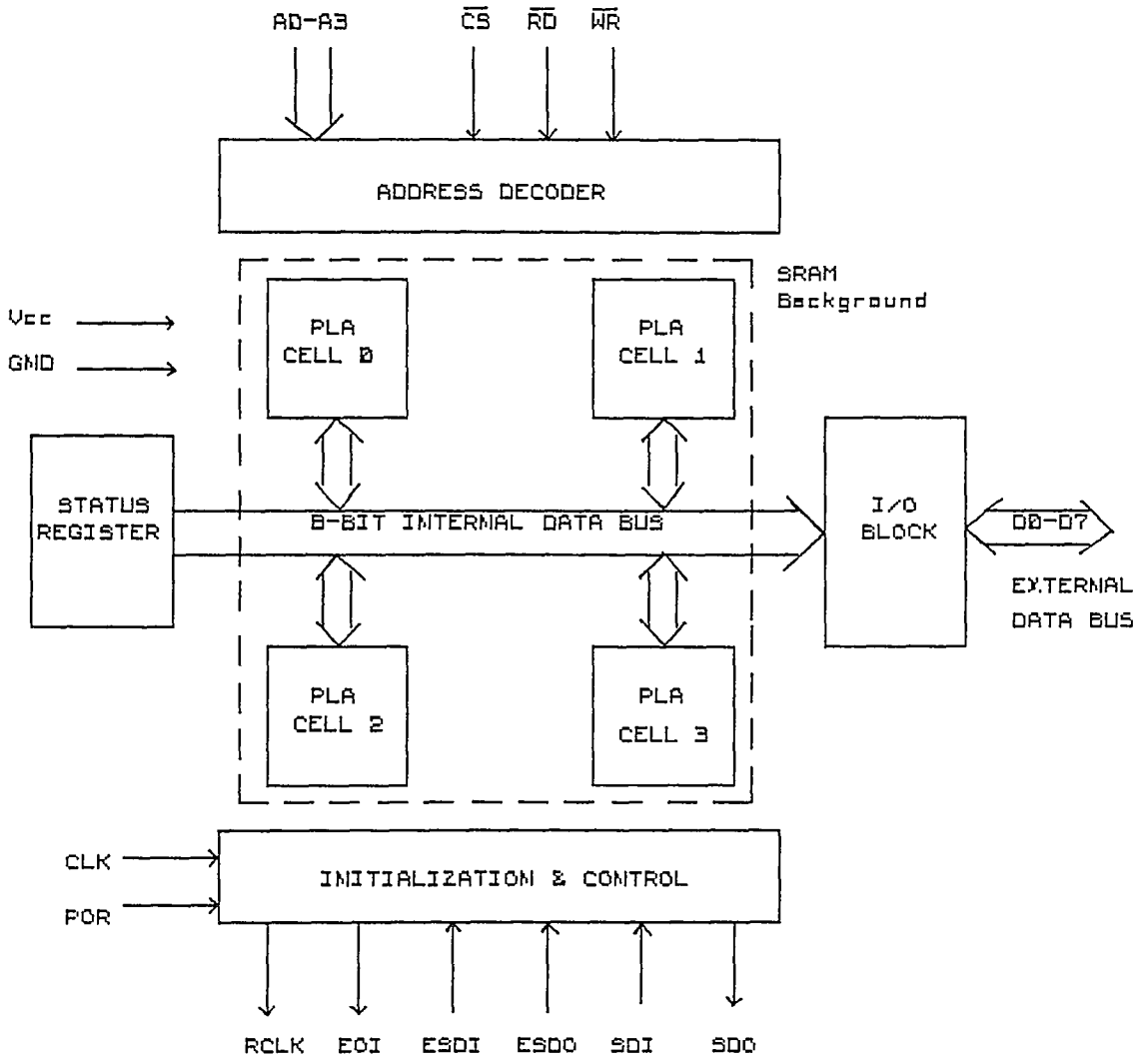


Figure 2.2 NJIT PLD Block Diagram

multiple PLDs from a serial source such as the XLINK

Other features of the chip include a status register. This five bit register indicates whether the SRAM has been properly (completely) initialized and the status of each PLA cell. The status register can be read at any time.

Each PLA cell has four feed back flip-flops and hence can be configured as a 16 state machine. There is a feature that enables each cell to signify end of its processing sequence. The CPU can configure the cells as state machines and also implement a *state detection* circuitry that would indicate the end of processing or any other function. This *End of Operation* signal can be used to latch the out put of the PLA cell and also generate an interrupt and set a status bit in the status register.

This is a distinctive feature compared to other contemporary designs. The NJIT PLD is not intended for use as a replacement of glue logic but for processing intensive hardwares for image analysis, signal processing etc. This justifies the multiplexing of the four PLA cells' inputs and outputs on the common data bus. Appendix A contains greater details about the NJIT PLD circuitry.

2.3 NJIT PLD Specifications

The pin out diagram of the NJIT PLD is shown in figure 2.3. Table 2.1 describes the various pins and their functions.

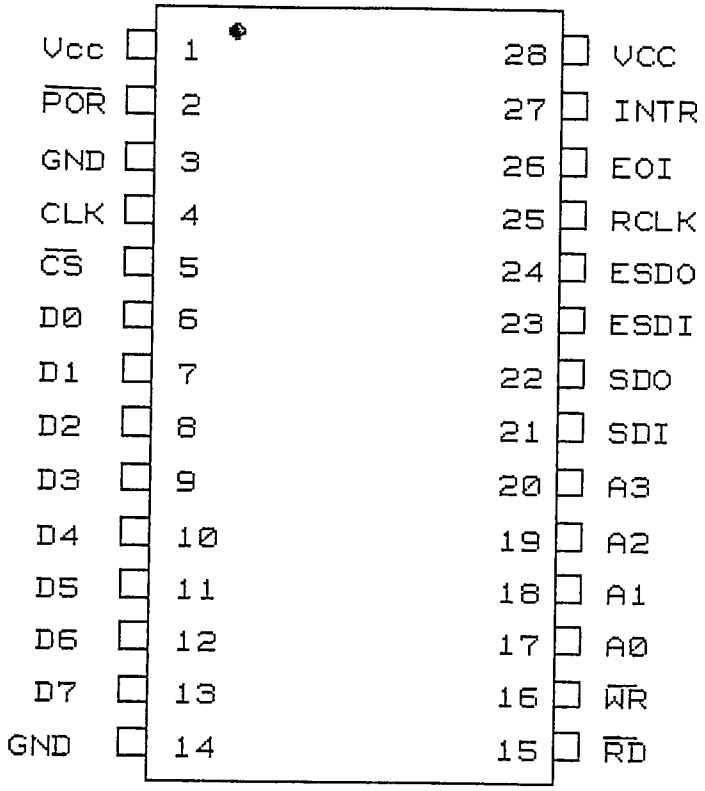


Figure 2.3 NJIT PLD Pin Out Diagram.

PIN	I/O	FUNCTION
V_{CC}	-	+5V Power Supply.
GND	-	Power Supply and signal ground.
A0-A3	I	4 bit address. The address is used to select different PLA cells and the status register.

Address	Block Selected
0	Cell 0 I/P (CIS0)
1	Cell 0 O/P (COS0)
2	Cell 1 I/P (CIS1)
3	Cell 1 O/P (COS1)
4	Cell 2 I/P (CIS2)
5	Cell 2 0/P (COS2)
6	Cell 3 I/P (CIS3)
7	Cell 3 O/P (COS3)
8	Nibble Mode 1 (NM1)
9	Nibble Mode 2 (NM2)
10	Status Register (SR)
11	PLA 0 O/P Latch Enable (EOP0)
12	PLA 1 O/P Latch Enable (EOP1)
13	PLA 2 O/P Latch Enable (EOP2)
14	PLA 3 O/P Latch Enable (EOP3)
15	Not Used

D0-D7	I/O	Bidirectional 8-bit data bus.

Table 2.1: NJIT PLD Pin Functions.

PIN	I/O	FUNCTION
$\overline{CS}$	I	Chip Select. A low on this pin enables the PLD to be read or written. The SRAM can be serially initialized irrespective of the status of this pin. The state transitions of the PLA cells are not affected by this signal.
CLK	I	Master Clock. All internal clocks are derived from this master clock. A clock signal meeting the clock specifications and must always be applied to this pin. Under no circumstances should it be gated off.
$\overline{POR}$	I	Power on reset. This active low signal is used to reset the chip at power up (and at other times as well). It results in a request of reprogramming of the SRAM. Power on reset 1. Resets the SRAM initialization circuitry such that it is ready to be reprogrammed at bit location 0. 2. Resets EOI to low state. 3. Resets all the bits in the status register. 4. Resets all internal flip-flops (PLA cell input, output and feedback latches).
$\overline{RD}$	I	Read. This active low signal together with $\overline{CS}$ and $\overline{WR}$ control the data transfer. When $\overline{CS}$ is low, a low $\overline{RD}$ would read out the data from the source whose address appears on the address lines.
$\overline{WR}$	I	Read. This active low signal together with $\overline{CS}$ and $\overline{RD}$ control the data transfer. When $\overline{CS}$ is low, a low $\overline{WR}$ would write in the data from the source whose address appears on the address lines.

Table 2.1 NJIT PLD Pin functions.

PIN	I/O	FUNCTION
SDI	I	Serial data input. When ESDI is high and EOI low the serial data on this pin is written to the SRAM. The 2960 bit serial data stream provided by an external device is latched at the rising edge of the RCLK.
SDO	O	Serial data output. When ESDO and EOI are high the SDO pin serially outputs the contents of the SRAM. Each positive edge of the RCLK latched the next bit on this pin.
ESDI	I	Enable serial data input. This active high signal sets up the SRAM for serial programming. If EOI is high (ie SRAM has been initialized since last reset) the ESDI pin status is ignored. It is then necessary to reset the chip again to reprogram the SRAM.
ESDO	I	Enable serial data output. This active high signal allows a serial read out of the SRAM when EOI is high. Serial data bits are shifted out at the rising edge of the RCLK.
RCLK	O	Request Clock. It has two fold purpose. When ESDI is high and EOI low it acts as a request clock to an external device which supplies serial data for SRAM programming. The data on the SDI pin is latched at the rising edge of RCLK. When ESDO and EOI are both high the RCLK latches out the serial data from the SRAM on the SDO pin at every rising edge.
EOI	O	End of initialization. This signal is set high when all the 2960 bit nodes of the SRAM have been programmed after a reset. Resetting the chip sets EOI low.
INTR	O	Interrupt. This signal is set high when a of the processing sequence in any PLA cell is completed. The processor must then read the output of a predetermined cell or check the status register to determine which PLA cell has finished its processing. A read from the appropriate cell or the status register will reset this line.

Table 2.1 NJIT PLD Pin functions.

2.4 Timing Waveforms

This section describes the basic timing sequences for the read, write power on reset, serial data write and read operations. The PLD connects as an asynchronous device to the μP bus.

Read Cycle

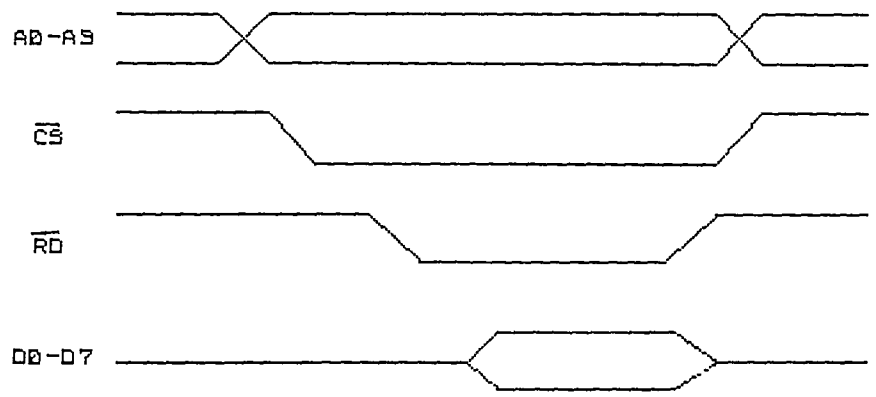
The microprocessor starts the read cycle by driving the address on the address lines, fig 2.4a. The address is decoded externally to generate the $\overline{CS}$ signal. A low $\overline{CS}$ sets up the PLD for an I/O operation. When $\overline{RD}$ goes low the PLD outputs 8 bit data, from the address specified on A0-A3, on the data bus. The address must remain stable and $\overline{CS}$ active while the data is being read.

Write Cycle

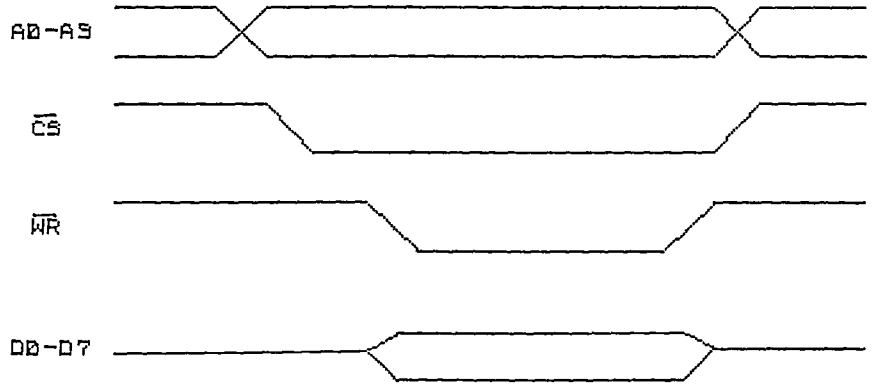
The microprocessor starts the write cycle by driving the address on the address lines, fig 2.4b. The address is decoded externally to generate the $\overline{CS}$ signal. A low $\overline{CS}$ sets up the PLD for an I/O operation. The processor outputs the data on the data bus and sets $\overline{WR}$ low. The PLD latches this 8 bit data, at the address specified on A0-A3, on the data bus. The address must remain stable and $\overline{CS}$ active while the data is being written.

Power On Reset

At power on the $\overline{POR}$ must be held low for a duration of at least 10ms to properly initialize the chip, figure 2.5a. The reset signal may be applied at any other time



(a) Read Cycle



(b) Write Cycle

Figure 2.4 NJIT PLD Read and Write Cycles

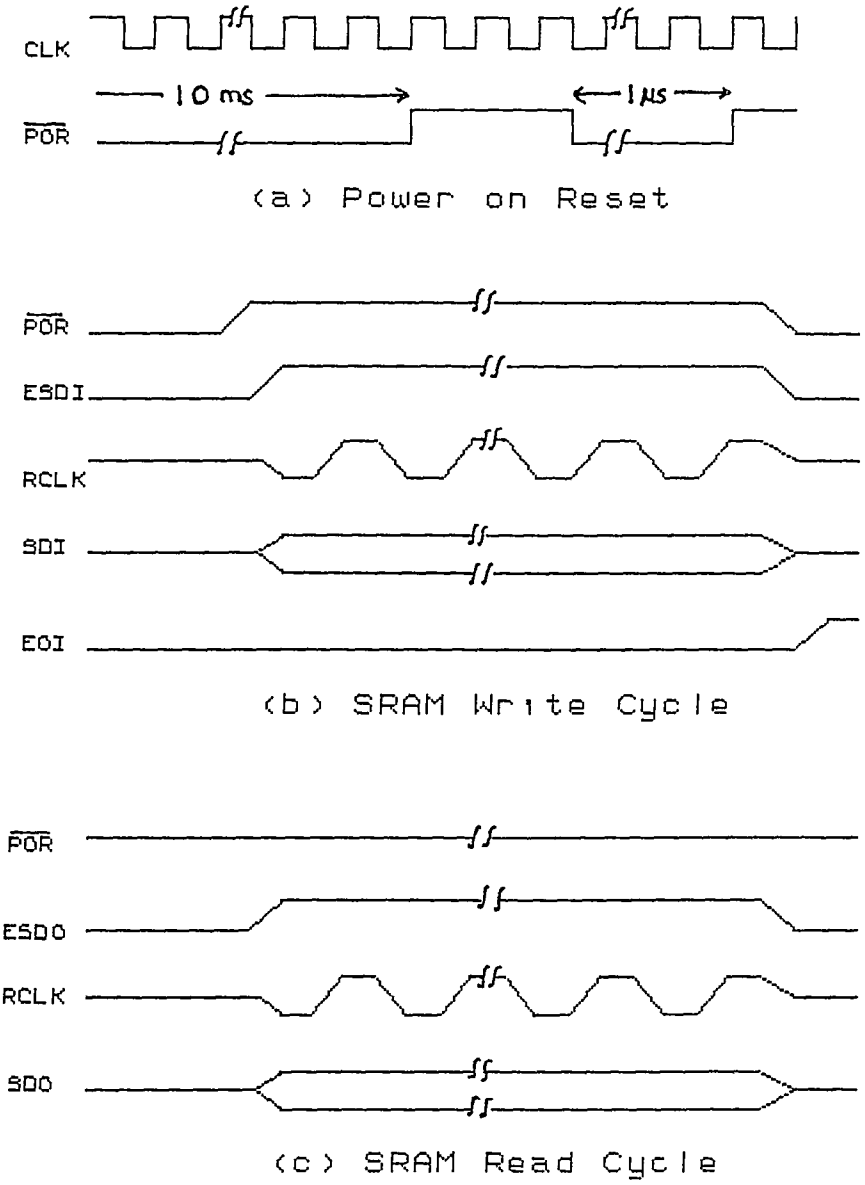


Figure 2.5 Power on Reset, SRAM read and write cycles

as well and its minimum duration must be $1\mu s$. After a reset the SRAM has to be reprogrammed.

Serial Data Write

After a reset the EOI pin is set low and the SRAM has to be programmed serially. When ESDI pin is driven high the PLD generates a request clock, RCLK to serially shift in the data on the SDI pin, figure 2.5b. After the 2960 bits of the SRAM have been programmed the EOI signal goes high and RCLK is tristated. ESDI must be held high until EOI goes low. The RCLK frequency is 1/16 of the CLK frequency. The initialization time is $16 * \frac{1}{f_{CLK}} * 2960 = \frac{48360}{f_{CLK}}\ secs.$

By connecting the EOI pin to the ESDI pin of another PLD multiple PLDs may be programmed sequentially from a single serial data source.

Serial Data Read

The SRAM contents can be read out at any time by pulling the ESDO pin high. Serial bit stream will be shifted out on the SDO pin at each positive transition of the RCLK, see figure 2.5c. It is not necessary to read out all the 2960 bits and ESDI may be driven low before all bits have been read. However when ESDO is driven high again the serial data from the next location in the SRAM will be shifted out unless the chip has been reset.

2.5 Programming Model

Figure 2.6 shows the programming model of the NJIT PLD. There are four input and four output registers, one pair for each PLA cell.

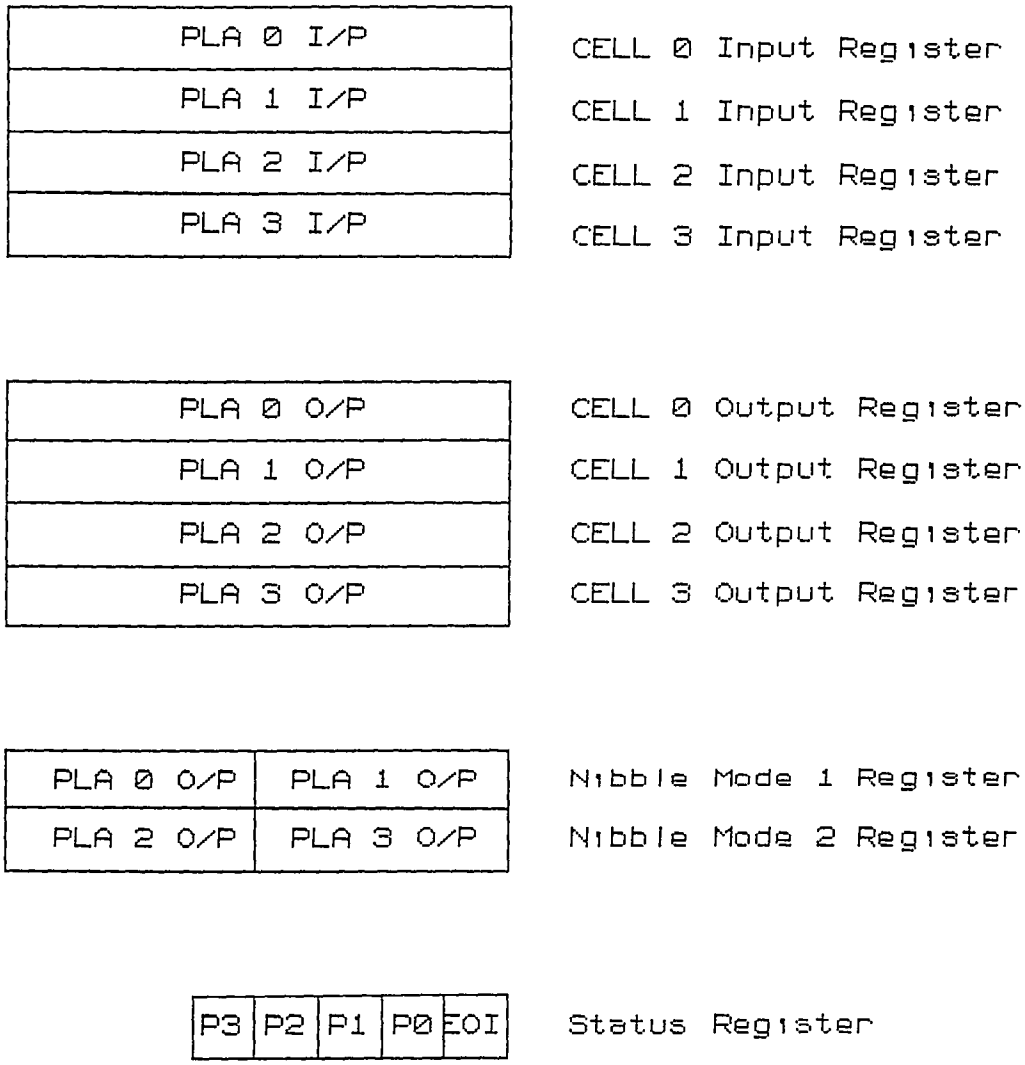


Figure 2.6 Programmer's model of the NJIT PLD

There is a 5 bit status register. Bit 0 is set at the end of SRAM initialization. Bits P0-P3 are set if the corresponding PLA cell has finished processing. At this time the result is available in the corresponding PLA O/P register.

Each PLA cell has an extra OR term that is used to detect the end of processing, latch the result at the PLA cell O/P register and set the corresponding bit in the status register.

There is a provision for four bit data modes referred to as nibble modes. Each nibble mode register allows simultaneous reading of 4 bits from two PLA cell O/P registers. Nibble mode 1 (NM1) reads the upper four bits from the PLA cell 0 O/P register and lower 4 bits from the PLA cell 1 O/P register. Nibble mode 2 allows reading of the upper four bits from PLA cell 2 O/P register and lower four bits from PLA cell 3 O/P register.

Chapter 3

Testing of NJIT PLD

3.1 Test Approach to the NJIT PLD

The NJIT PLD is a unique chip and so is its testing. Like any other PLA the NJIT PLD must pass through exhaustive production testing where in every fault is checked for. When in use the chip has the unique feature of *reprogramming.* The *personality* of the chip or in other words the circuit realization in each of the four PLA cells in the NJIT PLD may be changed at any time. This poses a unique test problem when testing the PLA cells in the NJIT PLD. The solution to this problem is equally unique and the reprogrammability of the chip is utilized to create a unified production and user test program for complete testing of the PLA cell.

The complete testing proceeds in a sequence of tests starting at power on. The chip is viewed as if partitioned into blocks and these blocks are tested either independently or together in groups. The following sections provide detailed discussion on the actual testing of the NJIT PLD in a packaged form.

3.2 Test Equipment for the NJIT PLD Testing

As noted earlier the NJIT PLD is tested in a completely packaged form by applying signals externally through the pins and observing the output responses. The actual application of test vectors and sensing of the outputs is done on the INTEGRATED MEASUREMENTS SYSTEMS (IMS) HS1 workstation. This comprises of a IMS 1000 LOGIC MASTER system and a VS1000 VERIFICATION STATION. The IMS 1000 connects to a computer or a terminal from which test programs can be run. The device under test is wired on a test board on the VS 1000 verification station. The Logic Master acts as a controller for the VS1000 and sends test data, clocks, power supply and acquires input data which may then be compared with reference response. More details about the IMS test system are presented in Appendix B.

3.3 Overview of NJIT PLD Testing

As discussed above the NJIT PLD has a unified production and user test set. This section describes the overall test scheme for the chip.

Referring to figure 2.2 the blocks of the NJIT PLD are:

1. 2960 bit serial read/write static RAM for *programming* a circuit in each of the four PLA cells.

2. Four PLA cells each with 8 inputs, 8 outputs, 4 feedback terms (through D type flip-flops). Each cell has twenty 20 input *product* terms and nine 20 input *sum*[1] terms.

[1]The actual PLA implementation is NOR-NOR for speed considerations. Here the standard *product* and *sum* term terminology is used to refer to the input NOR and output NOR terms.

3. Status register with five bits. Four of these bits indicate end of processing in the corresponding PLA cells. Bit 0 is set to indicate *end of initialization*, that is to signal completion of the SRAM programming.

4. Input-Output buffers to route data to/from any of the PLA cell input, output or the status register to the external data bus.

5. Address decoder to select a PLA cell input or output register for reading or writing and to select the status register for reading. In addition the address decoder can also clock the output registers of each PLA and hence load the PLA cell result in its output register during testing.

6. Timing and control block is responsible for proper powering up of the chip and generation of internal clocks, serial data requests, serial data read and write, interrupts and other status signals such as EOI etc.

The overall chip testing involves checking of each of these blocks individually. Failure in any one block implies an overall test failure. Some blocks can be checked independently of others while some cannot be. When testing for faults in a block that cannot be checked independently the other interacting blocks are assumed to function correctly or a fault in the interacting blocks will result in a faulty response from the block being tested. However each block must be examined for all possible faults either separately or in sections along with other blocks.

Figure 3.1 shows the skeleton of the test scheme for the NJIT PLD. After powering on the initial hardware status is checked. All the inputs are either driven to the inactive state or left in hi-Z state. The status of all the outputs and the hi-Z inputs is sensed and compared.

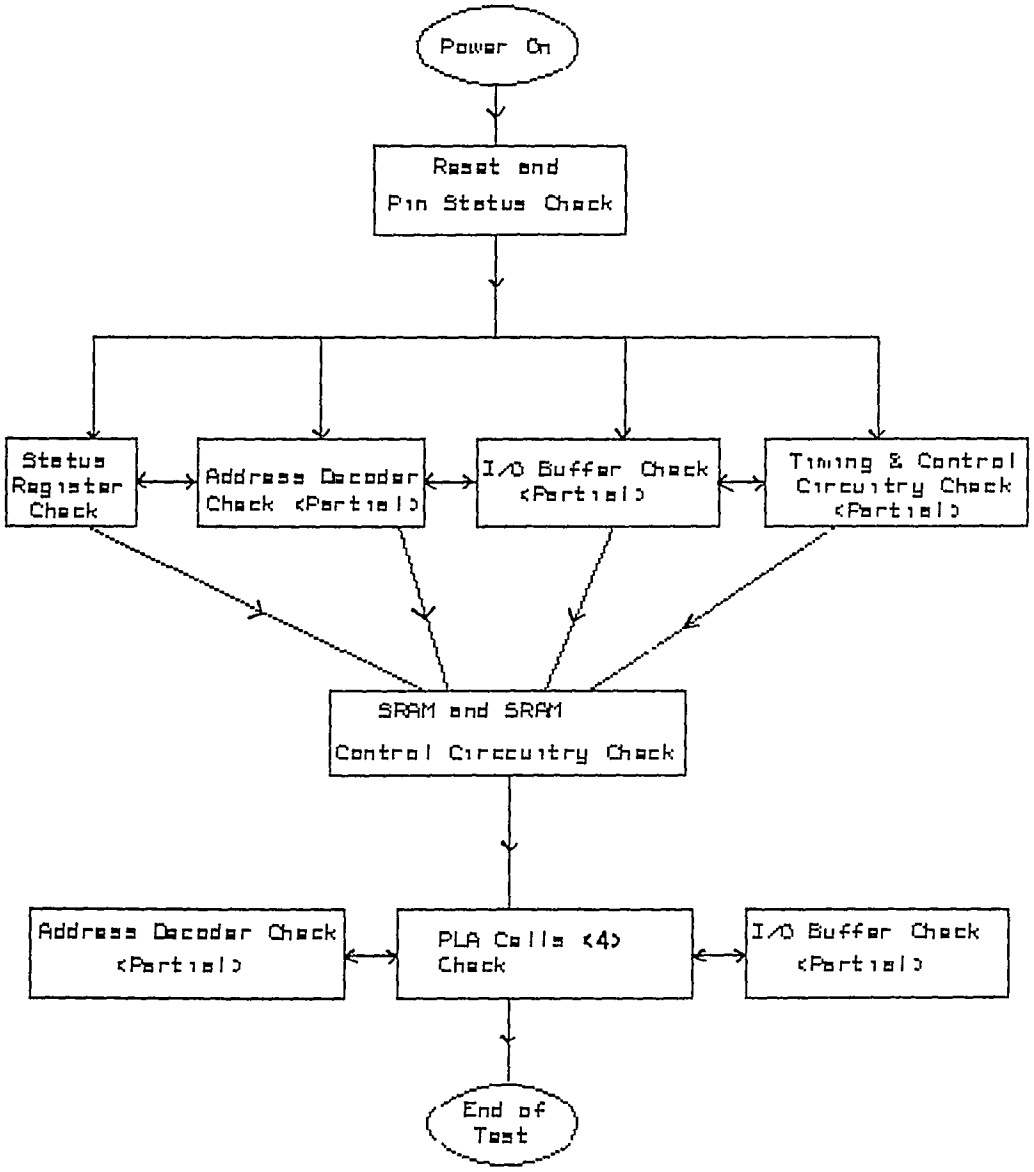


Figure 3.1 Skeleton of the NJIT PLD test scheme.

The status register is read next and its value is compared with expected value. Note that reading of the status register requires correct operation of the address decoder, input-output buffer and control circuitry to a certain extent. Neither of the above interacting sections can be checked just by itself. The probability of having some faults in any one or more from the status register, address decoder, input-output, or control blocks mask a fault in another block is almost zero. In general

> *In any interaction of two or more blocks a fault in any one block will not mask a fault in another block and vice-versa as long as these blocks interact only via signals controlling each other and do not share any data flow.*

Also note that the testing of any one block in such a combination would also check out the parts of other blocks that interact with the block under test. Hence along with the status register the *End of Processing* signal generation, interrupt generation and address decoding for the status register and other blocks used in this test are also checked. All these tests will hereafter be collectively called **Status Generation test.**

The SRAM is checked next. The SRAM is checked after the status generation test because it requires more time. For a go/no-go type of test all faults that can be detected quickly must be checked first. In testing of the NJIT PLD the different checks are performed in an increasing order of their time requirements. Checking of the SRAM also tests the initialization control circuitry and bit 0 (EOI) of the status register.

Finally each of the four PLA cells is checked. Checking of the PLA cells is

the biggest challenge in the test design on which various schemes have appeared in literature. However the NJIT PLD has a unique *reprogrammable personality* and is checked differently from all the other schemes. The NJIT PLD allows a unison of user and production testing. It is unique in the aspect that it allows a *complete fault detection of all the resources from the external pins.*

Each PLA cell is tested individually. Along with the testing of each cell the interacting sections of the address decoder, input-output buffer and the control circuitry is also tested. The testing of the four PLA cells is *concurrent* but *distinct*, that is at any time each of the four PLA cells is in a different state.

More detailed testing of each block is discussed in the following sections.

3.4 Power On Check

The is the first test in the sequence after the chip has been powered on and applied a reset signal. All the input pins are driven inactive and if those pins which do not have any inactive states are left in high impedance or tristate condition. The state of all the output pins and the tristated input pins is noted and compared with the values indicated in the table 3.1.

Pin Name	I/O	Power On Status
$\overline{POR}$	I	High
CLK	I	1MHz clock
$\overline{CS}$	I	High
$\overline{RD}$	I	High
$\overline{WR}$	I	High
$A_0 - A_3$	I	Tristate
$D_0 - D_7$	I/O	Tristate
$ESDI$	I	Low
$ESDO$	I	Low
SDI	I	Tristate
SDO	O	Low
$RCLK$	O	Low
EOI	O	Low
$INTR$	O	Low

Table 3.1 NJIT PLD pin status after power on and reset.

3.5 Status Generation Check

This check detects faults in all the circuitry associated with generation of status. The only exception is the *end of initialization* circuitry which is tested along with the static RAM. The status of the chip is indicated by the following pins and registers:

1. **Status Register:** The five bits in the status register indicate the *end of processing* for each of the four PLA cells and the *end of initialization.*

EOP3	EOP2	EOP1	EOP0	EOI

Each of these bits is set to **1** on occurance of the corresponding condition.

2. **Interrupt (INTR) Pin:** The occurance of end of processing in any of the PLA cells sets the INTR line high. This indicates an interrupt request. The CPU has to poll the status register to determine which of the four PLA cells had actually finished processing. A read from the status register or the output register of the PLA cell that has finished processing will reset this request.

3. **End of Initialization (EOI) Pin:** This signal is driven high by the PLD when the SRAM has been completely loaded with PLA cell configuration data. This signal is tested in the SRAM check.

The parts or sections of the chip that would be tested are:

1. Status Register.

2. Interrupt generation and reset.

3. End of processing generation in each cell.

4. Address decoder for addresses A,B,C,D and E (Hex).

The status generation check proceeds in the following steps: (see figure 3.2)

1. At the outset the status register is read. All the bits in the status register must be zero to confirm that

   (a) Initialization of SRAM is not yet complete.

   (b) No PLA cell has detected end of processing.

   Any deviation indicates a fault.

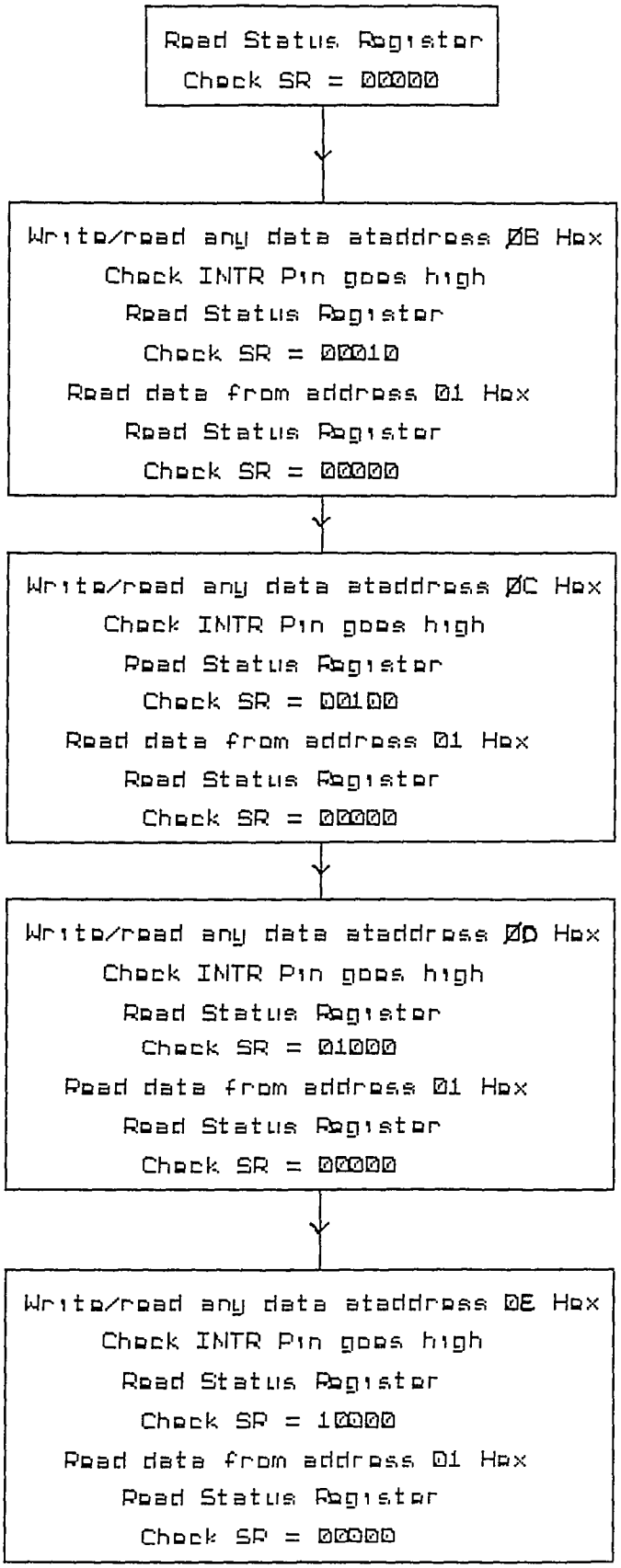


Figure 3.2 Status generation check

2. The second step is the checking of end of processing signal generation in the first PLA cell (Cell0). It proceeds as follows:

   (a) An address of 0B (Hex) is put on the address bus. This location is neither readable nor writable but has been created exclusively for test purposes. Any read or write activations at this address would be ignored by the PLD chip.

   Normally the output of a PLA cell is latched onto a cell output register/buffer. The clock signal for loading the output data of the PLA cell in its register is generated in the cell itself. Each PLA cell has nine output terms out of which only eight are brought on the data bus. The extra output is used to clock the cell output latch. Thus a state detection can be implemented and the cell output latched when the cell reaches a pre-defined state. For test purposes the application of an address 0B-0E (Hex) on the address bus would enable the latching of the corresponding cell's output. The cell output latching is independent of the $\overline{RD}$ and $\overline{WR}$ lines and the data present on the data bus is of no interest. Thus the end of processing status generation circuitry is activated.

   (b) The next step is to check the status on the INTR pin. Latching of the cell 0 output data must also set the INTR request high.

   (c) The status register is polled now to determine which cell had actually reached end of processing. All the bits except the EOP 0 bit in the status register must be zero.

   (d) The Cell0 output register is read. This should reset the INTR pin. The resetting of the INTR pin is now checked.

(e) Finally the status register is read again to verify that bit EOP 0 has been cleared. The status register again reads 00000.

3. The third step is the checking of end of processing signal generation in the second PLA cell (Cell1). It proceeds in a similar manner to the checking of EOP status generation for cell 0 except that address 0C Hex is read/written to set EOP1 and INTR signal and the Cell1 output is read from Cell Output Register at address 03 Hex.

4. The next step is the checking of end of processing signal generation in the third PLA cell (Cell2). It proceeds in a similar manner to the checking of EOP status generation for cell 0 except that address 0D Hex is read/written to set EOP1 and INTR signal and the Cell1 output is read from Cell Output Register at address 05 Hex.

5. The final step is the checking of end of processing signal generation in the fourth PLA cell (Cell3). It proceeds in a similar manner to the checking of EOP status generation for cell 0 except that address 0E Hex is read/written to set EOP1 and INTR signal and the Cell1 output is read from Cell Output Register at address 07 Hex.

This completes the Status Generation checking.

3.6 SRAM Testing

The static RAM is used to implement the desired circuit in the PLA cells. Each PLA cell consists of 740 programmable nodes (links or cross-points) in the input, output and feedback arrays. The total SRAM capacity is $4*740 = 2960$ bit capacity.

To keep address decoding simple and to reduce pin count of the chip the SRAM is read or written serially. This is justifiable since the SRAM needs to be read or written only once for each circuit implementation. The positive transition of the RCLK signal latches the data on the SDI pin onto the next SRAM bit location when ESDI pin is high. When ESDO pin is high the SRAM is set for serial read out and each positive transition of the RCLK signal latches the next bit on the SDO pin.

The RCLK is derived from the CLK input to the PLD by dividing it by 16. Thus the time taken to serially read or write the entire SRAM is

$$T_{SRAM} = 2960 * \frac{16}{f_{CLK}} \quad or$$

$$T_{SRAM} = 47360 * T_{CLK}$$

The time taken to read or write the SRAM is $47ms$ at a clock frequency of $1MHz$ and $4.7ms$ at $10MHz$.

The layout of the SRAM cells is interspersed with the layout of the input, output and feedback arrays. This justifies a straightforward testing of the SRAM rather than designing any specially designed or optimized tests. The SRAM tests include the following:

1. Each bit of the SRAM must be programmable to store values **0** and **1**. This checks for any stuck at **0** or stuck at **1** faults.

2. Each bit location of the SRAM must be capable of being set or reset independently of the neighbouring[2] bits.

[2]The neighbour criterion is determined from the physical layout of the SRAM cells and not necessarily according to the read or write sequence.

3. The SRAM read and write control circuitry must also be checked proper operation and timing relationships. The control circuitry includes the RCLK generator, SRAM address counters and read write buffers.

To perform all the above checks a 1010... pattern is shifted into the SRAM and then read out serially to verify correct programming. Next a complimentary pattern 0101... is shifted into the SRAM and then verified by another read out . Each of these two pattern result in a **checker board** pattern being programmed in all the four PLA cells as shown in figure 3.3. These checker board patterns will check the following sections of the PLD:

1. **SDI and SDO pins:** Any stuck at **0** or stuck at **1** fault on the SDI or SDO pins will result in a fixed output voltage at the pin SDO when the SRAM is read. The s-a-0 or s-a-1 fault on SDI will result in the programming of the SRAM with all **0**s or all **1**s. Hence the SRAM read out will be a fixed voltage.

2. **ESDI and ESDO pins:** The faults detected with the SRAM read and write enable circuitry are detected in three steps:

   (a) A s-a-0 on the ESDI or ESDO will never allow reading or writing of the SRAM. This will result in a fixed voltage level on the SDO pin when attempting to read the SRAM.

   (b) A s-a-1 on the ESDI would result in an unintentional programming of the SRAM immediately after power on. Once all the bits of the SRAM have been written (erroneously) to the reprogramming of the the SRAM is inhibited till a reset. Thus in the current test the checker board pattern will never get written to the SRAM because of some random pattern

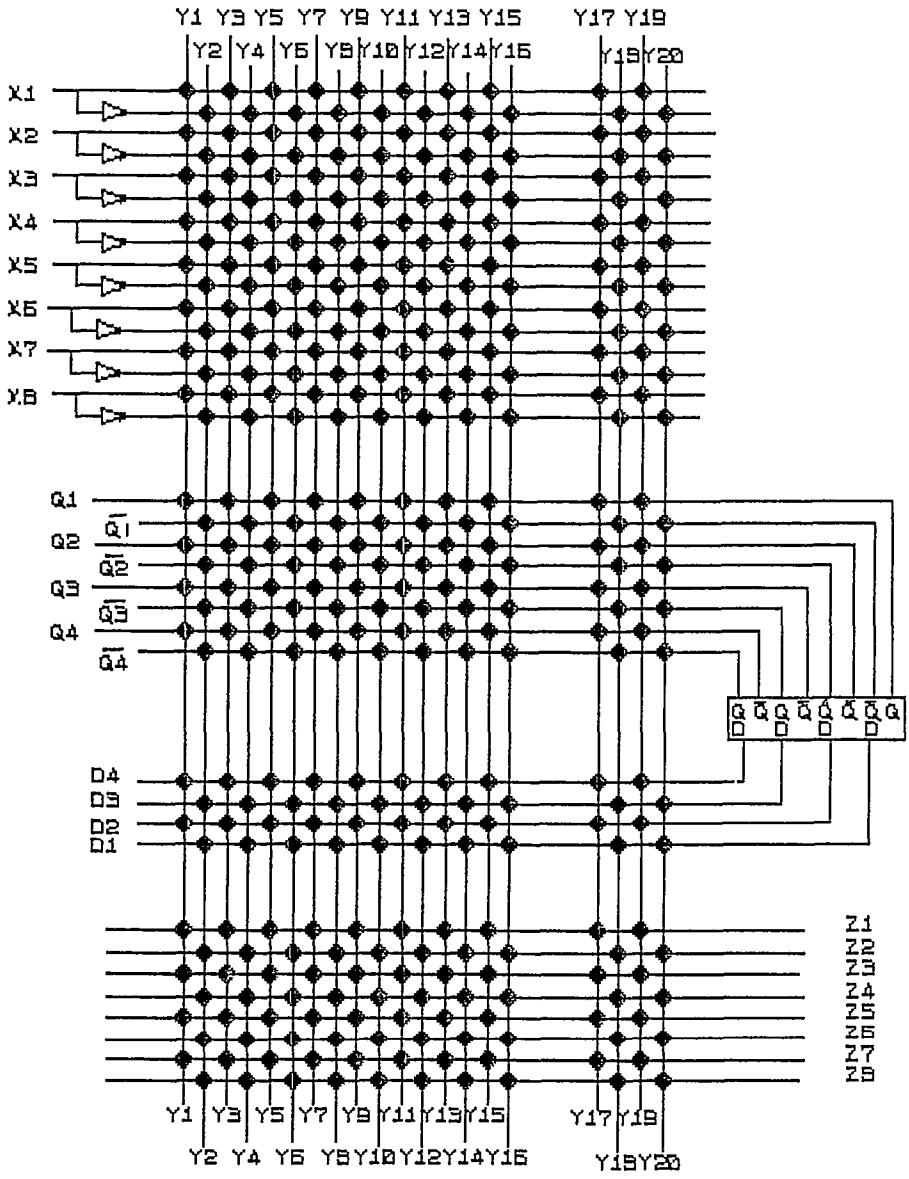


Figure 3.3 PLA Cell Configuration with Checkerboard SRAM Programming.

already being there. Further a s-a-1 on the ESDI will result in appearance of RCLK pulses immediately after power on and EOI pin will go high after all the bits in the SRAM have been programmed. This would be detected by the power on check.

(c) A s-a-1 on the ESDO is detected by the appearance of RCLK signal right after power on and shifting out of the random SRAM data on the SDO pin. This would be detected by the power on check.

3. **RCLK pin:** Any s-a-0 or s-a-1 faults on the RCLK pin would become evident while programming the SRAM with the checker board patterns. Any such fault would inhibit programming and readout. The $\div 16$ relationship between the RCLK and the CLK input to the chip can also be checked for at this time.

4. **EOI status circuitry:** The EOI pin must initially be low (as checked by the power on tests) and then go high after the programming is complete. At this point any s-a-0 fault on the EOI pin would be detected. Note that a s-a-1 fault on the EOI would have been detected by the power on check. Also after the SRAM has been initialized the EOI bit in the status register would have been set. This bit remains set till a reset occurs.

5. **SRAM faults:** The checking of the faults associated with the SRAM are now described.

   (a) **Stuck at 0 and stuck at 1 faults:** The checker board pattern 1010... is first written and then read from the SRAM. This checks for the s-a-0 faults on half of the bit cells (which have been programmed with

1s). The other half (those programmed with 0s) are checked for s-a-1 faults because any of these faults would disrupt the regularity of the 1010... pattern. Next the chip is reset and a complimentary pattern of the above, 0101..., is written to the SRAM. The SRAM contents are then read again. Any s-a-0 faults in the bit cells programmed with 1s (which had earlier been programmed with 0s and tested for s-a-1) or s-a-1 faults on bit cells programmed with 0s (earlier programmed with 1s and tested for s-a-0) would disrupt the regularity of the 0101... readout pattern.

(b) **Bridging faults:** Due to checkerboard pattern programming every bit cell programmed with 0 is surrounded by bit cells programmed with 1s and in turn every bit cell programmed with 1 is surrounded by bit cells programmed with 0s any bridging faults between the adjacent cells will disrupt the regularity of the checkerboard pattern and will get detected on readout.

6. **Reset circuitry:** After programming the 1010... pattern the chip is reset and the status on the out put pins checked again as in the power on check. The *status generation test* are performed again to verify the effects of reset. Next the 0101... pattern is serially loaded onto the SRAM and all the above tests are repeated.

To sum up the SRAM check tests the following circuits in the chip:

1. SRAM bit cells.
2. SRAM read and write pins (SDI and SDO).
3. SRAM read and write control (ESDI, ESDO, RCLK).

4. EOI status generation (EOI pin and EOI bit in the status register).

Chapter 4

PLA Cell Testing

4.1 Faults in PLAs

There can be three types of faults in PLAs [1-3].

4.1.1 Stuck At Faults

These faults result from a metal, polysilicon or diffusion path being shorted to ground or positive supply voltage. Even an open circuit manifests as a stuck at fault, for example a open circuit gate can appear as a stuck at **0** fault. Also a short between the source and drain may be construed as a stuck at **1** or stuck at **0** on the drain of a NMOS and PMOS transistor respectively. Any shorting of the polysilicon gate to the drain or source due to defects in the SiO_2 layers will also effect in a stuck at fault.

A stuck at fault may or may not propagate along the circuit network [4-8]. Hence test vector sets must allow for propagation of all possible stuck at faults. A stuck at fault may pass through several polarity inversions at each transistor stage. Figure 4.1 shows a 2-1 input AND-OR-INV gate and propagation of stuck at faults on the

inputs.

4.1.2 Bridge Faults

Bridge faults are caused by shorting of two or more metal, polysilicon or diffusion lines. As noted above certain types of shorts such as those between a gate and source manifest as stuck at faults. For testing purpose bridge faults represent the abnormal connection of two or more different inputs that results in a faulty output different from the output because of a stuck at fault Figure 4.1 also shows a bridge fault and the resulting response.

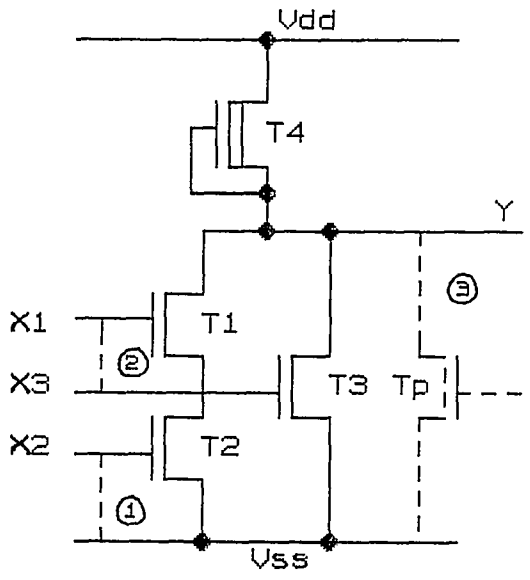
4.1.3 Cross Point Faults

A crosspoint fault is caused by the abnormal absence or presence of a diode or transistor. For example a missing transistor T3 in the circuit of figure 4.1 is a crosspoint fault. The presence of an extra parasitic transistor T_P also constitutes a crosspoint fault. Usually cross point faults arise in circuits using grid or array models such as memories, PLAs etc.

The cross point fault in the above example is equivalent to a stuck at 0 fault on the gate of transistor T3. However a parasitic transistor across the source and drain of T3 would be harder to detect especially if it turned on and off with time.

4.2 Testing of PLAs

The testing of PLAs is a unique problem of its own. Unlike the SSI and MSI circuits there is no standard way of testing a PLA. However the regular and structured architecture of PLAs allow their testing with simpler schemes than other LSI and



(a)

X1	X2	X3	Y
s-a-0	d	0	1
s-a-1	1	d	0
0	d	s-a-0	1
d	0	s-a-0	1
d	d	s-a-1	0

d = don't care

(b)

X1	X2	X3	Y
1	d	X1	0
0	d	X1	1

(c)

Figure 4.1 Faults in PLAs. (a) 1: Stuck at fault, 2: bridge fault and 3: cross point fault. (b) Output response for stuck at faults on input. (c) Output response for a bridge fault. Note how the bridge fault changes the circuit's logic to an inverter.

VLSI devices.

A PLA could be tested by the manufacturer or the user. These two groups have markedly different test philosophies, goals and equipment. The different aspects of production and user testing are now discussed.

4.2.1 Production Testing

The production testing must be exhaustive. It must check for all the types of faults that may result due to processing and packaging irregularities. The simple way of testing a PLA is straightforward but rather inefficient. A look at the simple testing technique will highlight the many a uniqueness of PLA testing. To test the PLA well we must excite all faults and check the output for incorrect responses. However most PLAs are one time programmable. This means that once we *burn* a circuit onto a PLA we cannot alter it. This is indeed the severest limiting factor for the straightforward testing technique. The fault coverage of such a test would depend upon the fraction of total nodes used in *burning* of the test circuit onto the PLA. This fraction is in turn dependant on the complexity of the circuit *burned* in. Greater complexity would require generation of more complex (with respect to size as well as sequence) test vectors. For any circuit's implementation on a PLA the following rule applies:

> Any non trivial circuit implementation cannot have a node coverage greater than 50% in the INPUT[1] array.

[1]Different PLA schemes use different implementations such as AND-OR, NAND-NAND, NOR-NOR etc. The general form used here is INPUT-OUTPUT array.

This is true because no non trivial[2] sum or product term can have a signal and its complement together. Hence the test circuit realization can use a maximum of 50% nodes in the INPUT array. Therefore the fault coverage of the INPUT array cannot exceed 50%. The node coverage (and hence fault coverage) of the OUTPUT array depends upon the number of INPUT terms in the OUTPUT array. For a very small number of INPUT terms it is possible to have 100% node coverage but this is very rare. Usually a PLA will have 8-24 INPUT terms per OUTPUT gate. Fig. 4.2 illustrates the node coverage possible with some PLAs implementations of medium complexity circuits. The density of dots in the AND and OR arrays gives an idea about the node coverage. This explains why simple testing of PLAs is rarely done by a user.

The above discussion highlights that it is impossible to test with 100% node coverage on a one time programmable PLA. However a manufacturer must test his product completely before marketing. Usually a manufacturer would take a sample of chips from a production run. Each of these chips would have a different *pattern* burned onto them so that they collectively cover all nodes. Different sets of the sampled chips would be tested with members of each set collectively covering all faults. This would give a statistical yield for the production run. A manufacturer might also test pre-packaged chips using wafer probes and other test equipment. Wafer probing is done when packaging costs are high and yield is relatively low. Wafer probe also helps in fault location and determination of fault causes. However the wafer probe method is quite delicate, expensive and time consuming. Hence it is desirable to be able to test a chip to the greatest possible extent from the external

[2]A non trivial sum or product term is one that does not have an output fixed at 0 or 1 irrespective of the input

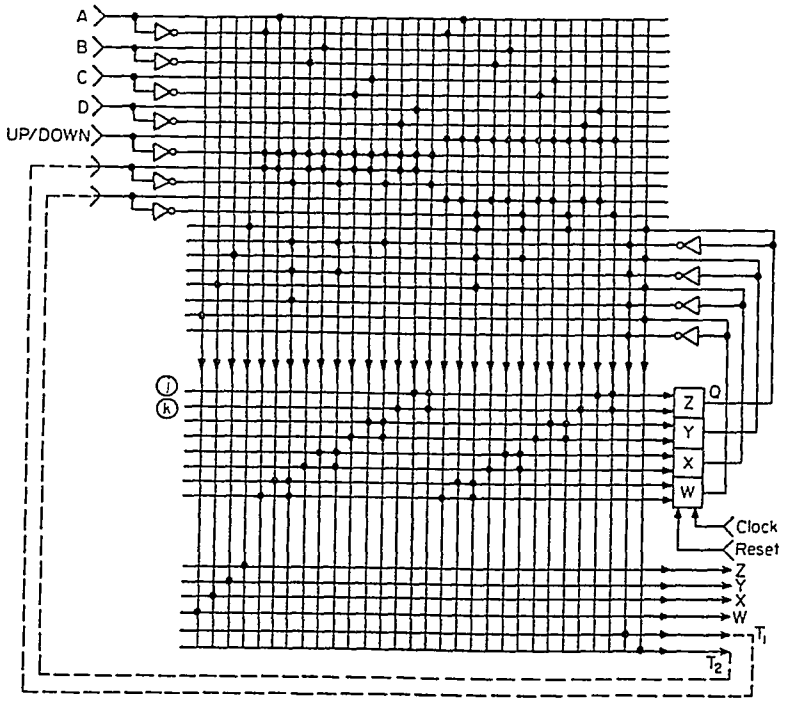


Fig. 4.2 a Four-bit up/down, variable-modulo counter using PLA array.

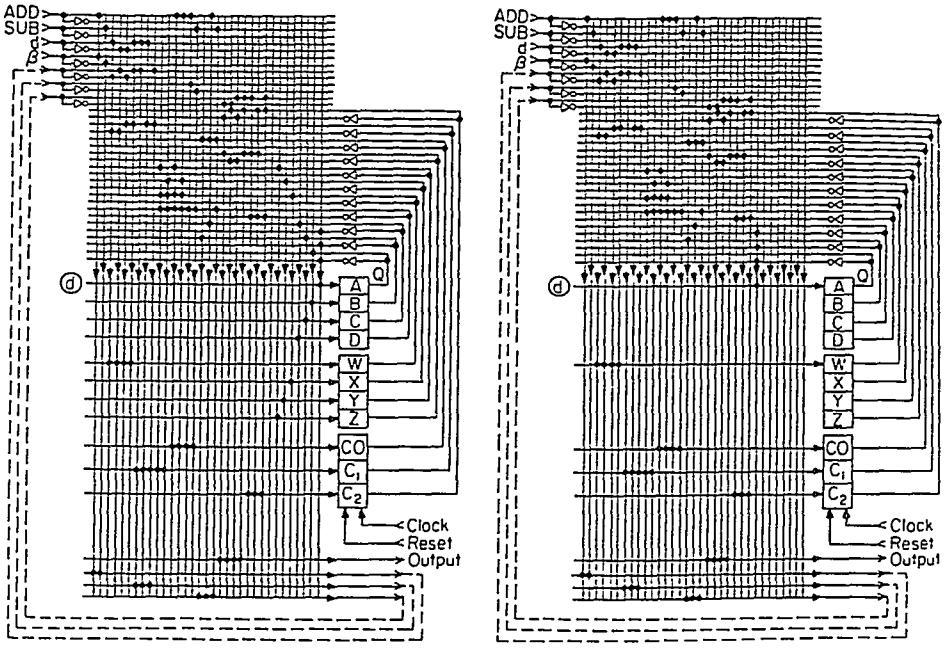


Fig. 4.2 b Bit-serial adder/subtracter using PLA format (4-bits)
PLA implementation with *D*-type flip-flops only, PLA implementation with shift registers substituted in the feedback circuit.

Adapted from "MOS/LSI Design and Application", William N. Carr & Jack P. Mize Texas Instruments Electronics Series, McGraw Hill, 1972.

pins.

4.2.2 User Testing of PLAs

The impossibility of testing every node of a one time programmable PLA was discussed above. In practice it is not even necessary for a user to test all the nodes of the PLA. A user would normally *burn* in a circuit of his own application. After that he is concerned only with the fault free working of his circuit and it is immaterial whether any faults develop in the unused resources[3] of the PLA.

To aid the complete testing of the user implemented portion of the PLA it is recommended that the user *burn* additional redundancy or test circuitry along with his own circuit. The redundancy or the test circuitry is then used to check itself and the user circuit. Many schemes of creating a testable PLA design have been proposed [9-13]. These include error detection and correction encoding of outputs, extra terms in the output function to create *disjoint* terms etc.

4.3 PLA Cell Testing in the NJIT PLD

A user normally *burns* some redundancy or test circuitry along with the actual circuit on the PLA. However the NJIT PLD may be reprogrammed at will. Each reprogramming may be intended to result in a different circuit realization. So the user might have to design a different test circuit for each application. Some times the new design may even be adaptive and indeterministic beforehand. This would make testable design very formidable.

[3]The unused input lines, output lines and other gates etc. are collectively referred to as the unused resources of the PLA.

However the NJIT PLD exploits reprogrammability to permit enhanced testing. Reprogramming of the SRAM matrix to yield a new circuit realization at will allows comprehensive production as well as user testing for a complete fault coverage. This is a unique unison of production and user testing and both tests are identical. The user is freed from creating a special design for testability and generating test vectors. The production testing is made much more economical because all faults are tested from the pins. Wafer probing is required only for location of faults due to processing imperfections. This reduces the time and costs of wafer probing. Packaged chips are completely testable for fault detection at any time.

The detailed testing of the PLA cell is discussed in the next sections. Along with the testing of the PLA cells sections of other interacting circuitry such as address decoder, input-output buffer and timing and control circuitry are also tested.

4.3.1 Taxonomy and Nomenclature of PLA Cell Sections

Figure 4.3 shows a complete schematic of a PLA Cell. The PLA is implemented as a NOR-NOR matrix. These will henceforth be referred to as INPUT NOR and OUTPUT NOR (abbreviated as I/P NOR and O/P NOR) matrices. The I/P NOR matrix has 20 NOR gates. Each gate has 24 *programmable* inputs (8 inputs, 8 complementary of inputs, 4 feedback Lines and 4 complementary feedback Lines). The O/P NOR matrix has 8 gates each with 20 *programmable* inputs. Each input to the O/P NOR matrix gate corresponds to a output of an I/P NOR gate.

The feedback is through 4 D type flip flops. The inputs and outputs of these flip flops constitute the FEEDBACK matrix.

The inputs to the I/P NOR matrix are called **X Inputs**. Thus $X_1X_2...X_8$ are the inputs to the PLA cell. These are latched from the external data bus $D_0 - D_7$

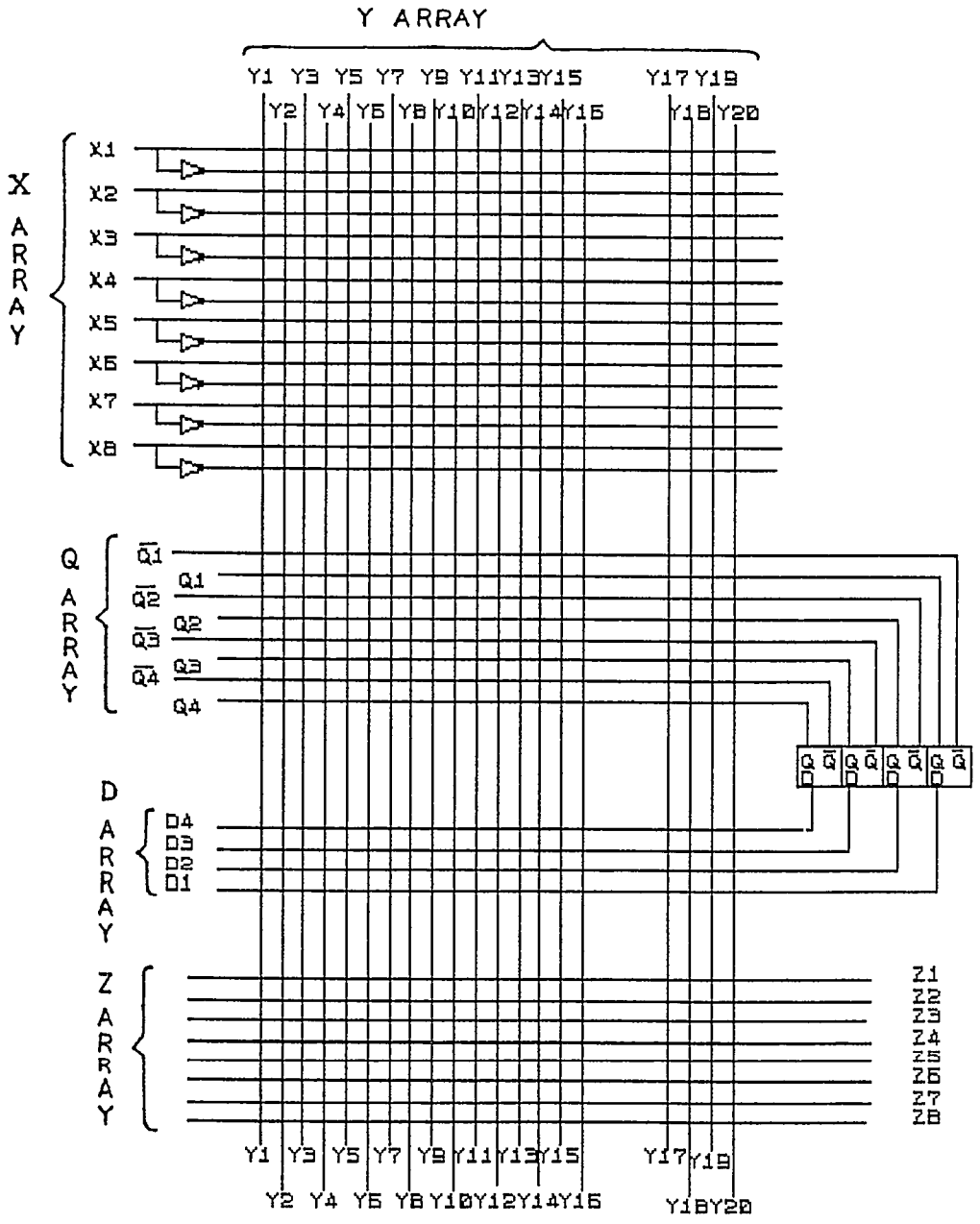


Figure 4.3 Naming of the different PLA Cell sections

to the Cell Input Latch of the PLA cell. Hence any test vector for the PLA cell check is a set of 8 elements in the form

$$X = \{X_1 X_2 ... X_8\}$$

where $X_i = 0$ *or* 1. The 8 input Lines and their 8 complements are collectively referred to as the **X Array** or the **X Lines**. Thus the X Array has 16 elements whereas there are only 8 X Inputs.

The output of each 24 I/P NOR gate run on metal Lines across the chip. These outputs of the I/P NOR gates are called **Y Lines** and are collectively referred to as **Y Array**. The status of these Y Lines in response to a test vector

$$X = \{X_1 X_2 ... X_3\} \ isgivenby \ Y = \{Y_1 Y_2 ... Y_{20}\}$$

where $X_i, Y_i = 0$ *or* 1.

The 8 output Lines are also called **Z Lines**. These form an 8 element **Z Array**

$$Z = \{Z_1 Z_2 ... Z_8\}$$

where $Z_i = 0$ *or* 1.

The feedback is through 4 D type flip flops. The inputs to these flip flops are called the **D Lines** and constitute a 4 element **D Array** such that

$$D = \{D_1 D_2 ... D_4\}$$

where $D_i = 0$ *or* 1.

There are four normal (Q) and four complementary ($\overline{Q}$) outputs from the D flip flops. These are collectively referred to as **Q Lines** and constitute a 8 element **Q Array** such that

$$Q = \{Q_1 \overline{Q_1} ... Q_4 \overline{Q_4}\}$$

where $Q_i = 0$ *or* 1.

Table 4.1 summarizes all these notations.

4.3.2 General Test Sequence

The test sequence for the PLA cell testing consists of the following steps.

1. Reset the PLD chip and program the SRAM so as to realize a test circuit configuration in each PLA cell.

2. Apply one or more test vectors on the X Lines and obtain the response from the Z Lines.

3. Compare the obtained response with the expected response and check for errors.

4.3.3 Test Length

The total length of the test depends upon various factors. It also depends upon the test equipment and apparatus set up times. However the test time measured herein will be the time used exclusively for the checking of the PLA cell. This time depends upon:

1. PLD reset time (T_{RESET}).

2. SRAM read and write times (T_{SRAM}).

3. Total number of test vectors. The time spent on applying a test vector to the PLA cell and obtaining the response is

$$T_{TV} = T_{WR} + T_{RD}$$

Group Name	Symbolic Representation	Remarks
I/P NOR Matrix	————————	Consists of 20 INPUT NOR gates each with 24 programmable inputs.
O/P NOR Matrix	————————	Consists of 8 OUTPUT NOR gates each with 20 programmable inputs.
FEEDBACK Matrix	————————	Consists of the four feedback flip flops and their input and output matrices.
I/P Array or I/P Lines	$X = \{X_1 X_2 ... X_8\}$	This is the 8 bit input to the PLA Cell. This comes from the data bus ($D_0 - D_7$) Lines.
X Array or X Lines	$X = \{X_1 X_2 ... X_{16}\}$	These are the 8 inputs and their complements. Thus the INPUT Array is a subset of the X Array.
Y Array or Y Lines	$Y = \{Y_1 Y_2 ... X_{20}\}$	These are the 20 outputs of the INPUT NOR gates.
Z Array or Z Lines	$Z = \{Z_1 Z_2 ... Z_8\}$	These are the 8 outputs of the OUTPUT NOR gates.
D Array or D Lines	$D = \{D_1 D_2 ... D_4\}$	These are the 4 inputs to the D type flip flops.
Q Array or Q Lines	$Q = \{Q_1 \overline{Q_1} ... Q_4 \overline{Q_4}\}$	These are the four normal (Q) and four complementary ($\overline{Q}$) outputs from the D flip flops.

Table 4.1: PLA Cell Sections

therefore $T_{test} = T_{Reset} + 2 * T_{SRAM} + n * (T_{WR} + T_{RD})$ where n is the number of test vectors in a test. The total time to test the chip is the sum of times for all the tests.

The reset time for the chip is fixed at $1ms$. The SRAM read and write times are also fixed and depend upon the CLK frequency. At $10MHz$ the read or write time for the SRAM is $4.7ms$. Hence the test design challenge has two aspects:

1. Design of such PLA configurations that permit a wider fault coverage for each implementation. Thus every programming of the SRAM should be designed to result in a circuit that checks for maximum number of faults in the PLA cell.

2. Minimization of total number of test vectors.

A comparison of the test times must be made when there are different and alternative ways to test. It must be kept in mind that the time taken to reprogram the SRAM is greater than of a write-read cycle by an order of 10000.

4.4 Test 1 (Testing of the X,Y and Z Arrays)

The first test involves a programming pattern that would detect s-a-0 and s-a-1 on the X Lines, s-a-0, s-a-1 and bridge faults on the Y Lines and s-a-0, s-a-1 and bridge faults on the Z Lines. The first step is loading of a special pattern into the SRAM. This pattern is *equal weightage, half matrix, single activation and interleaved* (abbreviated as EWHMSAI pattern), see figure 4.4. The characteristics of this pattern that aid testing are:

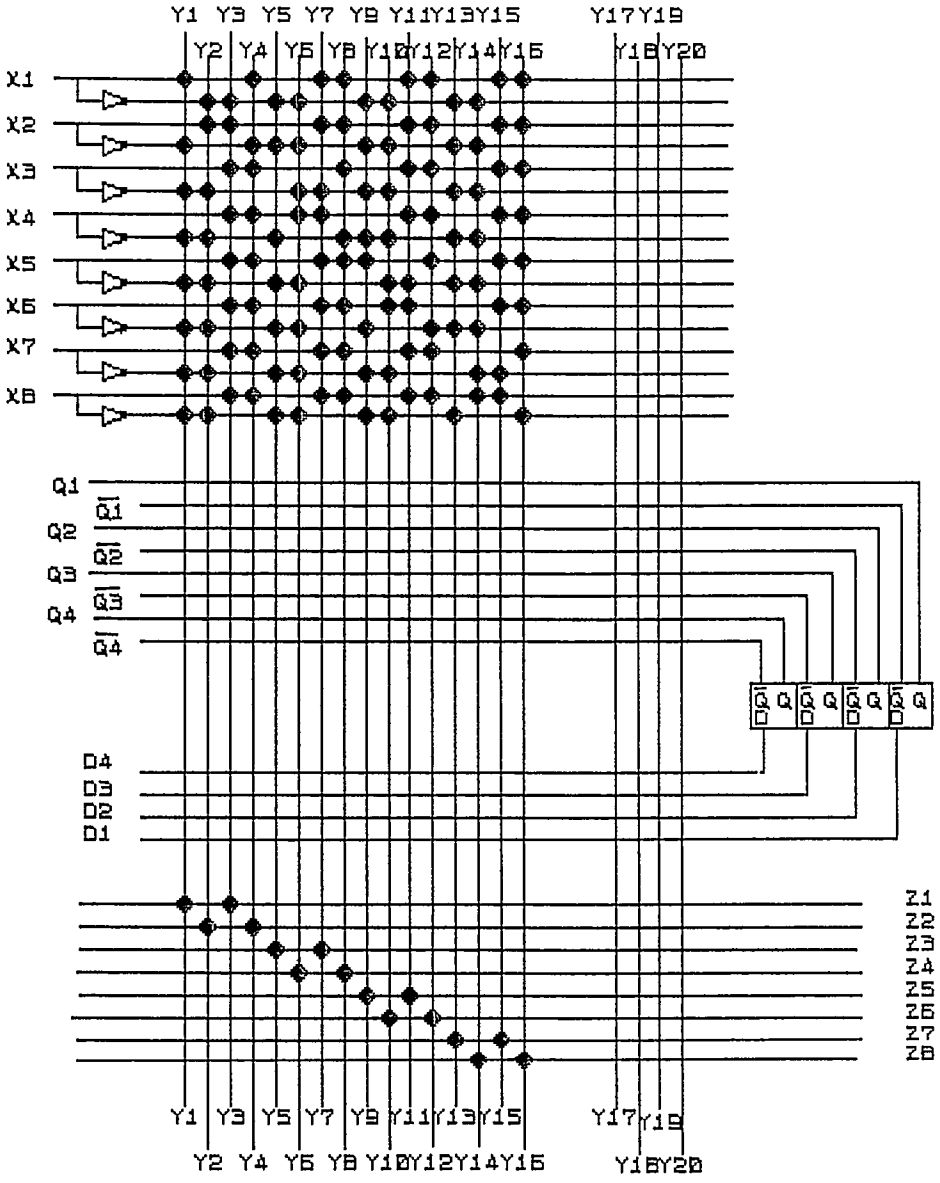


Figure 4.4 EWHMSAI Pattern for PLA Cell Test 1

1. **Equal Weightage:** All the X Lines are loaded the same. So are all the Y Lines and all the Z Lines. The equal loading makes the transient and heating effects be distributed along the chip area and minimizes spurious faults and other irregularities.

2. **Half Matrix:** As discussed in a previous section the INPUT NOR matrix cannot have a node coverage greater then 50%. The EWHMSAI pattern has the maximum node coverage of 50%. This is very advantageous in testing of cross point open faults since it is possible to test for all crosspoint faults with *two* programmings of the SRAM.

3. **Single Activation:** Single activation implies that in each of the X, Y and Z arrays only one line is active (high or low) at any time and all the rest are complementary. This is very essential for bridge fault detection.

4. **Interleaving:** Two adjacent signals that are input to the same gate can mask each other's bridge faults. Hence no two Y Lines form inputs to the same gate.

Design of the EWHMSAI Pattern: For a PLA with n inputs (including the complementary inputs as well) the EWHMSAI pattern will cover $(n*n)/2$ nodes in the Input matrix and $(n*m)/2$ nodes in the Output matrix where the PLA has m outputs. This pattern is derivable for an even number of X inputs (not including the derived $\overline{X}$ inputs). For PLAs having odd X inputs two even subsets of the inputs may be considered such that they together cover all inputs. The complementary of the EWHMSAI pattern shown in figure 4.6 is also an EWHMSAI pattern and is used to check the other $(n*n)/2$ and $(n*m)/2$ nodes in the Input and Output matrices

with the same test vectors as will be derived for the first EWHMSAI pattern.

As an example the design of the EWHMSAI pattern for the 8 input 8 output PLA shown in figure 4.2 will now be presented. Since there are 8 X and 8 $\overline{X}$ inputs the EWHMSAI pattern 1 will cover 16 Y Lines. Each of these Y Lines (or Input Nor Gates) will have 8 inputs. One input to each gate will be complementary type of all the other inputs to that gate. In the present example one Y Line will have $X_1, \overline{X_2}, \overline{X_3}, ..., \overline{X_8}$ as inputs, another Y Line will have $\overline{X_1}, X_2, X_3, ..., X_8$ as inputs and yet another Y Line will have $\overline{X_1}, X_2, \overline{X_3}, ..., \overline{X_8}$ as inputs and so on. Note that the inputs $X_1, \overline{X_2}, \overline{X_3}, ..., \overline{X_8}$ and $\overline{X_1}, X_2, X_3, ..., X_8$ are complementary of each other. All the complementary input patterns are placed such that there is one other pattern between them (interleaving). The Z Lines (output Nor gates) in the Output matrix have alternate Y Lines as inputs and every Y Line and its complimentary Y Line are both inputs to the same Z Line. This results in the EWHMSAI pattern shown in figure 4.3. For PLA structures with other I/P and O/P matrix sizes similar patterns may be derived by the procedures outlined above.

We now consider the various faults detected by using the pattern shown in figure 4.3. In the following discussion a italicized value indicates a **fault sensitive** value in an array. Thus $X = \{00100110\}$ indicates that X_2 and X_6 are fault sensitive and their value would change with any of the faults under consideration.

4.4.1 Stuck at 0 Fault Detection in the X Array:

The test vectors for the stuck at 0 fault detection, the resulting Y Line status and the responses on the Z Lines are shown in the table for Test 1a.

Figure 4.5 shows the signal paths that are tested with the first vector as high-

$X = \{X_1X_2...X_8\}$	$Y = \{Y_1Y_2...Y_{16}\}$	$Z = \{Z_1Z_2...Z_8\}$
X={0000 0000}	Y={**0000 0000 0000 0000**}	Z={**11111111**}
X={1111 1111}	Y={**0000 0000 0000 0000**}	Z={**11111111**}

Table 4.2: Test 1a (Stuck at 0 Fault Detection in the X and Z Arrays, Stuck at 1 fault detection in the Y Array)

$X = \{X_1X_2...X_8\}$	$Y = \{Y_1Y_2...Y_{16}\}$	$Z = \{Z_1Z_2...Z_8\}$
X={0111 1111}	Y={**1**000 0000 0000 0000}	Z={**0**111 1111}
X={1011 1111}	Y={**0**100 0000 0000 0000}	Z={1**0**11 1111}
X={1101 1111}	Y={0000 **1**000 0000 0000}	Z={11**0**1 1111}
X={1110 1111}	Y={0000 **0**100 0000 0000}	Z={111**0** 1111}
X={1111 0111}	Y={0000 0000 **1**000 0000}	Z={1111 **0**111}
X={1111 1011}	Y={0000 0000 **0**100 0000}	Z={1111 1**0**11}
X={1111 1101}	Y={0000 0000 0000 **1**000}	Z={1111 11**0**1}
X={1111 1110}	Y={0000 0000 0000 **0**100}	Z={1111 111**0**}
X={1000 0000}	Y={00**1**0 0000 0000 0000}	Z={**0**111 1111}
X={0100 0000}	Y={000**1** 0000 0000 0000}	Z={1**0**11 1111}
X={0010 0000}	Y={0000 00**1**0 0000 0000}	Z={11**0**1 1111}
X={0001 0000}	Y={0000 000**1** 0000 0000}	Z={111**0** 1111}
X={0000 1000}	Y={0000 0000 00**1**0 0000}	Z={1111 **0**111}
X={0000 0100}	Y={0000 0000 000**1** 0000}	Z={1111 1**0**11}
X={0000 0010}	Y={0000 0000 0000 00**1**0}	Z={1111 11**0**1}
X={0000 0001}	Y={0000 0000 0000 000**1**}	Z={1111 111**0**}

Table 4.3: Test 1b (Stuck at 1 Fault detection in the X and Z Arrays, Stuck at 0 in Y Arrays and Bridge Faults in Y and Z Arrays.)

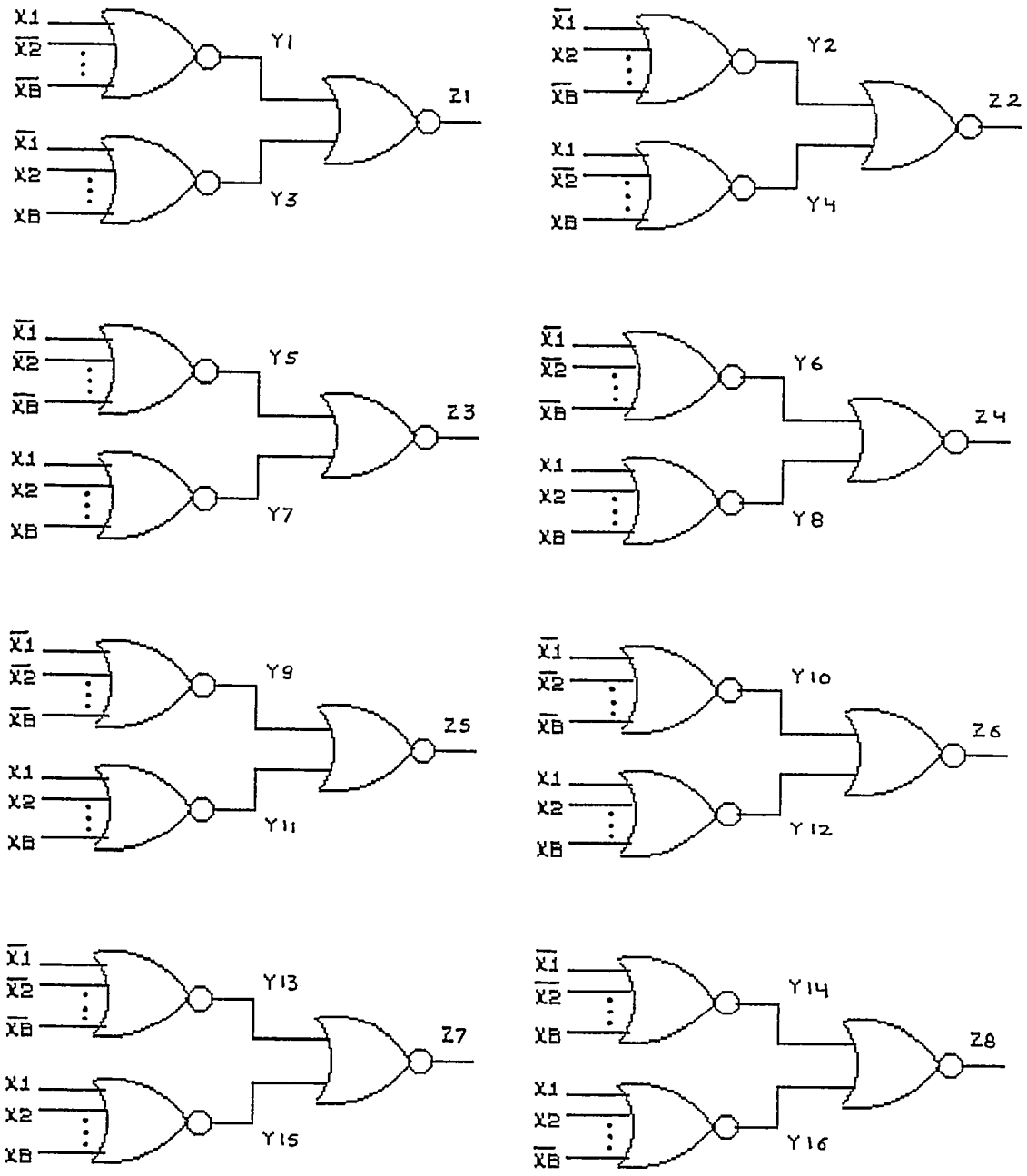


Figure 4.5 Circuit realization for EWHMSAI pattern 1.

lighted. The first test vector would detect s-a-0 faults on all the $\overline{X}$ Lines. The second test vector checks for all the s-a-0 faults on the X Lines.

4.5 Stuck at 1 Fault Detection in the X Array

Test 1b shows the sixteen test vectors for s-a-1 fault on the X Lines. However only the first eight or the second eight would by themselves detect all the s-a-1 faults on the X Lines. Note that a stuck at 1 fault on an X Line which is low (0) would result in the corresponding fault sensitive Y Line being driven low which would in turn make the fault sensitive Z Line go high (1).

Note that each test vector checks for s-a-1 on one X Line and seven $\overline{X}$ Lines.

4.5.1 Stuck at 0 Fault Detection in the Y Array

The sixteen test vectors in the s-a-1 test for X Array result in a *single activation* of the Y Lines, that is only one Y Line is **1** at any time. If this Y line is s-a-0 the corresponding fault sensitive Z Line will go high. Thus the s-a-0 faults on the sixteen of the Y Lines are detected by the s-a-1 for X Line test. Stuck at 0 on other four Y Lines are detected by TEST 4.

4.5.2 Stuck at 1 Fault Detection in the Y Array

The Y Line s-a-1 faults are also detected by the X Line s-a-0 fault detection tests. Note that in the X Line s-a-0 tests the Y Lines are always **0**. Any s-a-1 fault on the Y Lines will result in the corresponding Z output going low.

4.5.3 Stuck at 0 Fault Detection in the Z array

In the s-a-0 test for the X Lines all the Z outputs were always **1**. Any of these s-a-0 will show as a faulty response to the input test vector.

4.5.4 Stuck at 1 Fault Detection in the Z Array

During the s-a-0 fault detection in the X Array one out of eight Z Lines was normally **0**. A s-a-1 on the Z line would evidently show up as a faulty response to a test vector.

4.5.5 Y Array Bridge Faults

Again note that during the s-a-1 testing of the X Lines a *rolling* **1** pattern appeared at the Y Lines. Besides checking for s-a-0s on the Y Lines these vectors would also detect any bridging faults between adjacent[4] Y Lines. Again this is because of *single activation.* A bridge between any active (**1**) Y Line and a adjacent Y Line would result in the same logic on both of them. This would result in either double activation or null activation and generate a faulty output response. It is important to note the **interleaved pattern** in the OUTPUT NOR array. Had there been no interleaving then a bridge fault masking may have occured because the bridging of two Y Lines such that both evaluate as high would have gone un-noticed because a NOR gate responds with a **0** output when even more than one inputs are high. Interleaving guarantees that adjacent Y Lines are inputs to different NOR gates and a bridge resulting in double (or multiple) activation would result in multiple activations at the OUTPUT NOR array as well.

[4] A bridge fault between any non adjacent Y Lines would also bridge the intervening adjacent Y Lines.

4.5.6 Z Array Bridge Fault Detection

The X array s-a-1 fault detection tests result in a single Z Line being active (**0**) at any time. A bridge to any adjacent line, all of which are 1, would result in multiple activation or null activation. Hence the Z array bridge faults also get detected by the X Array s-a-1 fault tests.

4.5.7 INPUT and Output Array Crosspoint Faults

The checking for the INPUT and Output array crosspoint opens and shorts with the EWHMSAI 1 Pattern is defined in a later section.

4.6 Test 2 (Testing of the X,Y and Z Arrays)

The second test involves the programming of another EWHMSAI pattern in the PLA. This pattern is however complimentary of the pattern used in test 1. Thus all 0s in test pattern 1 are 0s in the test pattern 2 and vice versa. Figure 4.6 shows the new EWHMSAI pattern.

The second EWHMSAI Pattern is used to detect crosspoint shorts and opens in the INPUT array. Since each EWHMSAI pattern covers half of the matrix the two tests put together have 100% fault coverage. The details of the crosspoint tests are presented in another section.

4.7 Test 3 (Testing of X Array Bridge Faults)

This test is to check for bridge faults in the X array. Figure 4.7 and 4.8 show the PLA cell programming and the circuit realization.

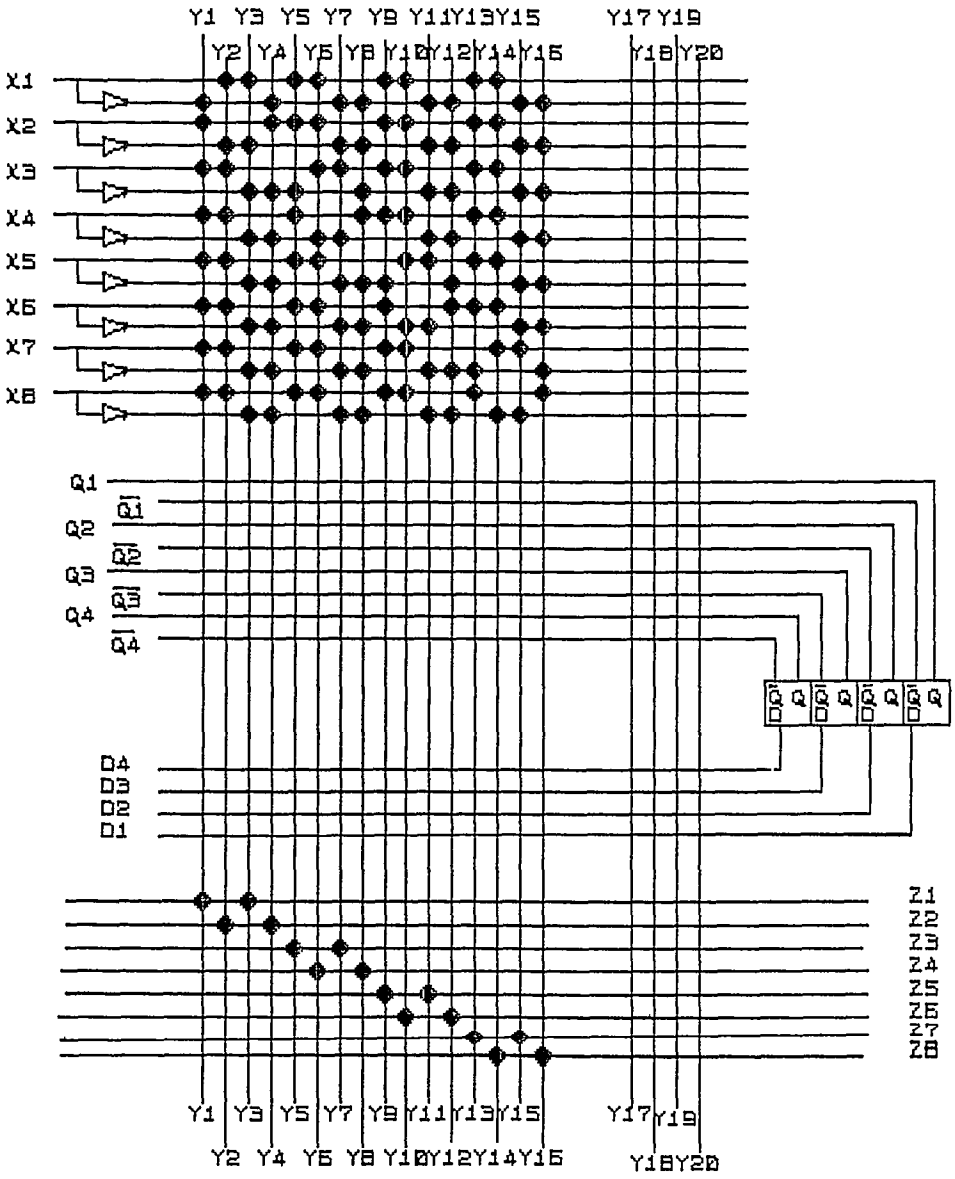


Figure 4.6 EWHMSAI Pattern for
For PLA Cell Test 2

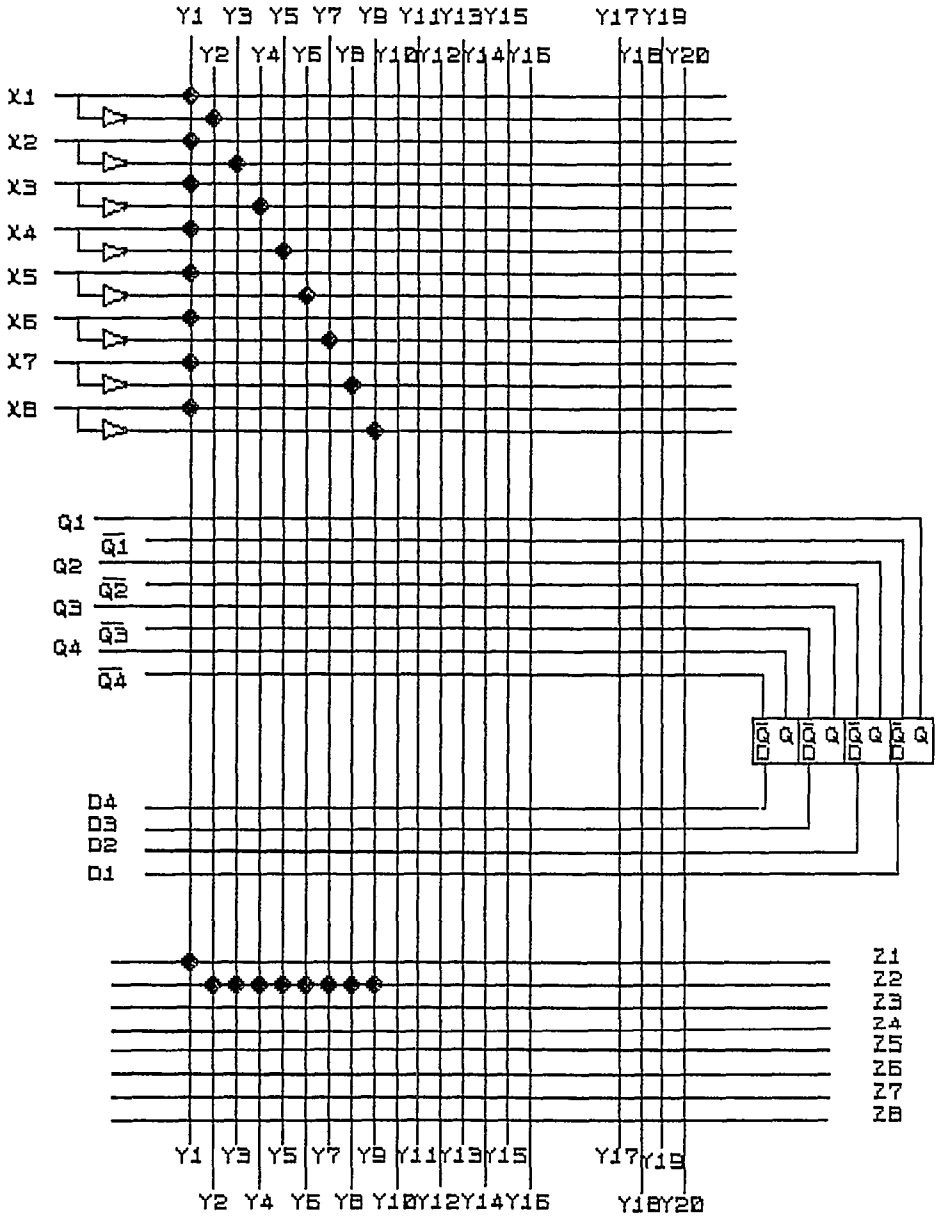


Figure 4.7 PLA Cell Pattern for Test 3

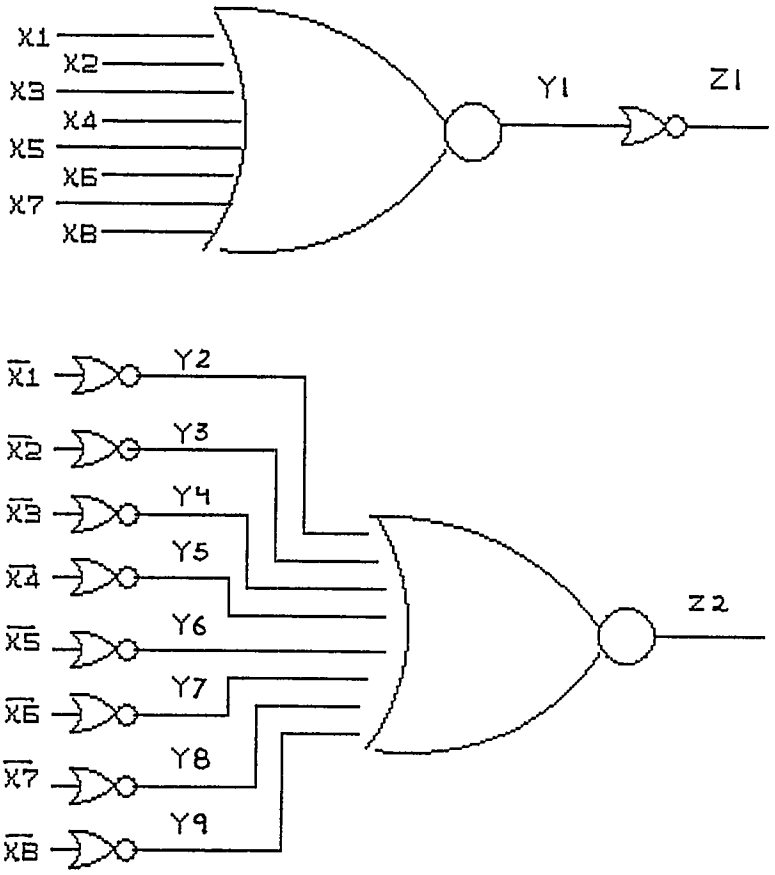


Figure 4.8 Circuit realization for Test 3 pattern.

$X = \{X_1X_2...X_8\}$	$Y = \{Y_1Y_2...Y_{16}\}$	$Z = \{Z_1Z_2...Z_8\}$
X={0000 0000}	Y={**1000 0000 0**ddd dddd}	Z={**01**dd dddd}

where d = *don't care.*

Table 4.4: Test 3 (Testing of X Array Bridge Faults)

If all the X Inputs are driven low then any two adjacent lines in the X array will be at different logic levels and a bridge between them would change the logic level on at least one of them. It now remains to design a circuit that would detect a abnormal change in the logic level of each X Line.

The test vectors and the resulting state of Y Lines and the output on the Z lines are given below.

Thus a single test vector is sufficient to detect all bridge faults in the X array. Also note the rather sparse node coverage in the INPUT and OUTPUT Arrays. This allows another test configuration for testing the FEEDBACK Array to be programmed together with the current test pattern.

4.8 Test 4 (Testing of the FEEDBACK Array

This test checks for all faults (except crosspoint) in the FEEDBACK Array. The different faults that are checked are:

1. Stuck at 0 and 1 on D Lines.

2. Bridge faults on D Lines.

3. Stuck at 0 and 1 faults on Q Lines.

4. Bridge faults on Q Lines.

$X_1X_2...X_8$	$Y_{16}Y_{17}...Y_{20}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_{15}Y_{14}...Y_9$	$Z_1Z_2...Z_8$
0 1010	1 0101	1010	10 01 10 01	0 1 1010 0	dd01 001d
1 0101	0 1010	0101	01 10 01 10	1 0 0101 0	dd10 101d

where *d=don't care.*

Table 4.5: Test 4a (Stuck at 0, Stuck at 1 and Bridge Fault Detection on D Lines, Stuck at 1 Fault Detection on Q Lines)

5. Stuck at 0 and 1 on $Y_{17}...Y_{20}$.

6. Bridge faults on $Y_{16}...Y_{20}$.

Note that $Y_{16}...Y_{20}$ Lines were not checked in the tests 1 and 2.

Figure 4.9 & 4.10 shows the PLA cell pattern and the resulting circuit realization.

4.8.1 Stuck at 0 Fault Detection on D Lines

To check for stuck at 0 fault on the D Lines each feedback flipflop is loaded with 1 and its output response checked. The inputs $X_2X_3X_4X_5$ undergo two inversions each before feeding the D input of the feedback flipflops. A stuck at 0 fault on any D Line would result in an abnormal output on Z_4 *or* Z_5.

The test vectors for the test are given in Test 4a.

4.8.2 Stuck at 1 Fault Detection on D Lines

This tests proceeds similar to the stuck at 0 fault detection on the D Lines with the difference that the response of the flipflops in response to a 0 input is observed (Test 4a).

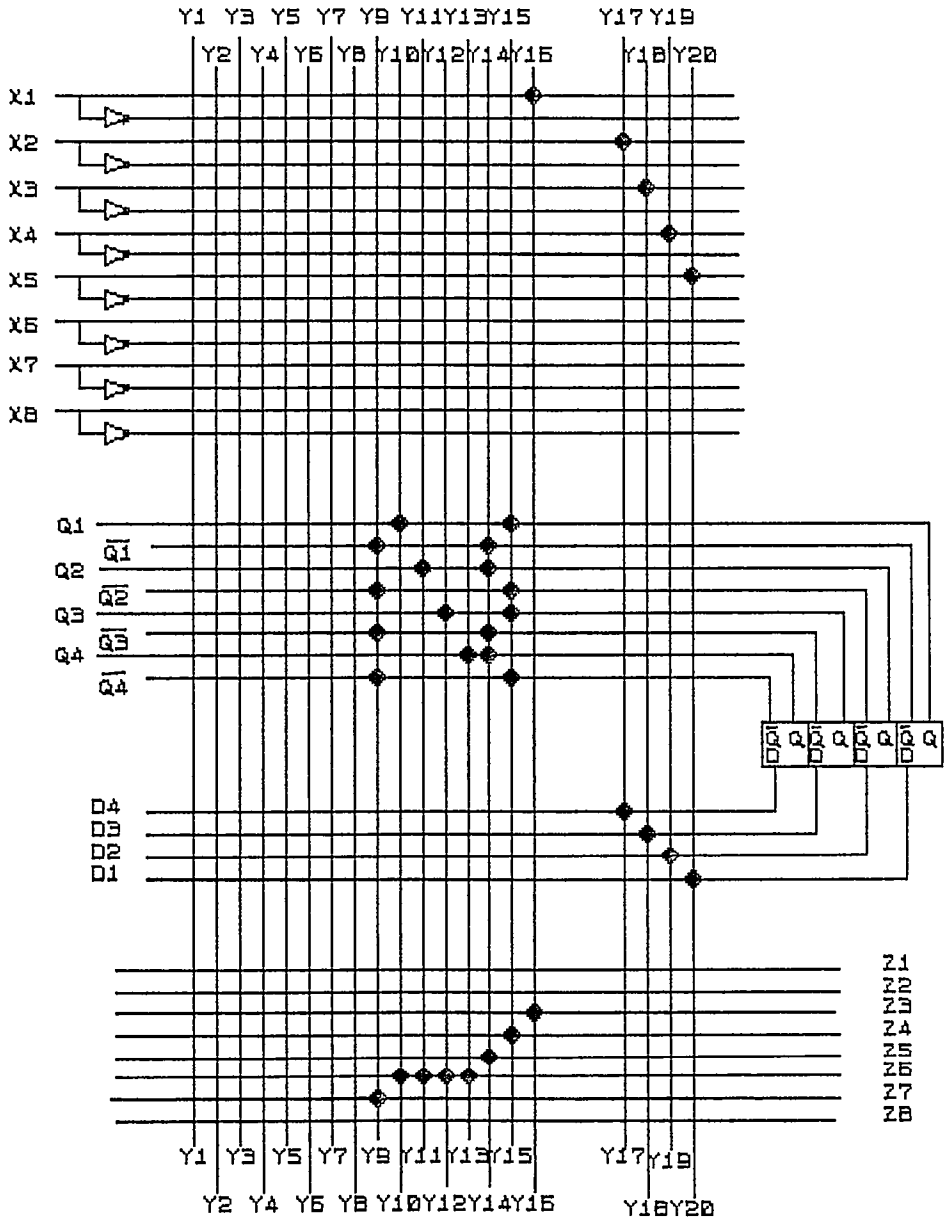


Figure 4.9 PLA Cell Pattern for Test 4.

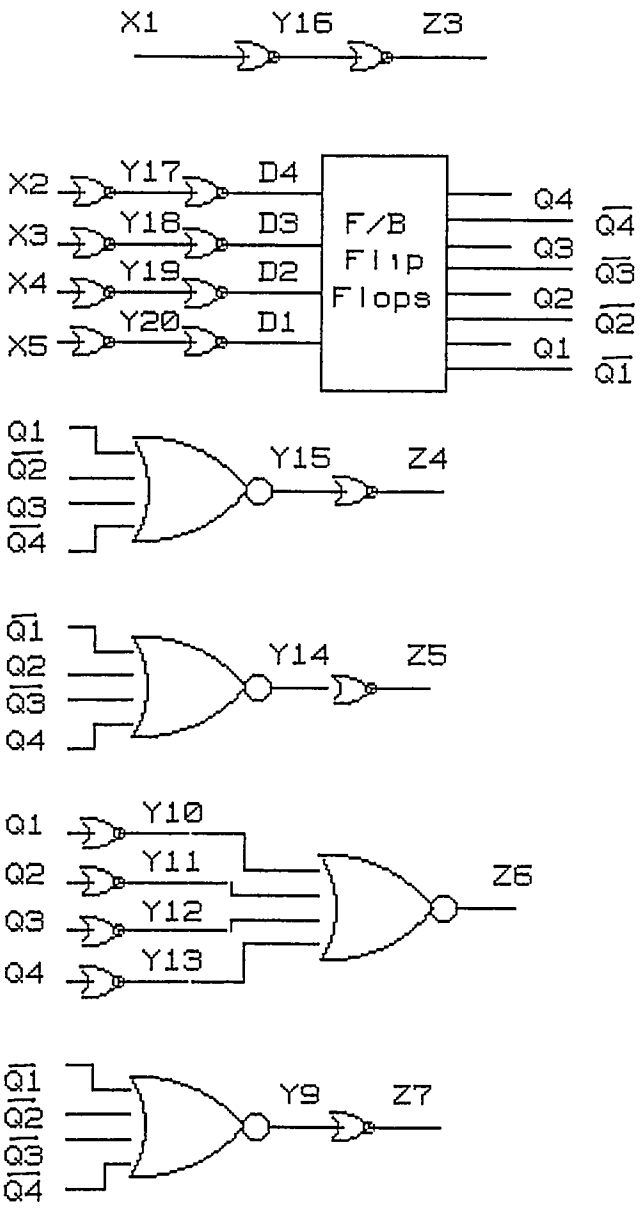


Figure 4.10 Circuit realization for Test 4 pattern.

$X_1X_2...X_8$	$Y_{16}Y_{17}...Y_{20}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_{15}Y_{14}...Y_9$	$Z_1Z_2...Z_8$
0 1101	1 0010	1101	1010 0110	0 0 1101 0	dd01 101d
0 0111	1 1000	0111	0110 1010	0 0 0111 0	dd01 101d
0 1110	1 0001	1110	1010 1001	0 0 1110 0	dd01 101d
0 1011	1 0100	1011	1001 1010	0 0 1011 0	dd01 101d

where *d=don't care.*

Table 4.6: Test 4b (Stuck at 0 Fault Detection on Q Lines)

4.8.3 Stuck at 0 Fault Detection on Q Lines

The four test vectors needed for s-a-0 and s-a-1 fault detection on the Q Lines are given below.

4.8.4 Stuck at 0 and stuck at 1 faults on $Y_{17}Y_{18}...Y_{20}$ Lines.

Note that these Y Lines were not checked during test 1 and 2 on I/P and O/P arrays. Any stuck at fault on the $Y_{17}Y_{18}...Y_{20}$ Lines would result in a stuck at fault on the D Lines and hence be detected by the tests for stuck at faults on the D Lines (Test 4a).

4.8.5 Bridge Fault Detection on D Lines

Again the two tests for the stuck at 0 or 1 fault detection on the D Lines would detect any bridge faults. This is possible because the D Lines are driven in an alternate pattern (0101 or 1010) in both the tests. Any bridge fault on two adjacent lines would result in their having the same logic level and hence appearing as if stuck at. Thus test 4a would also detect D Line bridge faults.

$X_1X_2...X_8$	$Y_{16}Y_{17}...Y_{20}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_{15}Y_{14}...Y_9$	$Z_1Z_2...Z_8$
0 1111	1 0000	1111	10101010	0 0 0000 1	dd01 10d

where *d=don't care.*

Table 4.7: Test 4c (Bridge Fault Detection on Q Lines)

4.8.6 Bridge fault Detection on $Y_{17}Y_{18}...Y_{20}$ Lines

Any bridge fault on the $Y_{17}Y_{18}...Y_{20}$ Lines would result in an apparent bridge fault on the D Lines and be detected by the same tests as for D Line bridge fault detection (Test 4a).

4.8.7 Bridge Fault Detection on Q Lines

The first test used for s-a-0 on the Q Lines (this test checked for $\overline{Q}$ line s-a-0 can be used for bridge fault detection Q lines as well. The method of detection of bridge faults on Q lines is analogous to the one used for bridge fault detection for X Lines (Test 3). The test vector and the fault sensitive responses are indicated below.

Thus seven test vectors detect all the faults associated with the feedback array (except crosspoint faults).**Note that the node coverage in the PLA cell is quite sparse and can be merged with Test 3 Pattern.** The resulting pattern is shown in figure 4.11. When testing the feedback array the X_8 line must be held **1** (to make $\overline{Q_8}$ equal **0**).

4.9 Crosspoint Fault Detection in the I/P and O/P Arrays

Crosspoint faults are the *permanent* presence or absence of a link between the X and Y , Q and Y, Y and D or Y and Z Lines. The crosspoint fault in the NJIT

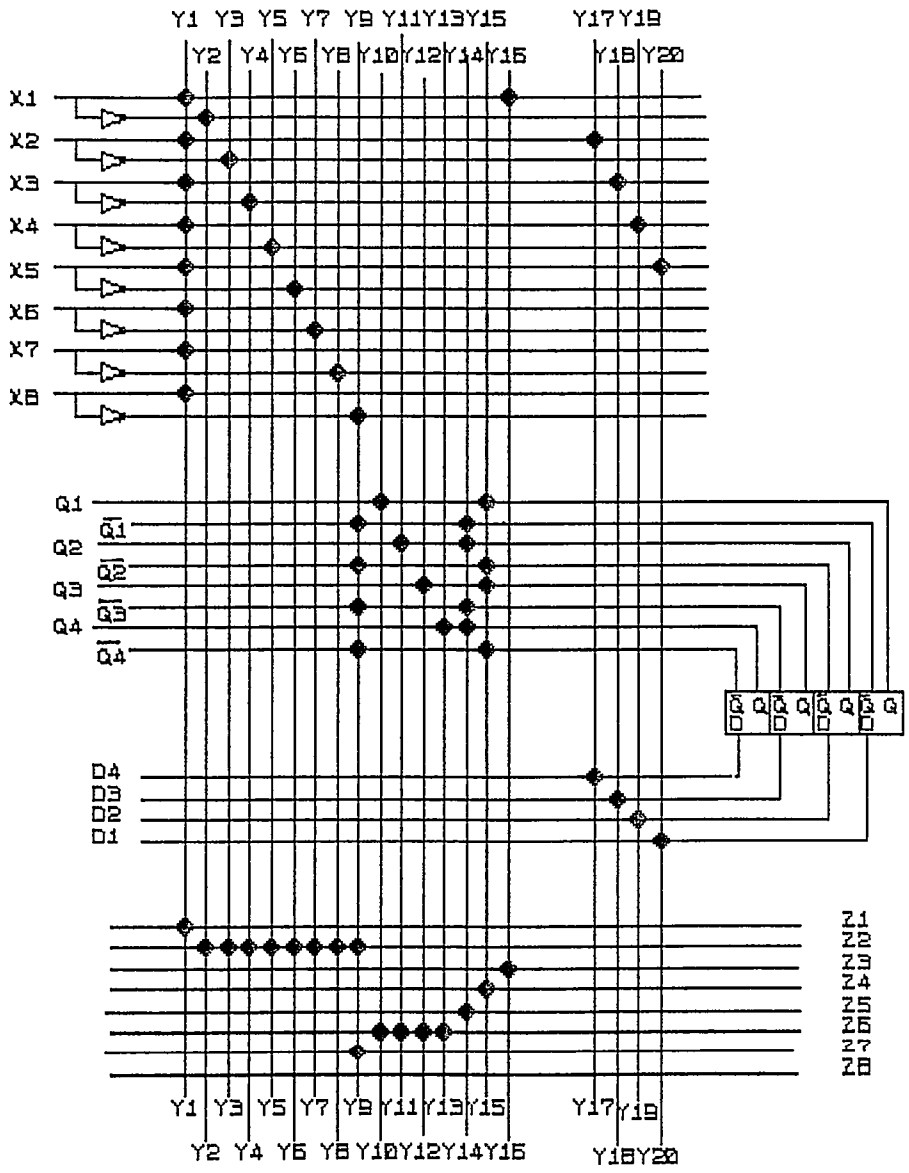


Figure 4.11 PLA Cell pattern for Tests 3 and 4 combined.

PLD can arise because of :

1. A fault in the SRAM bit cell.

2. A fault in the crosspoint transistor.

First discuss a general scheme for detecting crosspoint faults is presented and then the test vectors for crosspoint fault detection in the NJIT PLD are presented.

Consider the PLA cell pattern in figure 4.12 To test for all crosspoint shorts on Z_1 Line all the inputs $X_1X_2...X_8$ are set 1. This results in $Y_1Y_3Y_{15}$ all being **0** and $Y_2Y_4...Y_{16}$ being all **1**. This results in a normal output of **1** on Z_1 Line. Any extra crosspoint link in the O/P array will include a Y Line from the group $Y_2Y_4...Y_{16}$ and cause the output to be **0**. Thus a single vector can test all the crosspoint shorts on each output line (Z Line or Y Line). Further more a checker board pattern in the Output array will allow crosspoint short detection on all the alternate Z Lines simultaneously. Thus only two test vectors could detect all crosspoint shorts in the O/P Array. A similar approach may be followed for the the I/P array.

The detection of crosspoint open faults is more time consuming. To detect a crosspoint open that point must be made the only active path in the product or sum term (analogous to *single activation*). Thus an input of 01111111 would result in a normal O/P of **0** on the Z_1 Line but a crosspoint open between Y_1 *and* Z_1 would result in O/P being **1**. Again a checker board pattern would allow parallel detection on all similar lines and speed up the test.

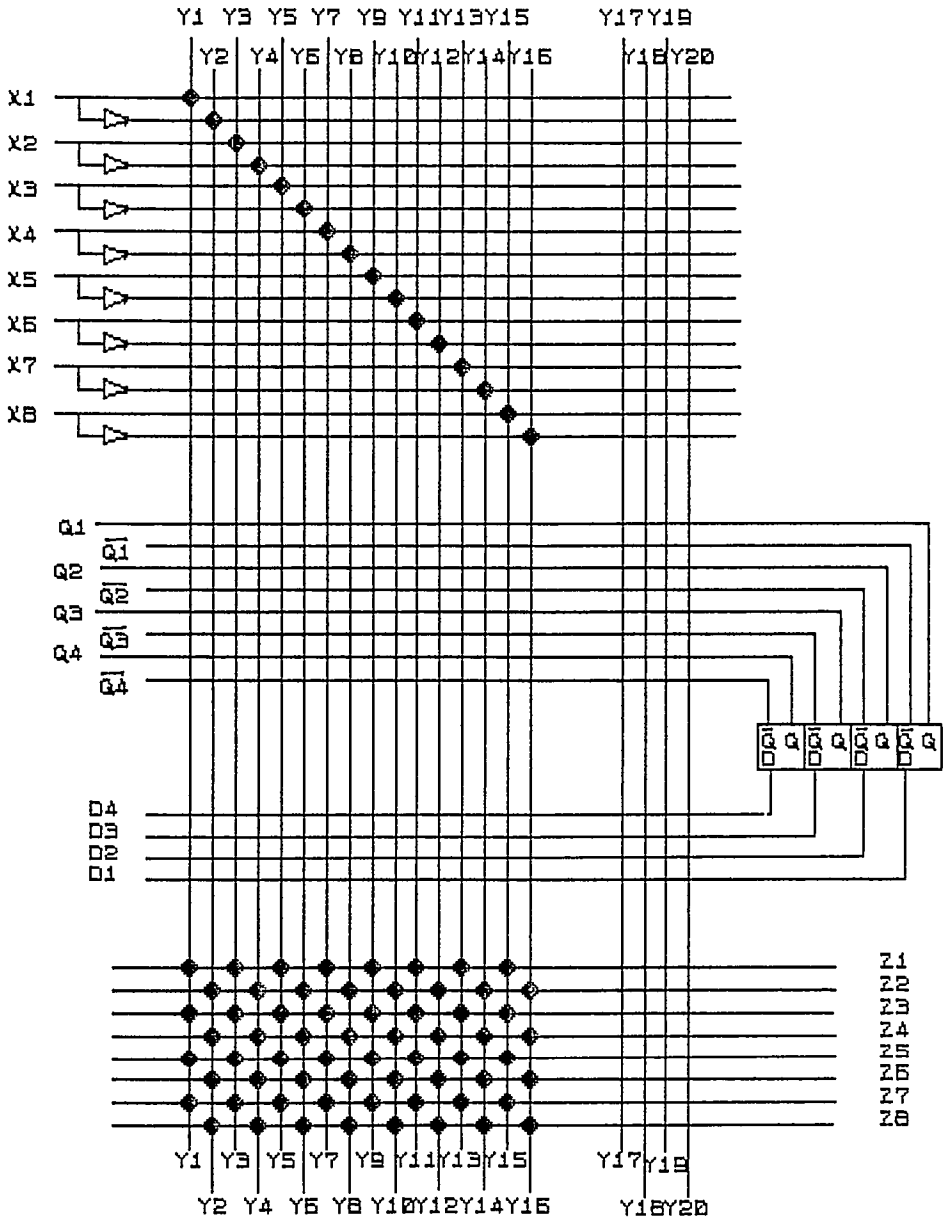


Figure 4.12 General scheme for checking crosspoint faults.

4.10 Crosspoint Fault Detection with EWHMSAI (Test 1) Pattern

The EWHMSAIP is not perfectly symmetric like the checker board pattern and hence it requires more test vectors to detect the crosspoint faults in the I/P Array. However the EWHMSAI pattern can produce Y outputs which are *singly activated.* These outputs can be then used to detect crosspoint open faults in a O/P matrix programmed with a checker board pattern. The EWHMSAI pattern can itself be used to check the I/P matrix for crosspoint faults. Referring back to Test 1a and 1b we find that all the Y Lines are either all **0** or at the most one line is **1** (single activation). **Therefore the interleaved pattern in the O/P matrix can be replaced by a checkerboard pattern right in the beginning.** The resulting pattern is shown in figure 4.13 and it may be called EWHMSAI-Checkerboard pattern. Now four alternate outputs out of the eight Z Lines will have same logic value but adjacent output lines would have different values making it still possible to detect s-a-0, s-a-1 and bridge faults described in Tests 1 and 2. *This highlights the elegance of EWHMSAI pattern.*

4.10.1 Crosspoint Short Detection in the I/P Array

The test vectors for detecting the crosspoint shorts by the scheme discussed above are now presented.

Note that the first eight vectors have been used in I/P array s-a-1 test too. Each of the 16 test vectors above checks for crosspoint shorts in the Y Line which is at **1**. The above test vectors are applied again after programming the second EWHMSAI-Checkerboard pattern 2 (Test 2).

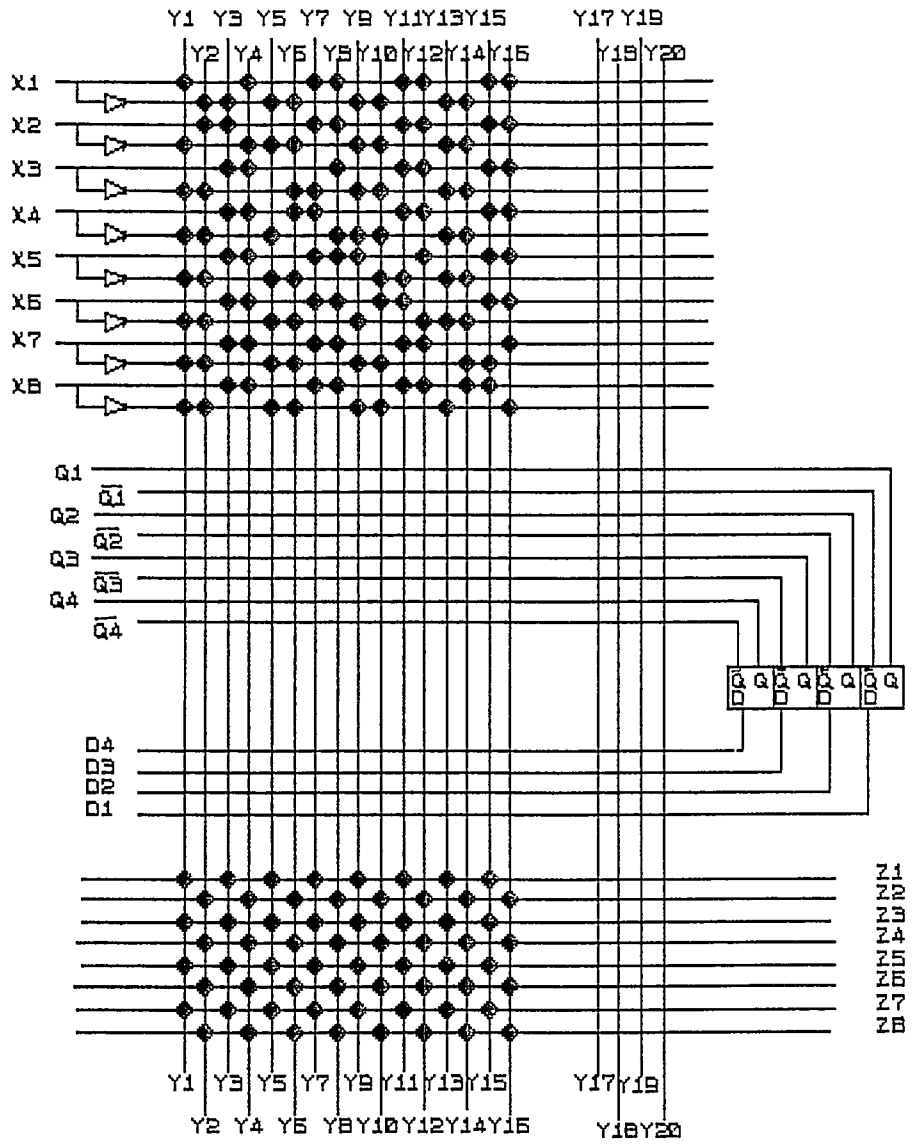


Figure 4.13 EWHMSAI-Checkerboard Pattern 1 for INPUT and OUTPUT array cross point faults.

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
0111 1111	1000 0000 0000 0000	0101 0101
1011 1111	0100 0000 0000 0000	1010 1010
1101 1111	0000 1000 0000 0000	0101 0101
1110 1111	0000 0100 0000 0000	1010 1010
1111 0111	0000 0000 1000 0000	0101 0101
1111 1011	0000 0000 0100 0000	1010 1010
1111 1101	0000 0000 0000 1000	0101 0101
1111 1110	0000 0000 0000 0100	1010 1010
1000 0000	0010 0000 0000 0000	0101 0101
0100 0000	0001 0000 0000 0000	1010 1010
1010 0000	0000 0010 0000 0000	0101 0101
1001 0000	0000 0001 0000 0000	1010 1010
1000 1000	0000 0000 0010 0000	0101 0101
1000 0100	0000 0000 0001 0000	1010 1010
1000 0010	0000 0000 0000 0010	0101 0101
1000 0001	0000 0000 0000 0001	1010 1010

Table 4.8: Test 1c (Crosspoint ShortDetection in the I/P Array)

4.10.2 Crosspoint Open Detection in the I/P Array

The test vectors and the responses on the Y and Z lines for crosspoint open fault detection in the I/P array are given below.

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
Y_1 Line		
1111 1111	0000 0000 0000 0000	1111 1111
0011 1111	0000 0000 0000 0000	1111 1111
0101 1111	0000 0000 0000 0000	1111 1111
0110 1111	0000 0000 0000 0000	1111 1111
0111 0111	0000 0000 0000 0000	1111 1111
0111 1011	0000 0000 0000 0000	1111 1111
0111 1101	0000 0000 0000 0000	1111 1111
0111 1110	0000 0000 0000 0000	1111 1111
Y_2 Line		
0011 1111	0000 0000 0000 0000	1111 1111
1111 1111	0000 0000 0000 0000	1111 1111
1001 1111	0000 0000 0000 0000	1111 1111
1010 1111	0000 0000 0000 0000	1111 1111
1011 0111	0000 0000 0000 0000	1111 1111
1011 1011	0000 0000 0000 0000	1111 1111
1011 1101	0000 0000 0000 0000	1111 1111
1011 1110	0000 0000 0000 0000	1111 1111
Y_3 Line		
0000 0000	0000 0000 0000 0000	1111 1111
1100 0000	0000 0000 0000 0000	1111 1111
1010 0000	0000 0000 0000 0000	1111 1111
1001 0000	0000 0000 0000 0000	1111 1111
1000 1000	0000 0000 0000 0000	1111 1111
1000 0100	0000 0000 0000 0000	1111 1111
1000 0010	0000 0000 0000 0000	1111 1111
1000 0001	0000 0000 0000 0000	1111 1111

Table 4.9: Test 1d (Crosspoint Open Detectionin the I/P Array)

Test 1d contd. (Crosspoint Open Detection in the I/P Array)

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
Y_4 Line		
1100 0000	0000 0000 0000 0000	1111 1111
0000 0000	0000 0000 0000 0000	1111 1111
0110 0000	0000 0000 0000 0000	1111 1111
0101 0000	0000 0000 0000 0000	1111 1111
0100 1000	0000 0000 0000 0000	1111 1111
0100 0100	0000 0000 0000 0000	1111 1111
0100 0010	0000 0000 0000 0000	1111 1111
0100 0001	0000 0000 0000 0000	1111 1111
.....		
.....		
.....		
Y_{15} Line		
1000 0010	0000 0000 0000 0000	1111 1111
0100 0010	0000 0000 0000 0000	1111 1111
0010 0010	0000 0000 0000 0000	1111 1111
0001 0010	0000 0000 0000 0000	1111 1111
0000 1010	0000 0000 0000 0000	1111 1111
0000 0110	0000 0000 0000 0000	1111 1111
0000 0000	0000 0000 0000 0000	1111 1111
0000 0011	0000 0000 0000 0000	1111 1111
Y_{16} Line		
1000 0001	0000 0000 0000 0000	1111 1111
0100 0001	0000 0000 0000 0000	1111 1111
0010 0001	0000 0000 0000 0000	1111 1111
0001 0001	0000 0000 0000 0000	1111 1111
0000 1001	0000 0000 0000 0000	1111 1111
0000 0101	0000 0000 0000 0000	1111 1111
0000 0011	0000 0000 0000 0000	1111 1111
0000 0000	0000 0000 0000 0000	1111 1111

Note that the Y Lines have many common tests. As shown in the case of Y_1 Line test all the Y Lines with their output shown as 0 are being checked for a

crosspoint open. Each test vector usually checks for two crosspoint open faults and the vectors 0000 0000 and 1111 1111 check for eight faults each. Since there are $8 * 8 = 256$ nodes programmed in the EWHMSAI pattern 1 the total tests required for crosspoint open fault detection are $(256/2) - 2 * 7 = 114$.

The above test vectors are applied again after programming the second EWHMSAI-Checkerboard pattern of Test 2, figure 4.14.

4.10.3 Crosspoint Short Fault Detection in the O/P Array

The crosspoint fault detection test in the O/P array is much quicker. This is because the O/P matrix is programmed with regular checkerboard which allows simultaneous testing of half (4) Z Lines. To test for crosspoint shorts between a Y Line and Z Lines the Y Line is made 1. All other Y Lines remain 0. A crosspoint short between this Y Line and any Z Line will then make the normally 1 output of that Z Line go 0.

The test vectors for the O/P Array crosspoint short are the same as the I/P Array crosspoint short. These are repeated below.

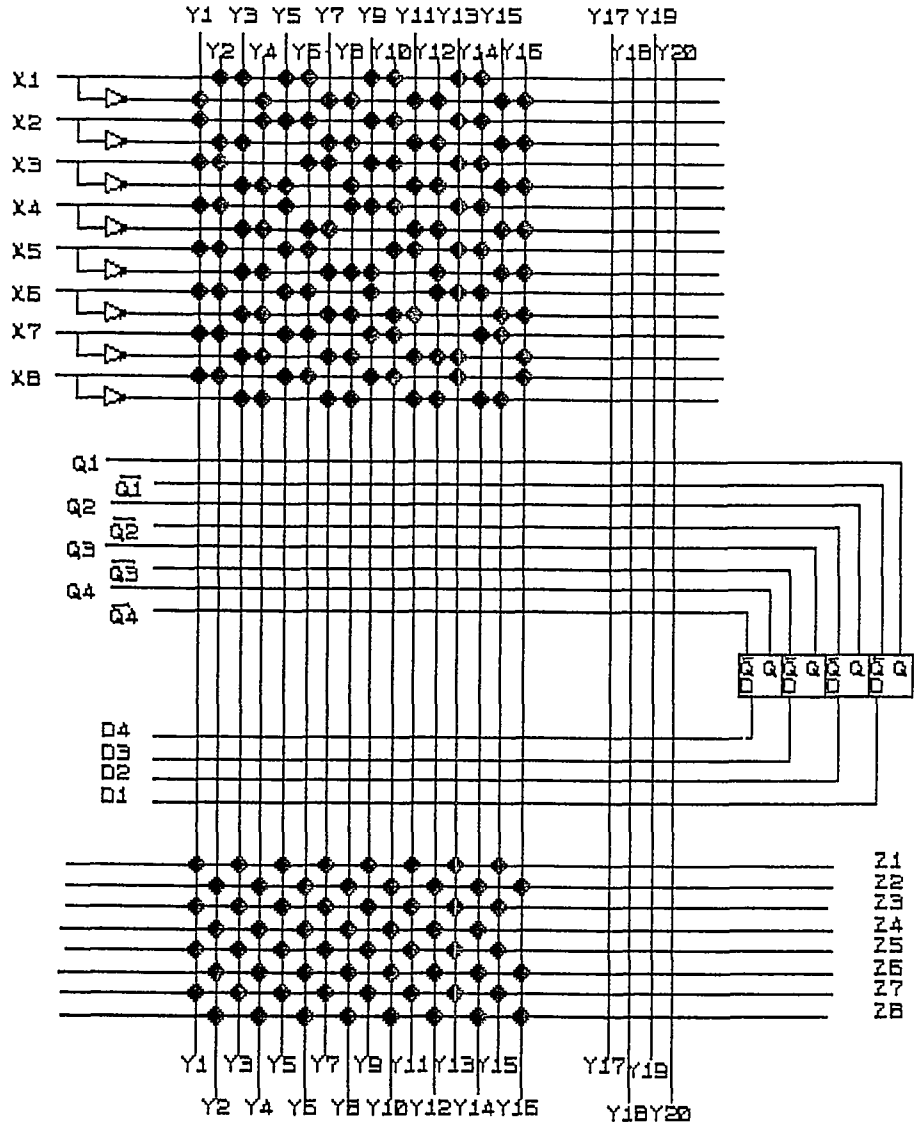


Figure 4.14 EWHMSAI-Checkerboard Pattern 2 for checking INPUT and OUTPUT array crosspoint faults.

O/P Array crosspoint fault test vectors

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
0111 1111	1000 0000 0000 0000	0101 0101
1011 1111	0100 0000 0000 0000	1010 1010
1101 1111	0000 1000 0000 0000	0101 0101
1110 1111	0000 0100 0000 0000	1010 1010
1111 0111	0000 0000 1000 0000	0101 0101
1111 1011	0000 0000 0100 0000	1010 1010
1111 1101	0000 0000 0000 1000	0101 0101
1111 1110	0000 0000 0000 0100	1010 1010
1000 0000	0010 0000 0000 0000	0101 0101
0100 0000	0001 0000 0000 0000	1010 1010
1010 0000	0000 0000 0000 0000	0101 0101
1001 0000	0000 0000 0000 0000	1010 1010
1000 1000	0000 0000 0000 0000	0101 0101
1000 0100	0000 0000 0000 0000	1010 1010
1000 0010	0000 0000 0000 0000	0101 0101
1000 0001	0000 0000 0000 0000	1010 1010

Note that the first eight vectors have been used in I/P array s-a-1 test too. Each of the 16 test vectors above checks for crosspoint shorts in the Y Line which is at 1.

The above test vectors are applied again after programming the second EWHMSAI-Checkerboard pattern of Test 2.

4.10.4 Crosspoint Open Fault Detection in the O/P Array

Again the checkerboard symmetry of the O/P array allows a quicker testing of the crosspoint open faults. The sixteen tests for the O/P array crosspoint short faults tested for the crosspoint shorts on the Y Lines which were 1 and not connected as input to the O/P Lines. They also tested for crosspoint opens on the Y Lines which were 1 and connected as input terms to the O/P Lines. This was possible because

of *single activation* in the Y array.

4.11 Test 5 (Crosspoint Fault Detection in the D Array)

This section discusses the scheme to detect crosspoint faults in the D Array. The testing scheme is similar to the one used for I/P and O/P arrays. The test vector is inverted through the I/P array. It is then applied to the D Array which has been programmed with the $Y_1 ... Y_4$ columns of the EWHMSAI Pattern of the Test 1. Figure 4.15 and 4.16 show the PLA cell programming and the circuit realization.

The output of the flipflops (Q Lines) are feed to the $Y_{17}Y_{20}$ Lines with a checkerboard pattern. The output from this matrix is fed to the O/P array where after an inversion it is available on the Z Lines. This implementation is very similar to the EWHMSAI-Checkerboard pattern with the minor variation of having inverter stages at the I/P and O/P and a clock cycle delay through the D flipflops. This test would check for the following.

1. Crosspoint open and short faults in the D array.
2. Crosspoint open and short faults in the $Q_1 ... \overline{Q_7} - Y_{17}Y_{20}$ array.

4.11.1 Crosspoint Short Fault Detection in the D Array

The test vectors for this test are analogous to those for the I/P array crosspoint short detection and are given below.

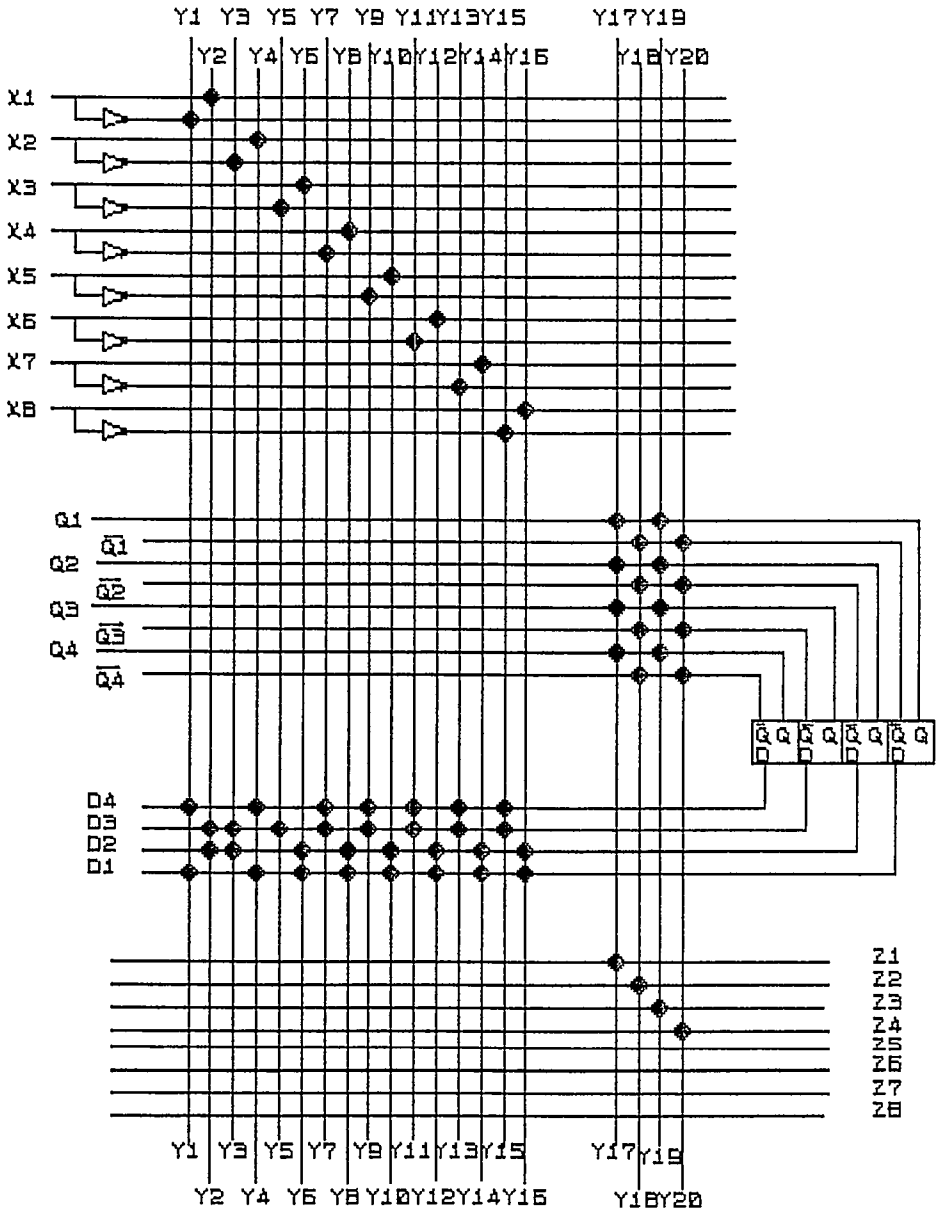


Figure 4.15 D array crosspoint check pattern 1.

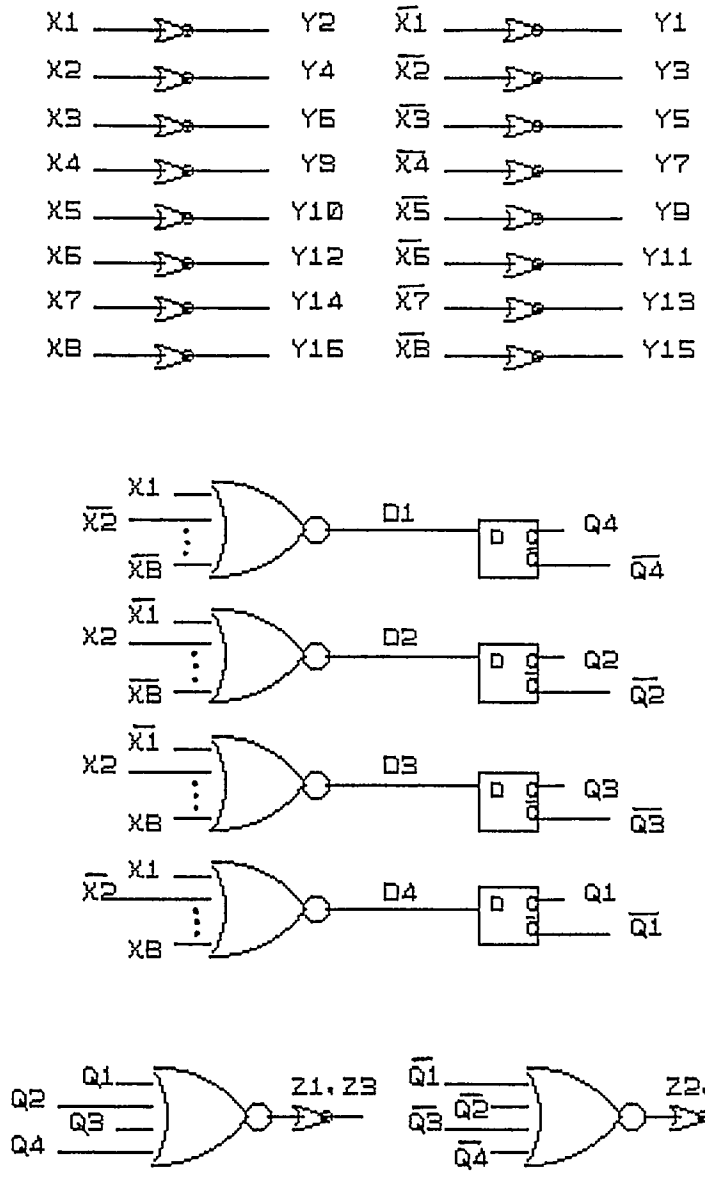


Figure 4.16 Circuit realization for D array crosspoint check pattern 1.

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$D_4D_3...D_1$	$Q_1\overline{Q}_1...Q_4\overline{Q}_4$	$Y_{17}Y_{18}...Y_{20}$	$Z_1Z_2...Z_8$
0111 1111	1001 0101 0101 0101	1000	1001 0101	0000	1111
1011 1111	0110 0101 0101 0101	0100	0110 0101	0000	1111
1000 0000	0110 1010 1010 1010	0001	0101 1001	0000	1111
0100 0000	1001 0101 0101 0101	0001	0101 0110	0000	1111

Table 4.10: Test 5a (Crosspoint Short Fault Detection in the D Array)

4.11.2 Crosspoint Open Fault Detection in D Array

The test vectors for crosspoint open faults in the D Array are given below. Note their similarity with the I/P array crosspoint open.

4.11.3 Crosspoint Short Fault Detection in $Y_{17}...Y_{20}$-X Array

The test for detection of crosspoint open for the D array results in all D inputs to be normally 0. This would check for any crosspoints in the $Y_{17}...Y_{20}$ Array. Note that only four tests out of the 32 would have been sufficient.

4.11.4 Crosspoint Open Fault Detection in the $Y_{17}...Y_{20}$ Array

The crosspoint short faults check for the D array resulted in *singly activated* D array. This would also check for crosspoint open faults in the $Y_{17}...Y_{20}$ Array. The checkerboard pattern allows testing with only two vectors.

All these test vectors are applied again to the complimentary pattern of figure 4.15 which is shown in figure 4.17.

$X_1X_2...X_8$	$Y_1Y_2...Y_{16}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_{17}Y_{18}...Y_{20}$	$Z_1Z_2...Z_8$
D_4 Line					
1111 1111	1010 1010 1010 1010	0000	0101 0101	0101	1010
0011 1111	0101 1010 1010 1010	0000	0101 0101	0101	1010
0101 1111	0110 0110 1010 1010	0000	0101 0101	0101	1010
0110 1111	0110 1001 1010 1010	0000	0101 0101	0101	1010
0111 0111	0110 1010 0110 1010	0000	0101 0101	0101	1010
0111 1011	0110 1010 1001 1010	0000	0101 0101	0101	1010
0111 1101	0110 1010 1010 0110	0000	0101 0101	0101	1010
0111 1110	0110 1010 1010 1001	0000	0101 0101	0101	1010
D_3 Line					
0011 1111	0101 1010 1010 1010	0000	0101 0101	0101	1010
1111 1111	1010 1010 1010 1010	0000	0101 0101	0101	1010
1001 1111	1001 0110 1010 1010	0000	0101 0101	0101	1010
1010 1111	1001 1001 1010 1010	0000	0101 0101	0101	1010
1011 0111	1001 1010 0110 1010	0000	0101 0101	0101	1010
1011 1011	1001 1010 1001 1010	0000	0101 0101	0101	1010
1011 1101	1001 1010 1010 0110	0000	0101 0101	0101	1010
1011 1110	1001 1010 1010 1001	0000	0101 0101	0101	1010
D_2 Line					
0000 0000	0101 0101 0101 0101	0000	0101 0101	0101	1010
1100 0000	1010 0101 0101 0101	0000	0101 0101	0101	1010
1010 0000	1001 1001 0101 0101	0000	0101 0101	0101	1010
1001 0000	1001 0110 0101 0101	0000	0101 0101	0101	1010
1000 1000	1001 0101 1001 0101	0000	0101 0101	0101	1010
1000 0100	1001 0101 0110 0101	0000	0101 0101	0101	1010
1000 0010	1001 0101 0101 1001	0000	0101 0101	0101	1010
1000 0001	1001 0101 0101 0101	0000	0101 0101	0101	1010
D_1 Line					
1100 0000	1010 0101 0101 0101	0000	0101 0101	0101	1010
0000 0000	0101 0101 0101 0101	0000	0101 0101	0101	1010
0110 0000	0110 1001 0101 0101	0000	0101 0101	0101	1010
0101 0000	0110 0110 0101 0101	0000	0101 0101	0101	1010
0100 1000	0110 0101 1001 0101	0000	0101 0101	0101	1010
0100 0100	0110 0101 0110 0101	0000	0101 0101	0101	1010
0100 0010	0110 0101 0101 1001	0000	0101 0101	0101	1010
0100 0001	0110 0101 0101 0110	0000	0101 0101	0101	1010

Table 4.11: Test 5b (Crosspoint Open Fault Detection in D Array)

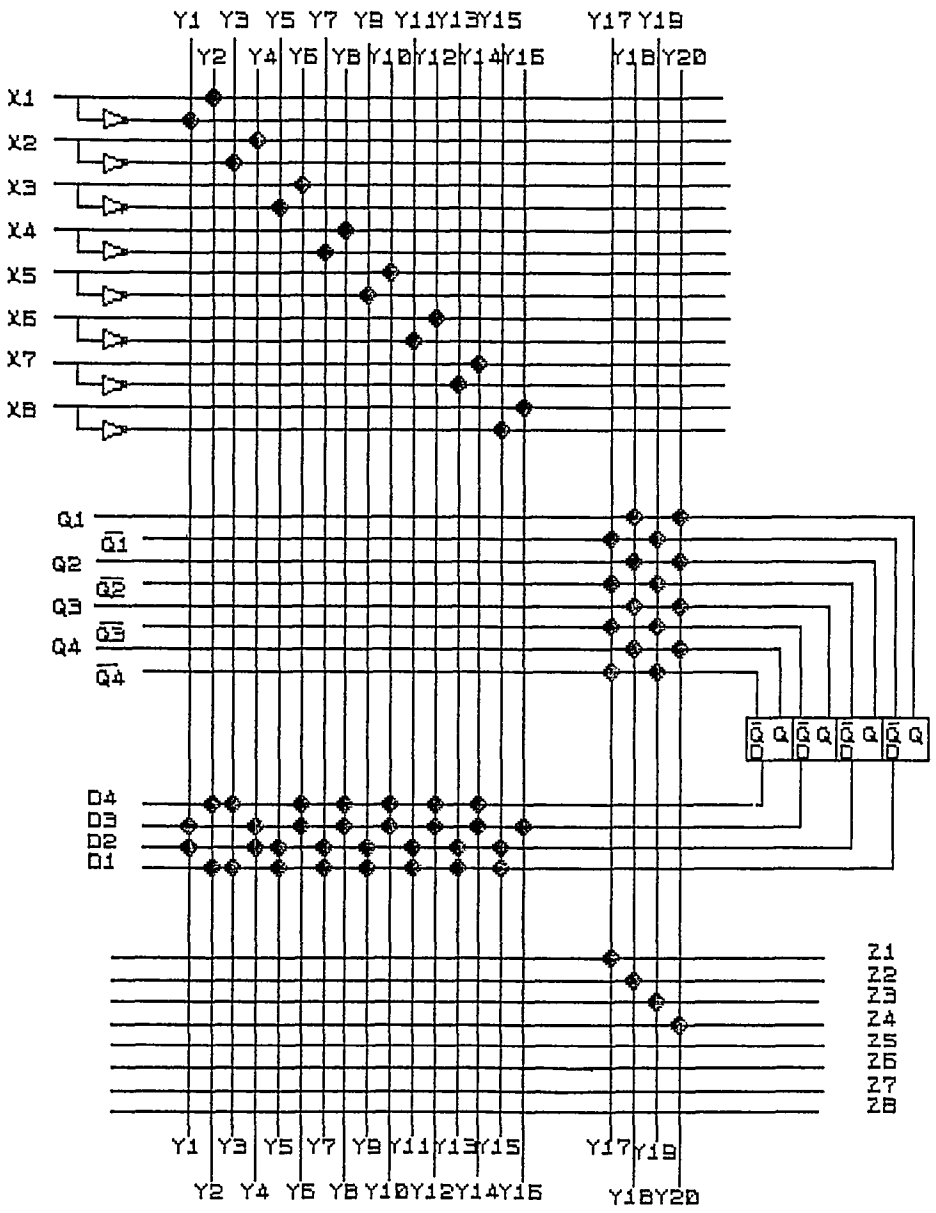


Figure 4.17 D array crosspoint check pattern 2.

$X_1X_2...X_8$	$Y_{17}Y_{18}...Y_{20}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
0111 1111	1000	0101	0110 0110	0101 0101 0101 0101	0000 0000
1011 1111	0100	1010	1001 1001	1010 1010 1010 1010	0000 0000
1000 0000	0010	0101	0110 0110	0101 0101 0101 0101	0000 0000
0100 0000	0001	1010	1001 1001	1010 1010 1010 1010	0000 0000

Table 4.12: Test 5c (Crosspoint Short Fault Detection in the $Y_{17}...Y_{20} - X$ Array)

4.12 Test 6 (Crosspoint Faults in the Q Array)

This test would check for the following.

1. Crosspoint open and short faults in the Q array.
2. Crosspoint open and short faults in the $X - Y_{17}...Y_{20}$ array.
3. Crosspoint open and short faults in the $D_1...D_4$-$Y_{17}Y_{20}$ array.

The $X - Y_{17}...Y_{20}$ array is programmed as a subset of the EWHMSAIP. The $D_1...D_4$-$Y_{17}Y_{20}$ array is programmed as a checkerboard pattern while a new type of pattern is programmed in the Q Array. Figures 4.18 and 4.19 show the PLA cell pattern and the resulting circuit implementation.

4.12.1 Crosspoint Short Fault Detection in the $Y_{17}...Y_{20} - X$ Array

The $Y_{17}...Y_{20} - X$ Array is programmed with a subset of the EWHMSAIP used for the Test 1. This will allow a testing of all crosspoint short and open faults with the tests given below.

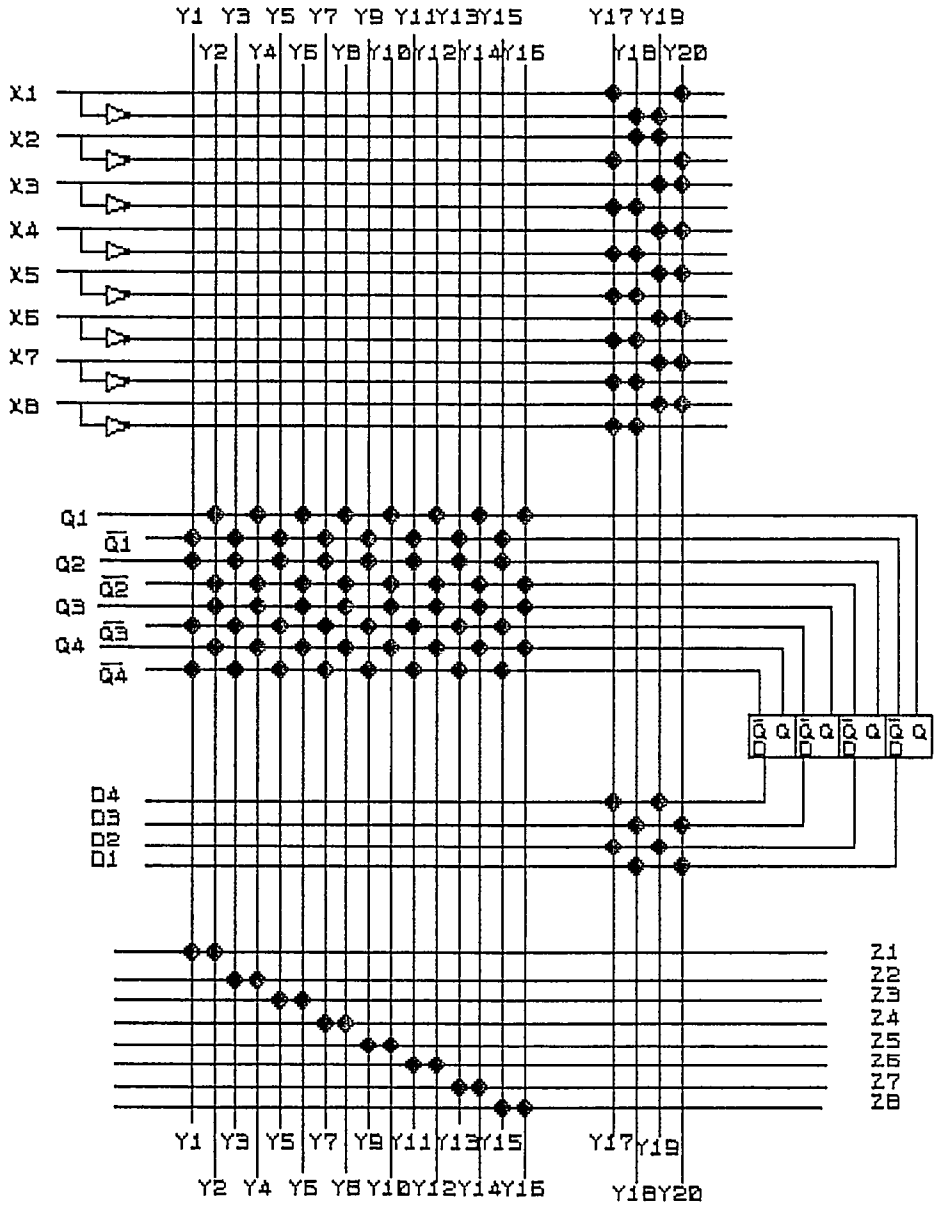


Figure 4.18 Q array crosspoint check pattern 1.

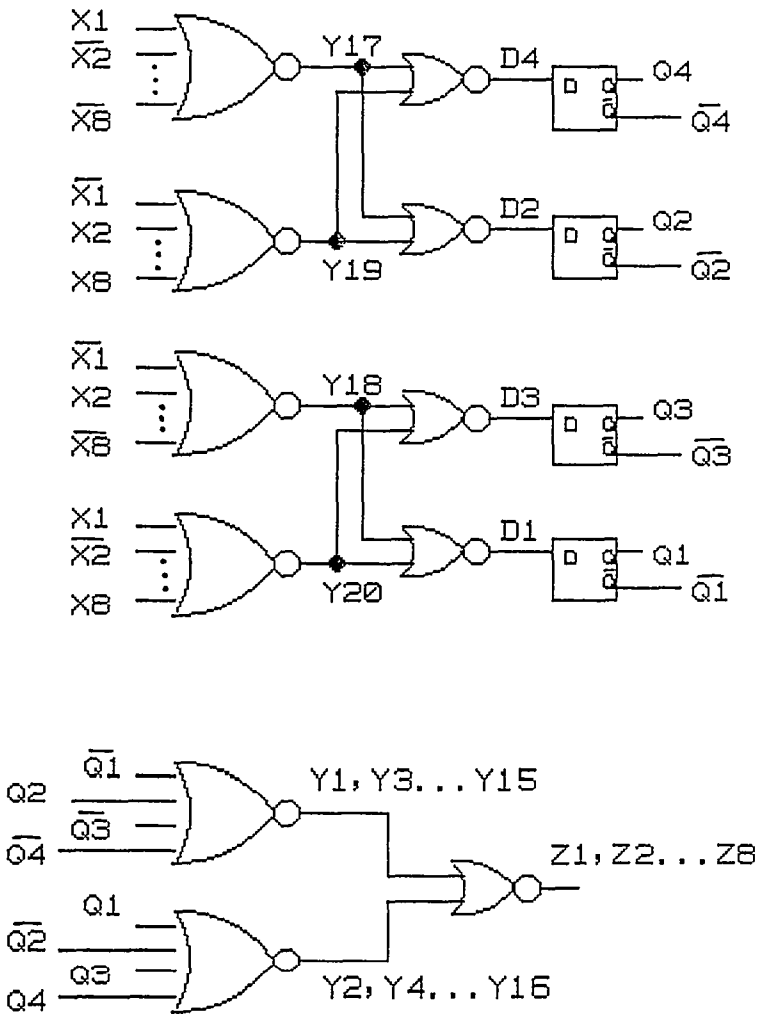


Figure 4.19 Circuit realization for Q array crosspoint check pattern 1.

4.12.2 Crosspoint Open Fault Detection in the $Y_{17}...Y_{20} - X$ Array

This test proceeds similar to the crosspoint open fault check for the I/P Array. The test vectors are given below.

All these test are repeated again with the complimentary pattern of figure 4.18 programmed. This complimentary pattern is showed in figure 4.20. The repeated tests may be called tests 6a and 6b respectively.

4.13 Conclusion

The previous sections showed how all the faults in the PLA Cell could be tested from the external pins. The approach allowed different circuit implementations to be programmed in the PLA resulting in a combined 100% controlability and 100% observability. The table below summarizes all the tests.

The above discussion shows the design of a highly optimal test pattern for PLA fault check. These EWHMSAI patterns allow all but input line bridge fault detection in the combinational section of the PLA. The feedback circuit is checked by programming other patterns. The total test patterns for the model PLA were 7. The total number of test vectors used were $2 + 2 * (192) = 470$ for the total of 2960 nodes, and 56 X,Y,Z,D and Q lines.

For PLAs that are of different size that is with smaller or larger input, output and feedback sizes similar patterns and tests can be derived.

$X_1X_2...X_8$	$Y_{17}Y_{18}...Y_{20}$	$D_4D_3...D_1$	$Q_1\overline{Q_1}...Q_4\overline{Q_4}$	$Y_1Y_2...Y_{16}$	$Z_1Z_2...Z_8$
Y_{17} Line					
1111 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
0011 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
0101 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
0110 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
0111 0111	0000	1111	1010 1010	0000 0000 0000	1111 1111
0111 1011	0000	1111	1010 1010	0000 0000 0000	1111 1111
0111 1101	0000	1111	1010 1010	0000 0000 0000	1111 1111
0111 1110	0000	1111	1010 1010	0000 0000 0000	1111 1111
Y_{18} Line					
0011 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
1111 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
1001 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
1010 1111	0000	1111	1010 1010	0000 0000 0000	1111 1111
1011 0111	0000	1111	1010 1010	0000 0000 0000	1111 1111
1011 1011	0000	1111	1010 1010	0000 0000 0000	1111 1111
1011 1101	0000	1111	1010 1010	0000 0000 0000	1111 1111
1011 1110	0000	1111	1010 1010	0000 0000 0000	1111 1111
Y_{19} Line					
0000 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
1100 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
1010 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
1001 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
1000 1000	0000	1111	1010 1010	0000 0000 0000	1111 1111
1000 0100	0000	1111	1010 1010	0000 0000 0000	1111 1111
1000 0010	0000	1111	1010 1010	0000 0000 0000	1111 1111
1000 0001	0000	1111	1010 1010	0000 0000 0000	1111 1111
Y_{20} Line					
1100 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
0000 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
0110 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
0101 0000	0000	1111	1010 1010	0000 0000 0000	1111 1111
0100 1000	0000	1111	1010 1010	0000 0000 0000	1111 1111
0100 0100	0000	1111	1010 1010	0000 0000 0000	1111 1111
0100 0010	0000	1111	1010 1010	0000 0000 0000	1111 1111
0100 0001	0000	1111	1010 1010	0000 0000 0000	1111 1111

Table 4.13: Test 5d (Crosspoint Open Fault Detection in the $Y_{17}...Y_{20} - X$ Array)

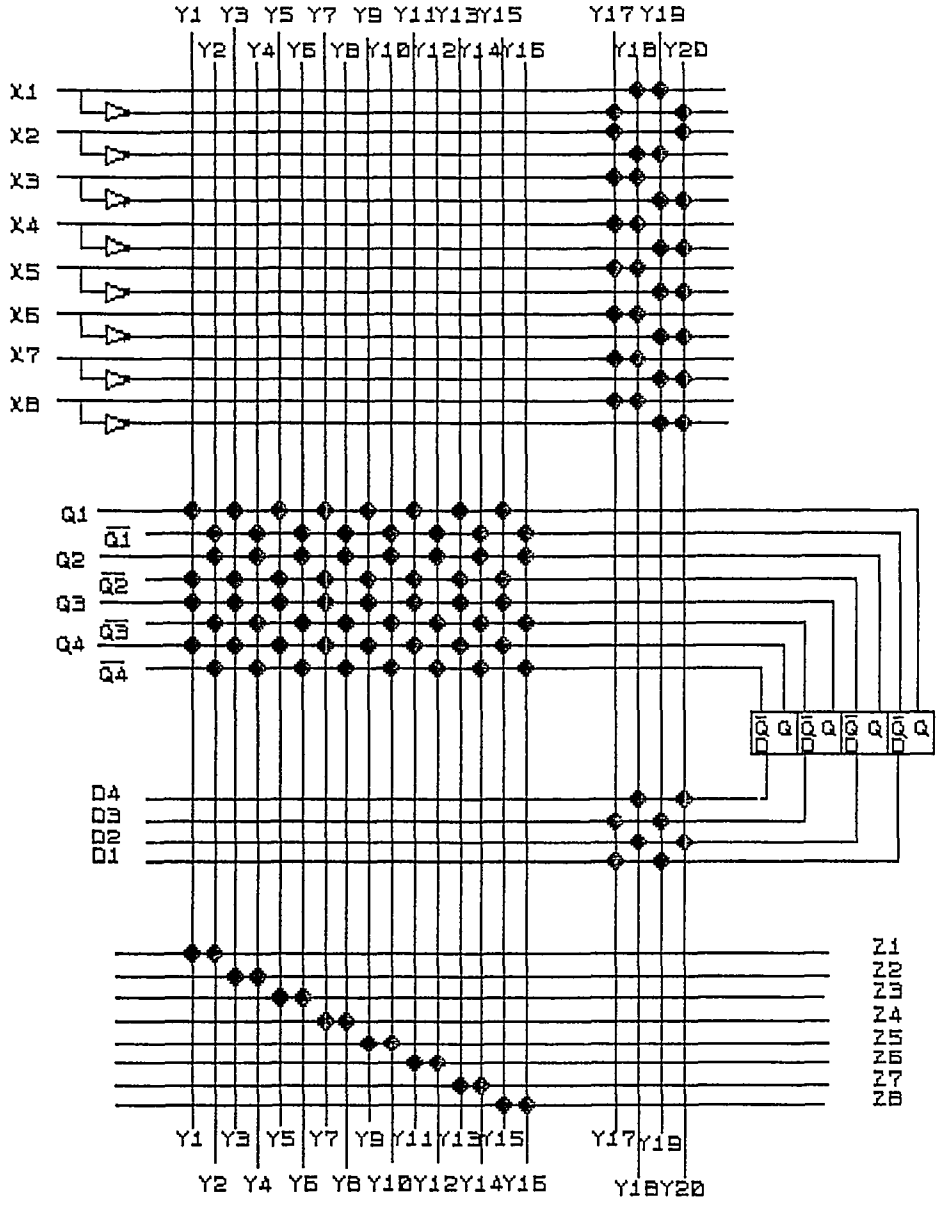


Figure 4.20 Q array crosspoint check pattern 2.

PLA Section	s-a-0	s-a-1	Bridge	Crosspoint Short	Crosspoint Open
X Lines	1a	1b	3	N/A	N/A
Y Lines	1b	1a	1b	N/A	N/A
Z Lines	1a	1b	1b	N/A	N/A
D Lines	4a	4a	4a	N/A	N/A
Q Lines	4b,4c	4a	4c	N/A	N/A
I/P Matrix	N/A	N/A	N/A	1c,2a	1d,2b
O/P Matrix	N/A	N/A	N/A	1c,2a	1d,2b
F/B Matrix	N/A	N/A	N/A	5a,6a	5b,6b

Table 4.14: Summary of Test Sequences

REFERENCES

[1] Nickel, V. V., *VLSI-the inadequacy of the stuck-at-fault model*, Proc. IEEE Test Conf., 378-381 (1980).

[2] Friedman, A. D., *Diagnosis of short circuit faults in combinational circuits*, IEEE Trans. Computers, 746-752 (July 1974).

[3] Mei, K. C. Y., *Bridging and stuck-at faults*, IEEE Trans. Computers, 720-727 (July 1974).

[4] Karpovosky, M. and S. Y. H. Su, *Detecting bridging and stuck-at faults at the input and output pins of standard digital components*, Proc. 17th Design Automation Conf., 494-505 (1980).

[5] Armstrong, D. B., *On finding a nearly minimal set of fault detection tests for combinational logic nets*, IEEE Trans. Electron. Comput., 66-73 (Feb. 1966).

[6] Chiang, A. C., I. S. Reed and A. V. Banes, *Path sensitization, partial boolean difference and automated fault diagnosis*, IEEE Trans. Computers, 189-195 (Feb. 1972).

[7] Kohavi, Zvi, *Switching and Finite Automata Theory*, Chapter 6, McGraw-Hill (1970).

[8] Carr, W. N. and J. P. Mize, *MOS/LSI Design and Application*, Chapter 8, Texas Instruments Elec. Series, McGraw-Hill (1972).

[9] Dias, F. J. O., *Fault masking in combinational logic circuits*, IEEE Trans. Comput., 476-482 (May 1975).

[10] Ostapko, D. L. and S. J. Hong, *Fault analysis and test generation for programmable logic arrays*, IEEE Trans. Comput., 617-626 (Sept. 1979).

[11] Cha, C. W., *A testing strategy for PLAs*, Proc. 15th Design Automation Conf., 326-334 (1978).

[12] Pradhan, D. K. and K. Son, *The effects of untestable faults in PLAs and a design for testability*, Proc. IEEE Test Conf., 359-367 (1980).

[13] Fujiwara, H. and K. Kinoshita, *A design of programmable logic arrays with universal tests*, IEEE Trans. Comput., 823-828 (Nov. 1981).

[14] Ramanathan K. S. and N. N. Biswas, *A design for complete testability of programmable logic arrays*, Proc. IEEE Test Conf., 67-74 (1982).

[15] Parvathala, P. K., *A Dynamic CMOS Programmable Logic Device*, Master's Thesis at New Jersey Institute of Tech., (May 1989).

[16] Integrated Measurement Systems, *Logic Master Series System Reference Mannual*, Version 1.1, Part No. 900-0002-004 (Aug. 1986).

[17] Integrated Measurement Systems, *Logic Master Series Operator's Mannual*, Version 1.1, Part No. 910-0002-006 (Aug. 1986).

Appendix A

Test Equipment for NJIT PLD

The equipment used for testing the NJIT PLD is the *Logic Master System* from INTEGRATED MEASUREMENT SYSTEMS. It consists of the following three units:

1. IMS 1000 Mainframe System (with 16 input and 16 output channels).
2. VS 1000 Verification Station.
3. A Terminal which may be connected through a mainframe like VAX or APOLLO or a dumb terminal (VT100 or higher).

The IMS 1000 mainframe is the heart of the test system. It has various interfaces for connecting as a GPIB compatible equipment, or for connecting to a mainframe (VAX or APOLLO), or to a IBM PC or compatible through a RS232 port and a Screen Link Program. A dumb VT100 (or higher) terminal may be connected instead. The host computer or the terminal is used to control the IMS 1000. The host computer can command the IMS 1000 in two ways:

1. By use of Screen Interface in which different screens (Configuration, Resource Allocation, Operating Conditions and Pattern Control) are used to execute the test experiments.
2. Command Language Interface permits the testing to proceed as directed by a written program. There are about 50 Logic Master commands. The commands may be used as parameters of the standard I/O commands in any programming language.

The IMS 1000 mainframe connects to different modules through POD Assemblies to the VS1000 Verification Station. The IMS 1000 can have different modules such as Pattern Generation, Data Acquisition and Programmable Power Supply be connected. The device to be tested is wired on the VS 1000 Verification station. The IMS 1000 transmits test vectors and acquires data from the D.U.T. through the POD assemblies. The setup for this thesis used an IBM compatible PC running a VT100 emulation software.

Figures A.1 to A.4 show the different screens as used in testing of a dual CMOS flip-flop CD4013. The configuration screen (Figure A.1) appears at power on and is a summary of the system set up. It displays the different modules available and the width (channels or signal lines) and memory depth (signal sample storage capacity) for each channel.

The resource assignment (Figure A.2) screen is used next to assign data output channels to the device under test (Force channels), power to the D.U.T. (Power channel), timing and clock signals (Timing channels) and data input channels. The data input channels could be simply *acquire* or *compare*. This resource assignment is then used to wire the device under test.

The third screen (Figure A.3) to be used is the Operating Conditions screen which allows the tunning of various parameters such as display radix (binary, decimal or hex), active high or low setup, logic low and high voltage threshold setup, signal formats (RZ, R1, NRZ, NR1, RI etc.), voltage threshold for input data samples and sampling time with respect to the system clock. It also allows adding short circuit current limits to the power supply and its disabled state definition.

The Pattern Control Screen (Figure A.4) allows entering of the test vectors the expected data and instructions such as *if error goto test n* etc. As shown in the figure A.4 after running the test three errors were found in the response as indicated under the error column. Note that the $\overline{Q_2}$ is stuck-at-1.

A further description about the Logic Master System is given in the Logic Master Series System Reference Manual and the Operator's Manual.

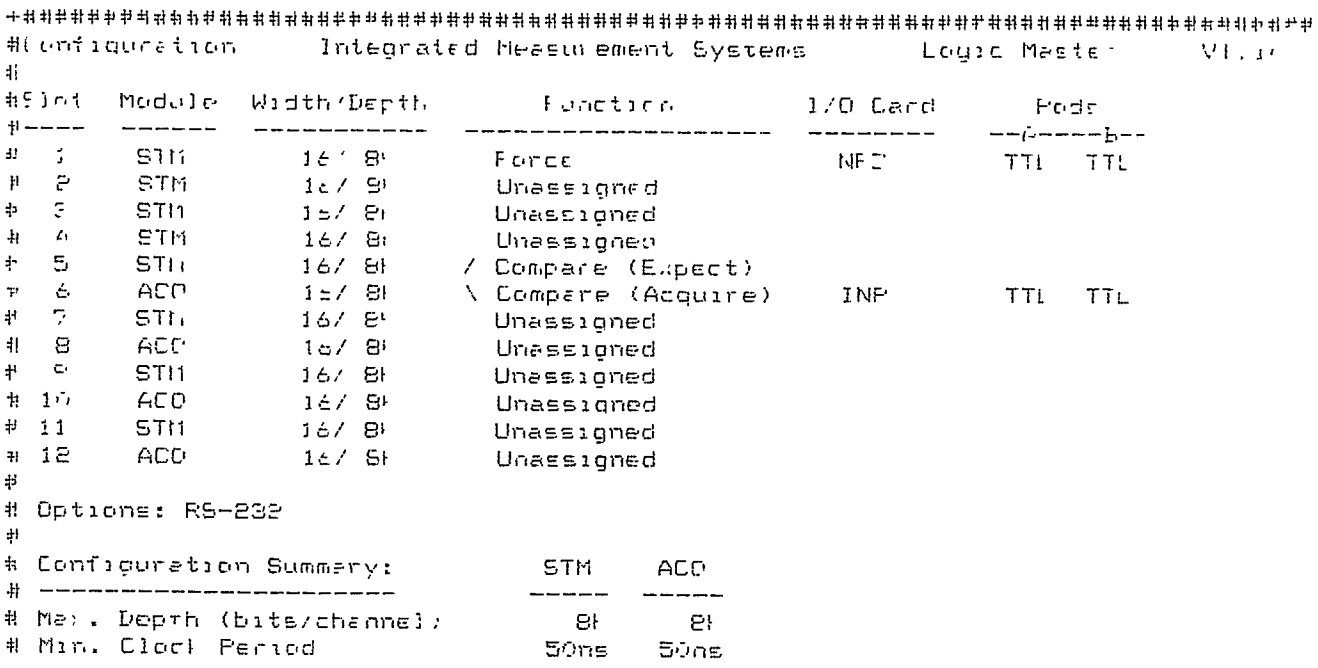


Figure A.1 IMS Test System 1000 Configuaration Screen.

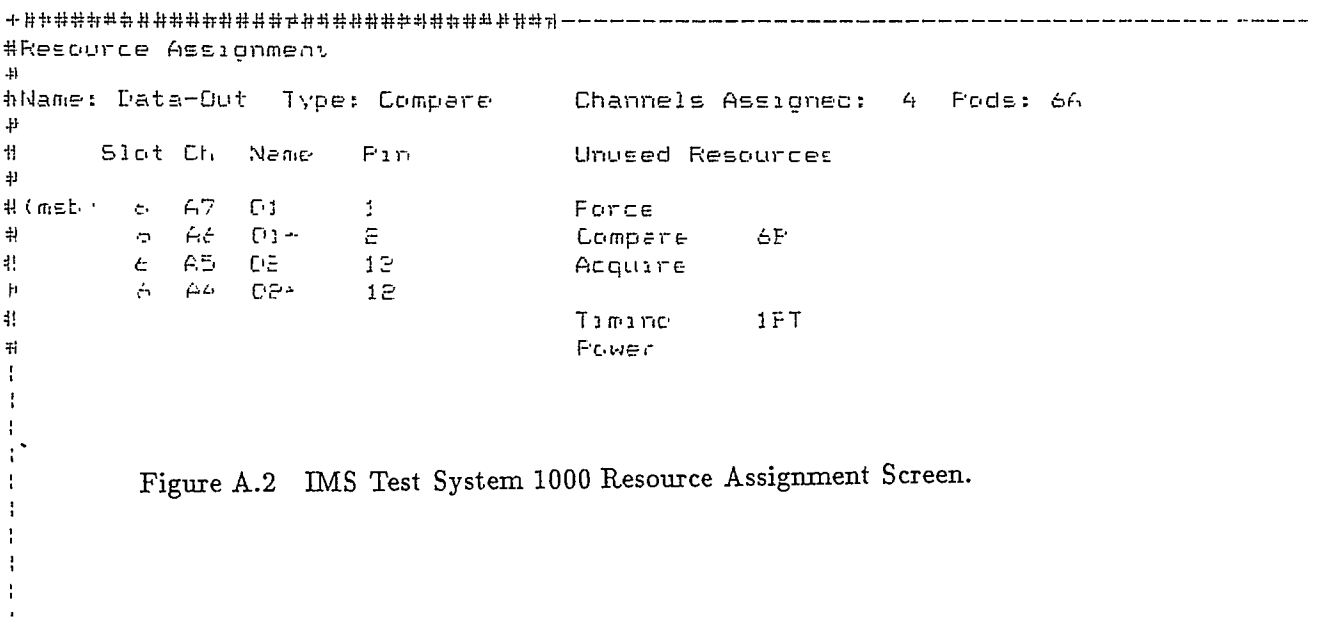


Figure A.2 IMS Test System 1000 Resource Assignment Screen.

```
+#########################################################################################
#Operating Conditions     System Clock: Internal Period:    1.000us
#
#                                                    0 cycles have compare errors
#FORCE/TIMING Groups
#  Name       Type     Display Pol  Low Drive  High Drive  Format  Delay    Width
#
#  Data-In    Force      Bin   Pos      200mV       3.50V    NRZ
#  Control    Force      Bin   Pos      200mV       3.50V    NRZ
#  Clk        Timing     Bin   Pos      200mV       3.50V    RZ       250ns    500ns
#
#COMPARE/ACQUIRE Groups
#  Name       Type     Display Pol  Threshold  Sample at
#
#  Data-Out   Compare    Bin   Pos      1.40V      900ns
#
#POWER Groups
#  Name       Type        Voltage      Limit      Holdoff   Disabled State
#
#  Vcc        Power         5.00V      250mA         0s          HIZ
#
#
#
#
```

Figure A.3 IMS Test System 1000 Operating Conditions Screen.

```
+#########################################################################+-----------------
#Pattern Control              3 cycles have compare errors              Cycle =       0
|
|              -Force/Timing-  -Compare (Exp)  -Compare (Acq)  ---------Instructi
|          E
|          r      SRSR C
|          r      eses 1        D D            D D
|          o  DD tttt C        0102           0102
|Sequence  r  12 1122 2        1*2*           1*2*          ------------------
|--------  -  --------------  -------------  -------------
|        0     00 0101 0       0101           0101
|        1  e  00 1010 0       1010           1011
|        2     00 0000 1       0101           0101
|        3  e  01 0000 1       0110           0101
|        4     10 0000 1       1001           1001
|        5  e  11 0000 1       1010           1001
|        6     00 0000 1       0101           0101           Halt
|        7     zz zzzz 1       XXXX
|        8     zz zzzz 1       XXXX
|        9     zz zzzz 1       XXXX
|       10     zz zzzz 1       XXXX
|       11     zz zzzz 1       XXXX
|       12     zz zzzz 1       XXXX
|       13     zz zzzz 1       XXXX
```

Figure A.4 IMS Test System 1000 Pattern Control Screen.