MICHIGAN STATE UNIVERSITY LIBRARIES
3 1293 10785 5185

MSU
LIBRARIES

RETURNING MATERIALS:
Place in book drop to remove this checkout from your record. FINES will be charged if book is returned after the date stamped below.

300 R
DEC 05 1997
354

THE DESIGN OF TESTABLE PROGRAMMABLE LOGIC ARRAYS

BY

Tsin-Yuan Chang

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electric Engineering
and Systems Science

APRIL 1987

THE DESIGN OF TESTABLE PROGRAMMABLE LOGIC ARRAYS

BY

Tsin-Yuan Chang

Department of Electric Engineering
and Systems Science
Michigan State University

Abstract

The key to easily testable programmable logic array (PLA) is the ability to activate any arbitrary one product line during the test. Based on this activation mechanism, two testable PLA designs for both function-independent and function-dependent tests are presented in this study. In the former, the design of the product line activator is proposed to reduce the complexity of the test pattern generation.

In the latter, methods to generate the test sets for both Growth faults and Disappearance faults from the product term specification of the PLA, are presented. The proposed algorithm can be applied not only to generate the test patterns, but also to detect the redundant crosspoints in a PLA.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his major advisor, Dr. Chin-Long Wey, for the guidance and encouragement in the course of this research.

He also wishes to thank the committee members Dr. P. D. Fisher and Dr. M. A. Shanblatt, for giving valuable suggestions and comments in this work.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER I

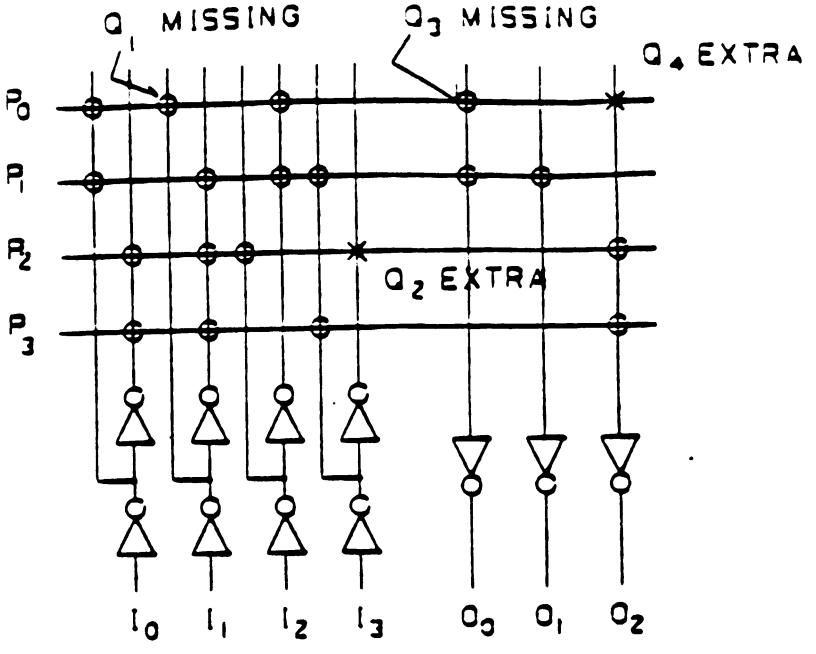
Introduction

Programmable logic arrays (PLAs) have become increasingly popular for implementing control logic in Very Large Scale Integration (VLSI) systems. Although PLA implementation of such function requires larger chip areas than the random logic implementation, the simplicity of the design and regularity of the structure of PLAs reduce the complexity of the overall chip design. Because of these advantages, a trend toward manufacturing larger PLAs is expected. The testing of such PLAs thus becomes a rather difficult problem.

Figure 1(a) shows the general structure of a PLA, and Figure 1(b) is the cubical notation of Figure 1(a). Table 1 summarizes the cubical notation:

Table 1. Cubical notation.

	AND plane	OR plane
1:	connect to complement input	connect to output
0:	connect to uncomplement input	no connect to output
x:	no connection	not used



(a)

	I_0	I_1	I_2	I_3	O_0	O_1	O_2
P_0	1	1	0	x	1	0	0
P_1	1	0	0	1	1	1	0
P_2	0	0	1	x	0	0	1
P_3	0	0	x	1	0	0	1

(b)

Figure 1. (a) A general structure of PLA, and (b) its Cubical notation.

A typical PLA is formed by an AND plane, accepting the true and complement bits of the input lines (I_0 - I_3), and an OR plane, providing the output lines (O_0 - O_2). The output of the AND plane is fed to the OR plane through product lines (P_0 - P_3). Pull-down transistors or crosspoints are formed at the intersections of the input lines and output lines with the product lines to implement the desired logic function. In NMOS technology, both AND and OR planes are built by NOR functions.

In recent years, research has extensively dealt with the test generation and fault detection of PLAs. The optimum design to minimize the test pattern generation cost is achieved by eliminating the expensive stage of test generation. Enhancement in testability is accomplished through the use of additional logic to control individual product lines in test mode. Typically, such a control is achieved by using either a shift register (or shift register with multiplexer), or using extra bit lines to form a decoder (or decoder-like structure).

Redundancy has been used extensively by semiconductor manufacturers to enable the repair of partially defective memory chips [Sch78][Moo86], multiprocessor systems [KoP86], and processing arrays [SaS86]. It has been proved that the yield of integrated circuits using redundancy has been enhanced significantly [Moo86][KoP86][SaS86][SMD80][CCH79]. Recently, a novel design of

fault-tolerant PLA has been proposed [WVL86], in which the redunancy is used to repair the defective PLAs. It has been shown that the yield of such design is significantly improved [WCV86]. However, this significant yield improvement could be offset if the hardware overhead is increased due to the redundancy required for the testing of PLAs. Therefore, the issue of hardware overhead reduction is of significant importance to the design of fault-tolerant PLAs.

In Chapter II, the physical failures in a PLA are discussed. Three types of faults are considered: **crosspoint faults**, **bridging faults**, and **stuck-at faults**. Considerable research efforts which are being devoted to the testing of PLAs can be divided into two classes: test generation and testable design. The existing techniques in both approaches are reviewed in Chapter III. Note that the techniques considered are by no means exhaustive. However, the additional information can be found from [Zhu86].

In order to reduce the hardware overhead for the design of fault-tolerant PLAs, the designs for both the function-independent and function-dependent tests are investigated here. The use of extra logical circuitry to design a product line activator for an alternative decoder structure PLA is presented in Chapter IV. The product line activator is employed to activate only one product line at a time so that the observability in the output lines is increased.

Since this acivation mechanism is independent of the function the PLA realize the test is referred to as function-independent. The proposed design offers the following salient features: (1) homogenous and regular structure; (2) less chip overhead than the shift register approach; (3) no performance degradation during the normal operation due to the added hardware; (4) no additional I/O pins required; and (5) no extra test sequence needed.

As far as the hardware overhead reduction is concerned, the use of no extra logic to the test of PLAs is studied in Chapter V for the further reduction. Based on the activation mechanism developed in Chapter IV, methods to generate the test sets from the product terms specification of the PLA under test, are presented. Since the test sets are determined by the logic functions realized by the PLA, this test approach is referred to as function-dependent. The proposed algorithm can be applied not only to generate the test patterns, but also to detect the redundant crosspoints in a PLA. The algorithm has been implemented on a VAX 11/780 in FORTRAN.

The conclusions and futher research directions are given in Chapter VI.

CHAPTER II

Fault Models

The fault models considered in the design of NMOS PLAs are: **crosspoint faults**, **bridging faults**, and **stuck-at faults** [SGM84].

A **crosspoint fault** is caused by the unintentional presence (or absence) of a transistor. Crosspoint faults can be subdivided into two classes: missing crosspoint faults and extra crosspoint faults. The former is due to a missing contact at the crosspoint in the AND plane or the OR plane; the latter is due to the unwanted presence of a contact at the crosspoint.

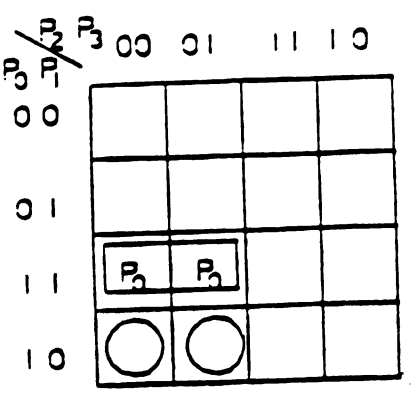
According to the location of the crosspoint faults, it is possible to distinguish four types of faults: **growth faults**, **shrinkage faults**, **disappearance faults**, and **appearance faults**. A growth fault is caused by a missing crosspoint in the AND plane resulting in the disappearance of an input variable from a product term. A shrinkage fault is caused by an extra crosspoint in the AND plane resulting in an additional input variable in a Boolean product term. A disappearance(appearance) fault is due to a missing (extra) crosspoint fault in the OR plane.

The four types of crosspoint faults in the PLA of Figure 1 are shown in Figure 2 with their effects on the corresponding product terms. For example, if the contact at Q_1 is missing, it is equivalent to the product term P_0 growing in size (Figure 2a). On the other hand, if there is a spurious contact at Q_2, this has the effect of shrinking the product term P_2 (Figure 2b) [AbF86]. When there is a missing device in the OR plane, for example Q_3, the product turm P_0 is disappeared from the output O_1 (Figure 2c). However, when an extra device appears at Q_4, the extra product term P_0 will appear the output O_2 (Figure 2d).

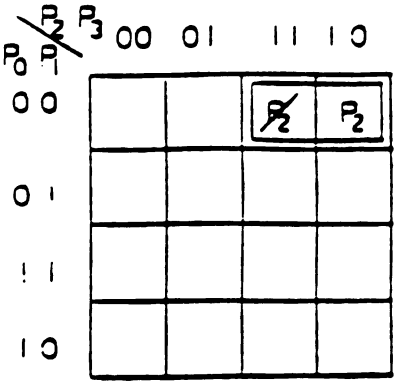
In practice, the "extra device" fault model in modern VLSI circuits has much less significant than the other fault models normally considered for PLAs. It has been shown that extra devices represent less than 0.5% of faults mapped from physical failures [KhB85].

A bridging fault is a short between two adjacent or crossing lines. This fault forces the same logic value to appear in both the lines. A bridging fault can occur either in the AND or the OR plane.

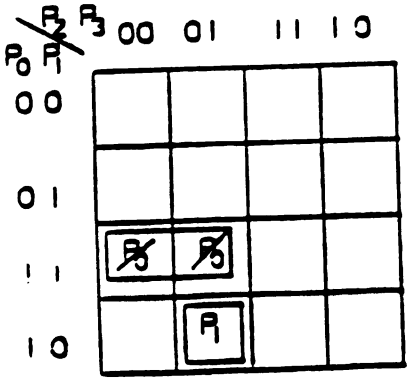
If a bridging fault is present between two adjacent metal lines, it will cause either a logical AND, or a logical OR, of the bridged product terms in the plane of occurrence. If a bridging fault makes a bridge between the drain diffusion line and the grounded diffusion line, it is a stuck-at-0 fault. Alternately, if a bridging fault



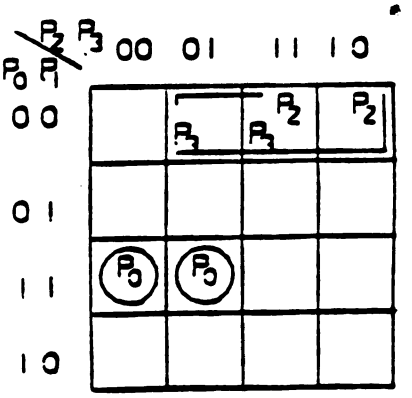
(a) Growth fault.



(b) Shrinkage fault.



(c) Disappearence fault.



(d) Appearence fault.

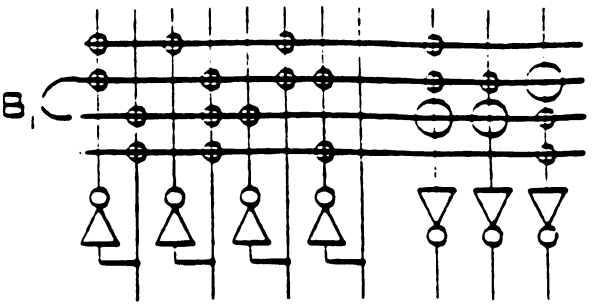
Figure 2. crosspoint faults.

occurs between two transistor drain diffusion lines, it will turn into a metal line bridging fault.

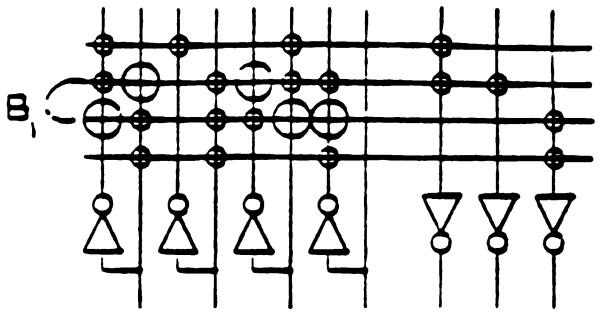
Consider a bridging fault occured at B_1 of the AND plane (Figure 3), if high dominates, it will result in logical OR of the two bridged product lines, so that the output functions are changed as shown in Figure 3a. Nevertheless, if low dominates, a logical AND of the two bridged product lines is resulted, and then the bridged product lines in the AND plane are altered as in Figure 3b. Consider a short between two output lines and assume that high dominates, so that both outputs will be at logical 1 in the faulty PLA if at least one output is at logical 1 in the fault free PLA, as shown in Figure 3c [SoG86].

A **stuck-at fault** is the simplest type of fault that can occur in a PLA. A stuck-at fault is the result of a metal (or diffusion) line opened, or shorted to ground or VDD. A single break in the line can result in the line stuck-at-zero [FKH80], [FuK81]. This corresponds to a metal line which is opened in either the AND plane or the OR plane. If the opening occurs at the diffusion line of a transistor, the same scenario of the missing crosspoint applies. When a metal line is shorted to ground in the AND (OR) plane, then this is equivalent to a stuck-at-zero (one) fault. Finally, if a metal line is shorted to VDD, then a stuck-at-one fault is present.

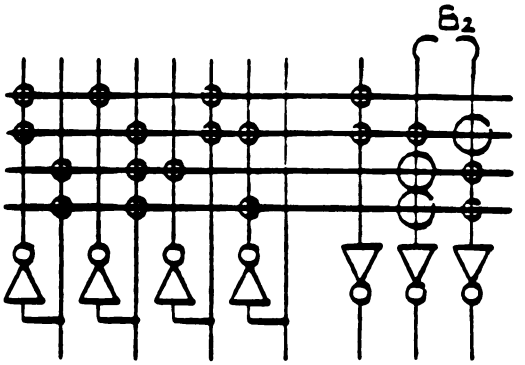
All single stuck-at faults in a PLA, except output stuck-at-one, are equivalent to crosspoint defects [SoG86].



(a) Bridging fault at B_1, High dominates.



(b) Bridging fault at B_1, Low dominates.



(c) Bridging fault at B_2, High dominates.

Figure 3. Bridging faults.

The various fault models which may occur in a PLA can make test generation a complex process. However, analysis of the relationship between different types of faults reduces the complexity of the problem. A complete test set for single crosspoint fault also covers most single stuck-at faults in input decoders and output lines, as well as many shorts and a large portion of multiple faults [Smi79][Osh79]. Any stuck-at fault or bridging fault (of AND type) is equivalent to multiple crosspoint fault [Min 84]. It has been verified that 98% of all multiple crosspoint faults of size 8 and less are inherently covered by every complete single crosspoint fault test set in a PLA [Agr80]. These results indicate that single crosspoint faults should be of primary concern in testing. In case other classes of faults are considered significant, special effort must be made to ensure for their high fault coverage.

CHAPTER III

Test Methods

A PLA corresponds to a two-level sum-of-product combinational circuit. To test the PLA one may simply convert the PLA into a two-level gate and then find tests for stuck-at faults using the existing test generation algorithm, such as D-algorithm. However, as far as the fault behavior is concerned, the faults such as an extra crosspoint fault in the AND plane can not be modeled as a stuck-at fault in the gate circuit. Therefore, high fault coverage is not guaranteed. On the other hand, traditional test generation algorithms are not always effective for PLAs because PLAs have high fan-in, fan-out, redundancy and special fault models. Although exhaustive testing and random testing approaches are effective on some combinational circuits, they are impractical as the size of the PLA increases.

Considerable research efforts have been devoted to the testing of PLAs could be divided into two classes: test generation and testable designs.

3.1. Test Generation

Most of the earlier approaches fall in this class. Regularity of the structure is exploited to derive optimal or near optimal test sets to detect different types of faults in PLAs. However, the basic idea behind most PLA test generation algorithms is path sensitization, to select or deselect a product line and then sensitize the chosen product line through one of the output lines. Knowing a PLA's personality, tests of this nature can be easily found.

In [SGM83],[SGM84], the well-known Shannon's expansion theorem

$$F(x,y) = xF(1,y) + \bar{x}F(0,y)$$

is employed to find the test patterns of all possible faults. Consider the example of Figure 1. Suppose a test is generated for a missing device fault at transistor Q_1, it checks

$I_0I_1I_2I_3$	$O_0O_1O_2$			$I_0I_1I_2I_3$	$O_0O_1O_2$
1 0 0 x	1 0 0	with	P_2	1 0 0 1	1 1 0
			P_3	0 0 1 x	0 0 1
			P_4	0 0 x 1	0 0 1

and rules out P_3 and P_4 because the input bit I_0 is different. The problem reduces to:

		I_0	O_0	O_1		I_0	O_0	O_1
does	P_2	1	1	1	cover	x	1	0

By applying Shannon's theorem with x = 0 and checking F(0,y), the following question is posed:

$$\text{does} \quad P_2 \; \begin{matrix} I_0 & O_0 & O_1 \\ 1 & 1 & 1 \end{matrix} \quad \text{cover} \quad \begin{matrix} I_0 & O_0 & O_1 \\ 0 & 1 & 0 \end{matrix}$$

Clearly, the test pattern (1000) can be selected to detect the fault considered here. Test patterns for other crosspoint faults can be obtained in the similar way.

The test generation methods provide a software solution for the PLA testing problem. They do not require any hardware modification to the PLA. However, as PLAs increase in size, a larger number of test patterns have to be generated and stored. Sophisticated automatic test equipment (ATE) is needed to execute the test process. Hence the testing becomes a time-consuming and expensive task. To alleviate this problem, more hardware oriented approaches have been developed in which the extra built-in self-test (BIST) circuitry is addded to the original PLA such that the modified PLA can be more easily tested.

3.2. Testable designs

Most of testable design techniques proposed to date fall into one of the following categories: special coding, parity checking, divide-and-conquer, and signature analysis. Some techniques are combinations of these design philosophies.

A. Special coding:

A technique proposed by [KhM81] makes use of the following fact about a PLA.

- The input bit lines in the AND plane naturally form a two-rail code.
- For a non-concurrent PLA, during normal operation the signals on the m product lines forms a 1-out-of-m code.
- The fault-free output patterns are determined by the PLA's personality matrix. They can be coded into some error detection code by adding extra output lines to the OR plane.

The proposed testable PLA with concurrent error detection capability, as shown in Figure 4, employes three checkers: C1 is a totally-self-checking (TSC) 1-out-of-m checker on all product lines, and can detect any non-concurrent fault, such as product line stuck-at-1(0), or any missing and/or extra crosspoint faults in the AND plane. C2 is a TSC two-rail code checker which detects all single stuck-at faults in the input lines. C3 is an output code checker whose complexity depends on how the outputs are coded.

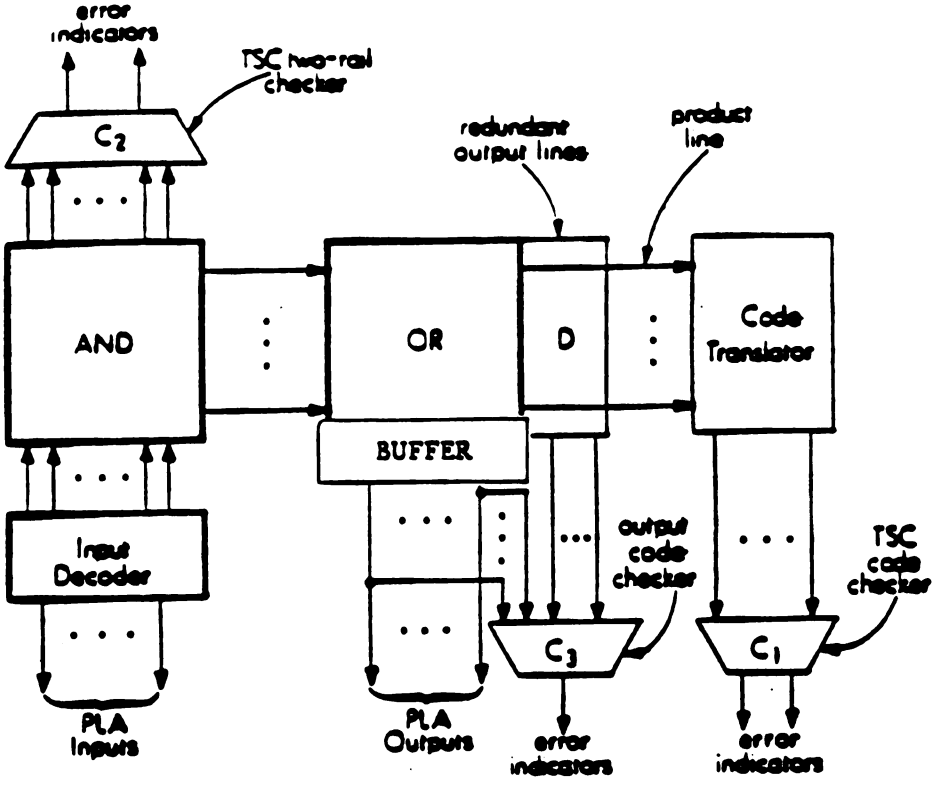


Figure 4. A concurrent testable PLA design [KhM81].

B. Parity Checking

Since PLAs have a regular array structure, they can be designed to test by a small set of deterministic tests which are function-independent. This is based on following important observations:

- One can add extra lines (product lines or output lines) to make the connections of each line odd (even). Then any single crosspoint fault will change the parity and can be dectected by the parity checker.
- In order to easily test a PLA, it must individually control each input line and product line, and sensitize each product line through the OR plane.

In [FuK81], two parity checkers are employed, one in product lines and the other in output lines. In order to make the odd parity, the following additional circuits are added (Figure 5): a product term selector, a modified input decoder, and some extra product lines and output lines. The parity checker is composed of XOR gates. The purpose of using the product term selector and the modified input decoder is to activate one row and one column in the PLA, so that every crosspoint can be uniquely selected and tested. Usually the product term selector is a special designed shift register whose size is the width of product line.

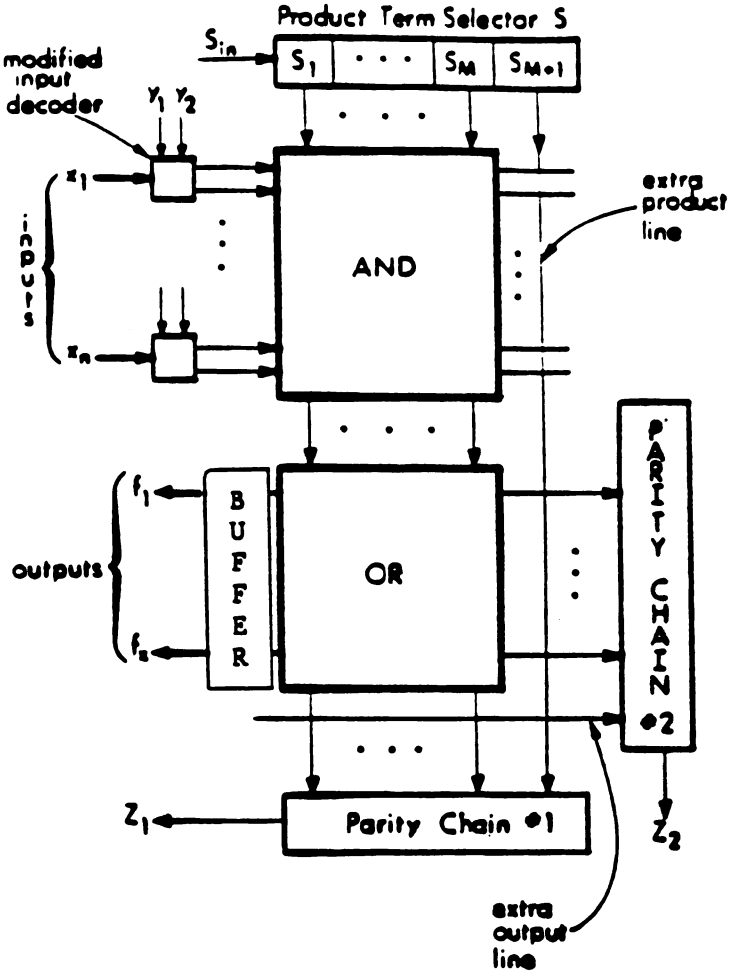


Figure 5. A testable PLA with universal test set [FuK81].

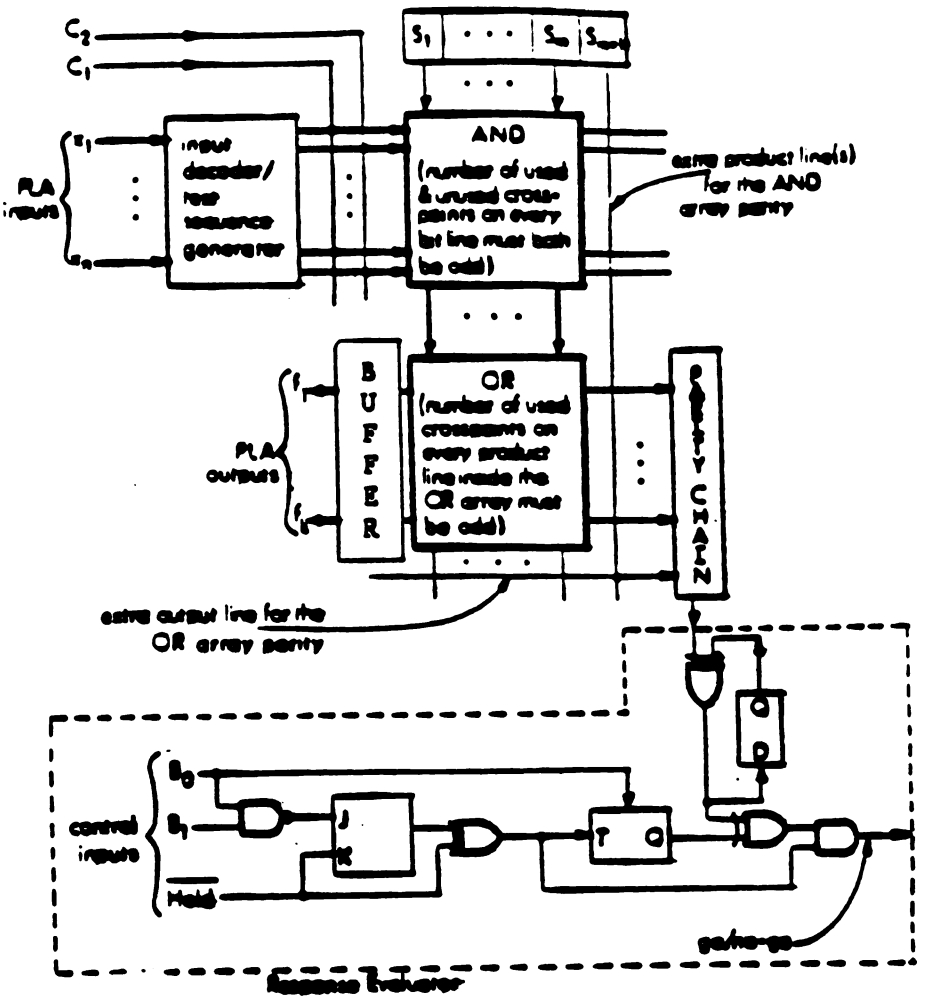


Figure 6. A testable PLA with cumulative parity comparison [Fuj84].

A more efficient design has been proposed [Fuj84], in which the parity check method (Figure 6) is replaced by a cumulative parity comparison method, the value of the accumulated parity signals in a flip-flop is compared with the expected value at specific times to detect the faults. Two control signals C1 and C2 are added which act just the same as the modified input decoder. One or two extra product lines are used to make every input bit line contain the odd number of used and unused crosspoints in the AND plane. The same is done for the OR plane by adding extra output lines. Only one parity chain is used at outputs.

An interesting property of this scheme is that the sequence of cumulated parity bits at 2n+2m+1 selected check points is simply a sequence of alternative 0's and 1's. Hence it is very easy to generate the expected value on-line. It has been proven that the fault coverage of this scheme is very high; all single and $(1-2^{-(m+2n)})$ of all multiple crosspoint, stuck-at and bridging faults are covered, where m and n are the number of product lines and output lines, respectively.

C. Divide-and-conquer and signature analysis

In the divide-and-conquer strategy a suitable testable design methodology is selected for each testable part such that every part can be embedded in a testable structure.

In the signature approach, a set of input patterns is applied and the results are compressed to generate a "signature". When the test is invoked, this signature is compared with a known correct value to determine if the PLA is faulty.

A design of BIST PLA architecture, as shown in Figure 7, has been proposed by Daehn and Mucha [DaM81], in which the combination of divide-and-conquer and signature analysis strategies is employed. This design implements the non-linear feedback shift registers as both test pattern generators and output response compressers. Such registers are basically a modified form of what is known as a built-in logic block observer, or BILBO. Basically the PLA is partitioned into four blocks: input decoder, AND plane, OR plane, and output buffer. Then, three BILBOs are inserted between these blocks. Testing of each block is done as follows: let the input BILBO of that block operate as a test generator and the output BILBO of that block operate as a signature analyzer. The result is shifted out for inspection. These blocks are tested one by one.

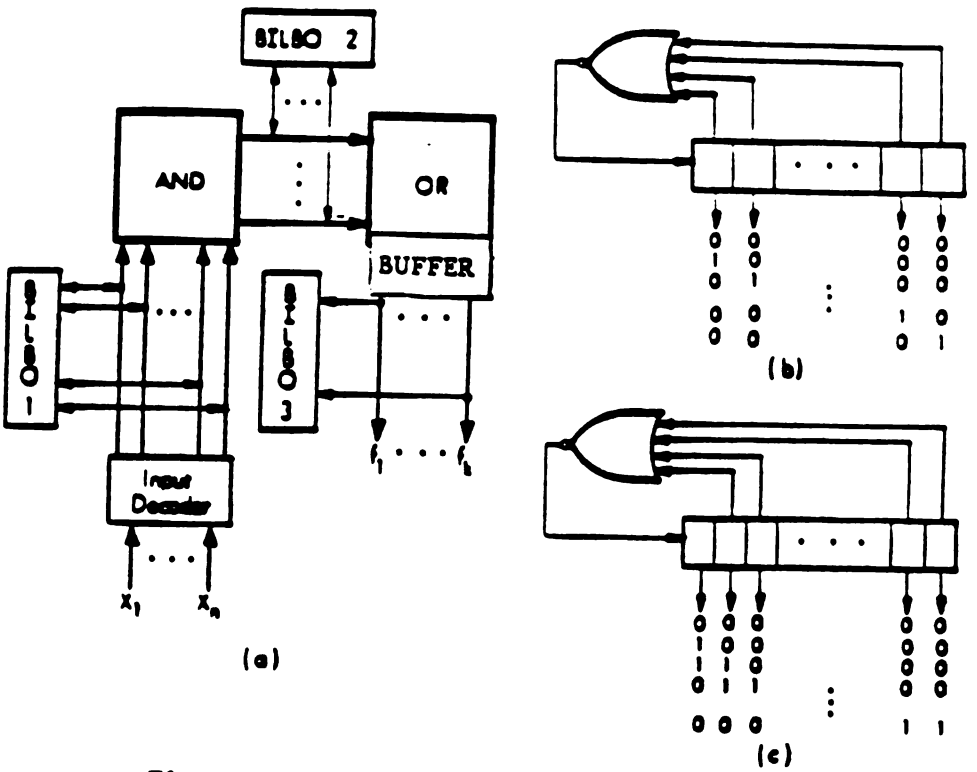


Figure 7. Test a PLA by BILBOs.

After partitioning, the AND plane and the OR plane are just arrays of NOR gates. All inputs are controllable and all outputs are observable, thus testing becomes a very simple task. It is known that a k-input NOR gate can be fully tested by a simple sequence such as

```
0 0 0 . . . 0
1 0 0 . . . 0
0 1 0 . . . 0
0 0 1 . . . 0
0 0 0 . . . 1.
```

A NOR gate array can be tested by the same pattern. The test generator can be achieved by a non-linear feedback shift register as shown in Figure 7(b), which produces above patterns. It has been shown that all single stuck-at faults, crosspoint faults and bridging faults in AND plane and OR plane are detectable using this sequence [DaM81].

Input decoder is tested by a similar sequence generated by the non-linear feedback shift register shown in Figure 7(c).

For the testable PLA designs, it has been recognized that the key concept in enhancing PLA testability has been the provision of means to control individual product lines [BoM84]. Most of the current testability designs accomplish this goal by incorporating shift registers in the PLA design [FuK81]. The data stored in the shift

register is used to control the product lines. Unfortunately, the area of the shift register added generally cannot match with the compact PLA layout and thus is significant overhead [HJA84].

In [BoM84], some extra input lines, as shown in Figure 8, are added to the original AND plane so that the augmented AND plane acts as a decoder that uniquely selects each product line. In this approach, a set of main test patterns and the corresponding auxiliary test patterns are employed to detect the faults. A main test pattern is generated in a way that one and only one product line is selected at a time. The auxiliary test patterns are generated by flipping the bits in the main test pattern, one at a time, for each main pattern. The purpose of an auxiliary pattern is to disable the chosen product line while maintaining the deselection of other product lines. An algorithm has been proposed to assure the Hamming distance between every pair of main test patterns to be two [BoM84]. In order to satisfy the Hamming distance requirement, a heuristic is implemented to increase the number of input lines, or crosspoints in the chosen input lines. It has been proved in [BoM84] that all multiple stuck-at faults, as well as all multiple extra and multiple missing device faults, are detected.

Later in [KhB85], a method was proposed to reduce the extra hardware overhead. The algorithm is based on the fact that as long as the effect of deselecting a product line with an auxiliary test

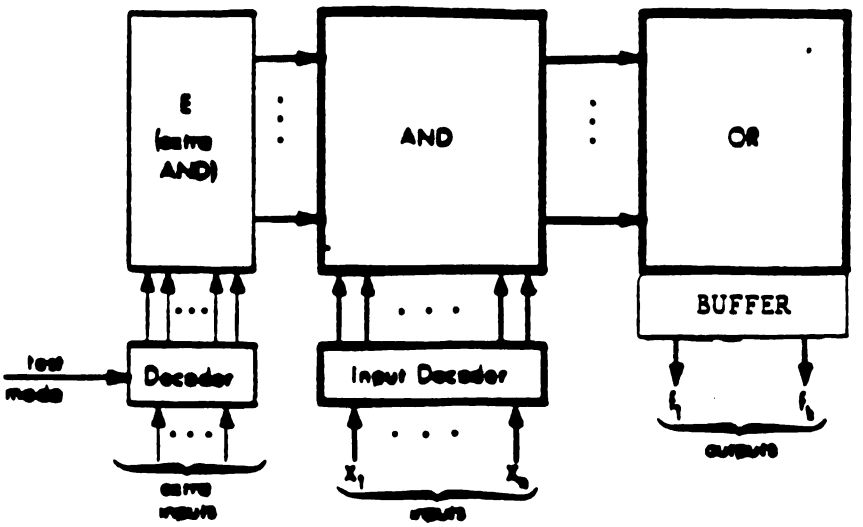


Figure 8. A testable PLA design [BoM84].

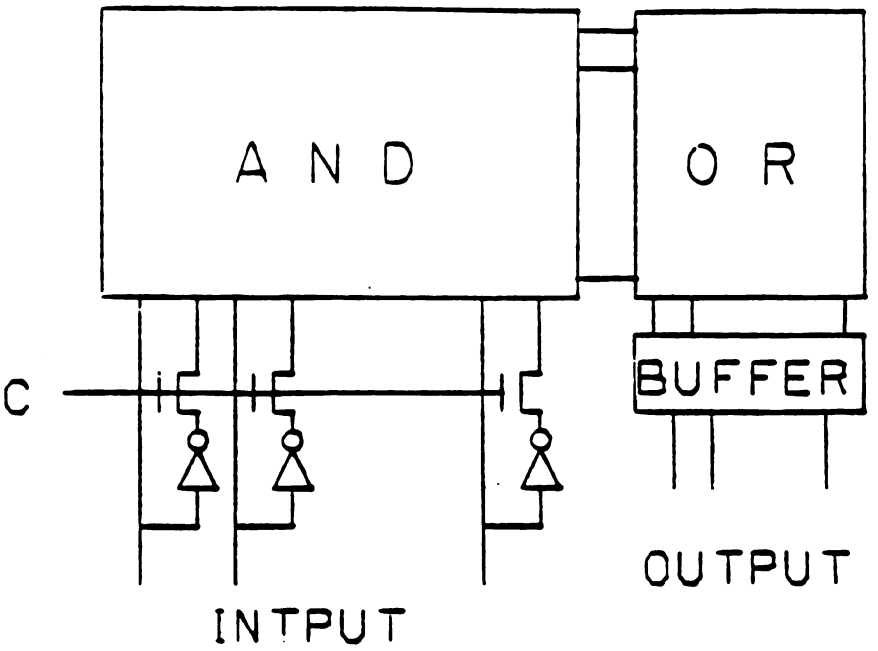


Figure 9. A testable PLA design [HaR85].

pattern is propagated to at least one output line, the distance of two is not necessary. In addition, it was claimed that "extra device" fault model is not significant in modern VLSI circuits. Therefore, the requirement of choosing a main test pattern is relaxed by the following fact. It is not necessary to find a vector that uniquely selects a product line if the collective output response of the other selected product terms do not cover the observability of the chosen product line. A modified algorithm of [BoM84] was then proposed to take these advantages and reduce the number of input lines added.

One major disadvantage in adding new input lines is that the number of input lines needed for providing testability is not a constant and depends on the functions implemented by the PLA. In addition, the extra input lines have to be disabled during normal operation.

One unique feature of PLAs is that the input lines feed their true and complement bits into the AND planes This fact places an important restriction on the applicable test patterns as mentioned before.

In fact, a Hamming distance of one between each pair of main test patterns is sufficient to assure the existence of main test patterns. This can be demonstrated by the PLA shown in Figure 1. "1101" is a good main test pattern for P_0 even though the Hamming distances between P_0 and all other product lines but p_4 are only one. However, if the

Hamming distance between two main test patterns is less than two, there is always the possibility of accidentally selecting another product line when a bit of the main test pattern is flipped to generate an auxiliary test pattern. In the above example test pattern for selecting P_0, if the leftmost bit is switched, the auxiliary test pattern, "1001", will accidentally select P_1. This is due to the fact that the distance between P_0 and P_1 is less than two. [KhB85] pointed out the special conditions when this dilemma can be tolerated.

In essence, it is not necessary to add any extra input line for an irredundant PLA, even if two product lines have zero Hamming distance and the same outputs. This motivates the study of Chapter V.

Recently, a new testable PLA design has been proposed by Ha and Reddy, as shown in Figure 9, in which the normal PLAs are augmented with the addition of pass transistors in the input decoder [HaR85]. The transistors are used to temporarily disconnect the true bits of the inputs, so that their previous values can be retained for a short period by the line parasitic capacitances, and new values can then be applied to the complement bits. As a result, arbitrary test patterns can be applied to all true and complement bits. In other words, each true or complement bit of an input line can be controlled independently. This significantly enhances the testability of PLAs.

One shortcoming of this method is that two phases are required to apply a test pattern which the values of the true bits are assigned in

the first phase and the values of the complement bits are assigned in the second phase. A more severe disadvantage is that a test pattern can only be maintained for a brief time due to the small parasitic capacitances available on the lines.

To improve the deficiency of the above approaches, an alternative design of easily testable PLA is presented in the next chapter.

CHAPTER IV

The Design of Easily Testable PLAs

The key to easily testable PLA design is the ability to activate one and only one product line. This activation mechanism enhances the observability in the OR plane, and it is usually achieved by adding shift registers to select the product lines. The data stored in the shift registers are used to operate the activation mechanism. However, a shift register cell is wider than a product term. This makes the shift register wider than the PLA, with cells extending beyond either end. This mismatch wastes area and distorts the floor plan of the PLA [BHM84].

Recently, a decoder structure has been proposed by Sato and Tohma [SaT82] to achieve the same activation mechanism. In this approach, Figure 10, two control lines C_1 and C_2 are used either to deactivate all AND word lines or to acivate only one AND word line of the input decoder, and another control line C_0 is applied to disable the PA (Product line Activator) during the normal operation. It has been shown that the proposed design structure is more homogenous than that of shift register. However, the need for extra input pins and performance degradation due to the modified input decoder, limits its applicability.

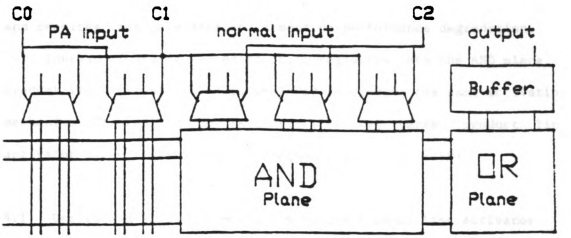


Figure 10. A testable PLA design with decoder structure.

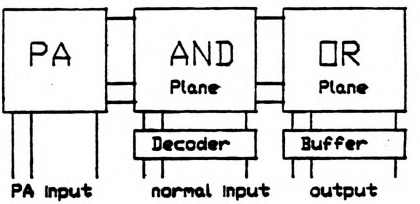


Figure 11. A testable PLA design with product line activator (PA).

The objective of this chapter is to propose an alternative decoder-like structure that performs the same activation mechanism, and requires less chip area and almost no performance degradation.

Instead of adding the extra shift registers into the AND plane, a product line activator (PA) is employed to operate the same activation mechanism (Figure 11). The PA consists of two parts: product line activator circuit (PAC) and code sequence generator (CSG).

4.1. Design and VLSI Implementation of the Product Line Activator

Consider a conventional 3-bit input line decoder with PLA implementation as illustrated in Figure 12. A product line can be activated according to the decoded input bits. For example, if a sequence of the numbers from 0 to 7 (represented in binary form) is applied one at a time, then only one product line is activated in the order from the top to the bottom. This activation mechanism in the decoder is much easier than that of the shift register. However, the only disadvantage behind this decoder structure is the need of the extra input pins. Therefore, in order to reduce the extra pins, the code sequence must be generated and applied internally.

The code sequence generation can be accomplished by using a linear feedback shift register (LFSR). In fact, a LFSR implemented with EX-NOR gate consumes less chip area than that with EX-OR gate

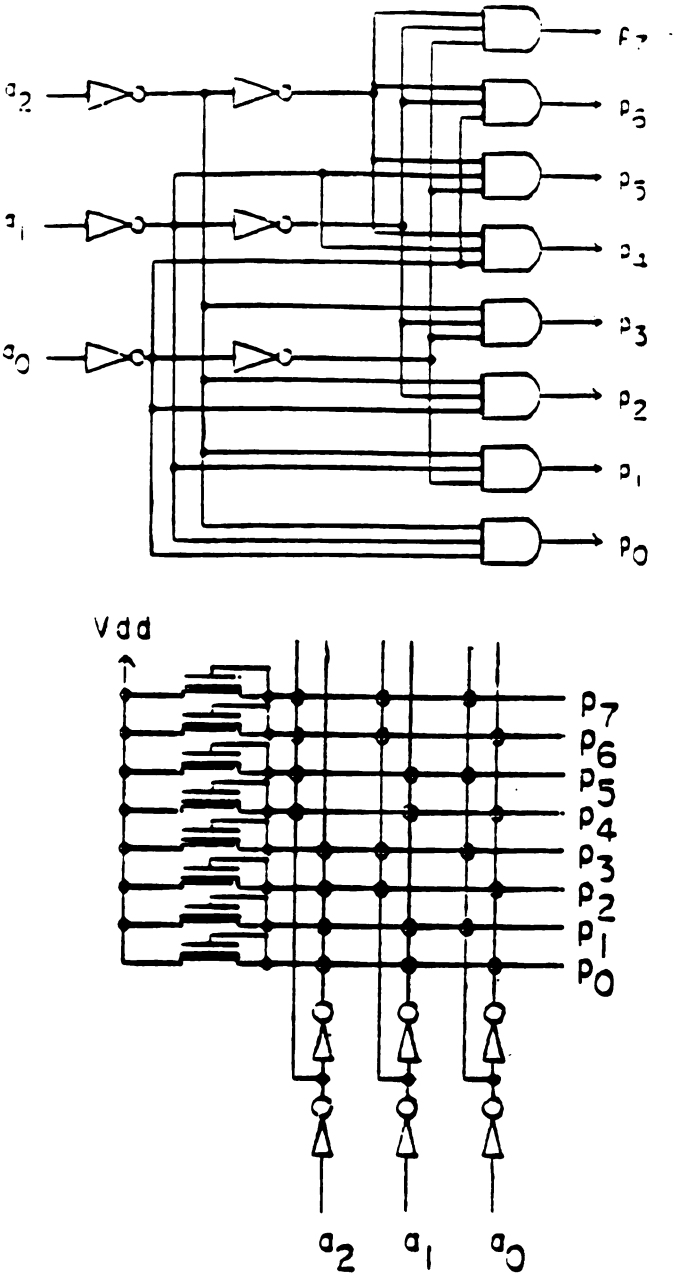
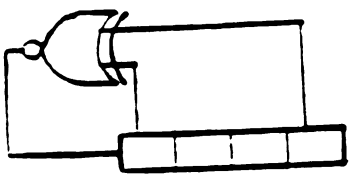


Figure 12. The 3-bit input decoder.

A four-bit modified LFSR is shown in Figure 13(a). With the seed, $(b_3b_2b_1b_0)$=(0000), a total of 15 (2^4-1) code sequences is generated in Figure 13(b). On the other hand, when a seed (1111) is applied, the modified LFSR will generate the same pattern (1111). The design of PAC with PLA implementation is illustrated in Figure 13(c).

Since the length of the cyclic code sequence generated by the modified LFSR may not coincide with the number of product lines in a PLA, a control signal is needed to restart the operation of the modified LFSR, so that the code sequence can be applied periodically. As shown in Figure 13(c), the extra product line, BMPLI, is employed as the control line. In this implementation, BMPLI is programmed as same as the logic function of the bottom-most product line.

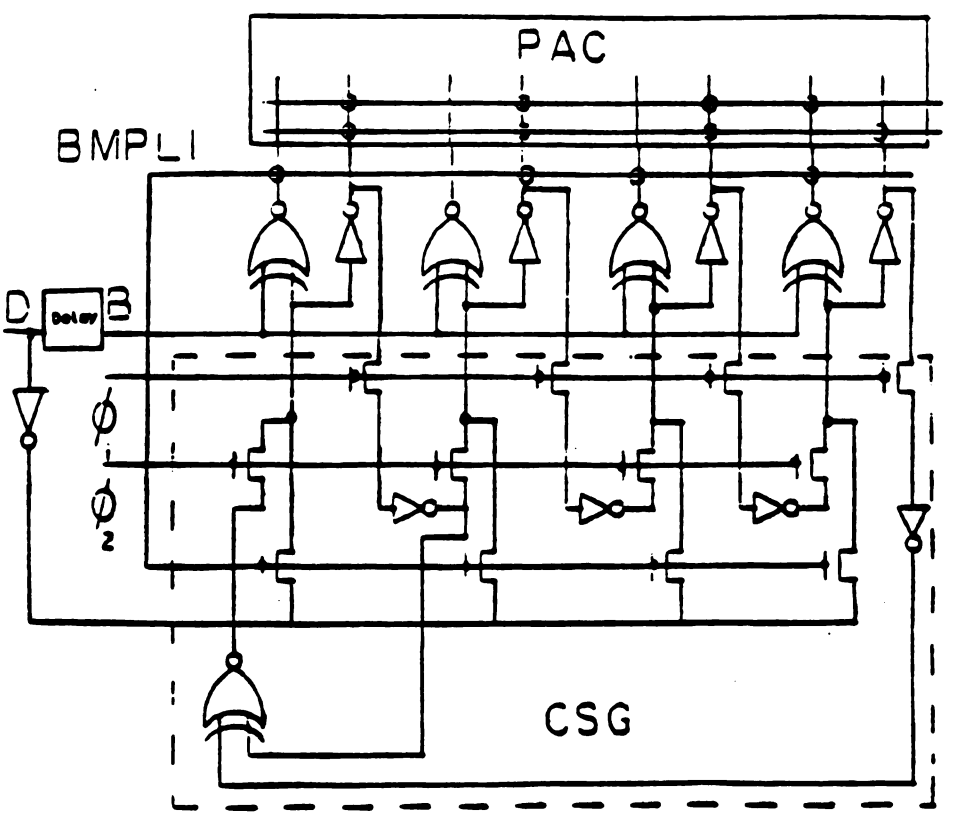
In fact, the PA is used only when the test process is performed, and it must be disabled during the normal operation. The D-line is use to control this operation, where D=0(1) for normal operation mode (for test mode). During the test mode, D = 1, the code sequence is generated by the modified LFSR with a seed of $(\overline{D}\ \overline{D}\ \overline{D}\ \overline{D})$ or (0000). This sequence is continously generated until the corresponding sequence of the bottom-most product line is recognized. Once the code bits stored in the shift registers match with this corresponding sequence, the BMPLI are set to 1. When the test mode is completed, the D-line is then set to 0 for the normal operation. As a result, the seed $(\overline{D}\ \overline{D}\ \overline{D}\ \overline{D})$ becomes to (1111) and will be loaded to the shift registers to disable the PA.



(a) A 4 bits LSFR scheme.

0	0	0	0
1	0	0	0
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
1	1	0	1
1	1	1	0
0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

(b) Code sequence.



(c) The design of PAC and CSG

Figure 13. A schematic diagram of PA design.

4.2. Chip Area Overhead

A floor plan of the PA design is illustrated in Figure 14. all dimensions are in the units of lambda ,λ, [MeC80]. The number of bits required for the PA design depends on the number of product lines in the PLA. For a PLA with m product lines, a k-bit PA is needed, where $k = \lceil \mathrm{Log}_2 (m+1) \rceil$. The chip area required to design the PAC is essentially the same as the area required for the k input lines in a standard PLA. According to Mead and Conway's design rules and [NeM83], a product or output line requires 8λ in width, while an input line consumes 16λ in width. Therefore, the chip area for PAC is approximately $128mk\lambda^2$, or $(8m\lambda)(16k\lambda)$. In addition, each bit of CSG may approximately consume 16λ x 155λ. In other words, the chip area required for the entire PA design is

$$A_{PA} = (8 \times m + 155) \times 16 \times \lceil \mathrm{Log}_2 (m+1) \rceil \lambda^2 \qquad (1)$$

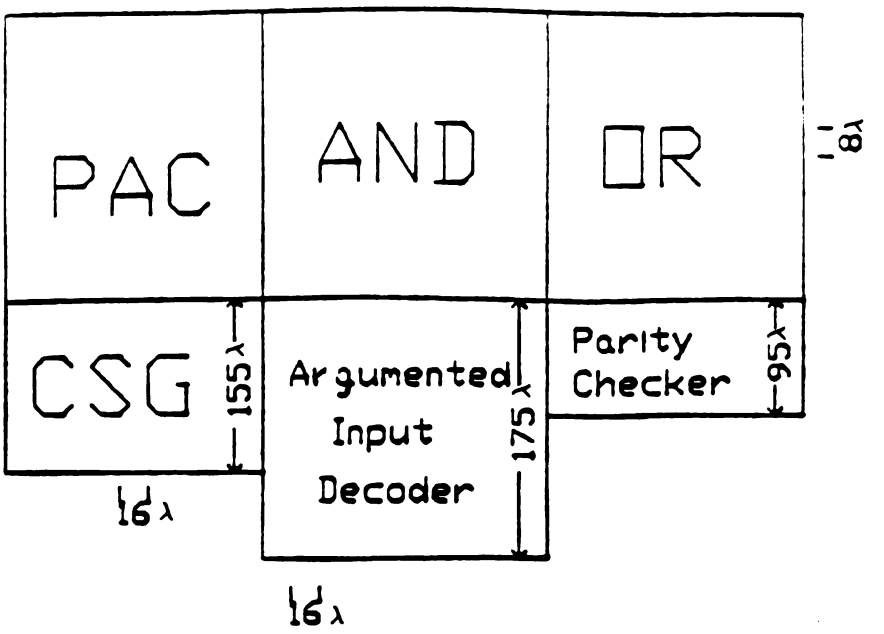


Figure 14. The floor plan of the proposed PA design.

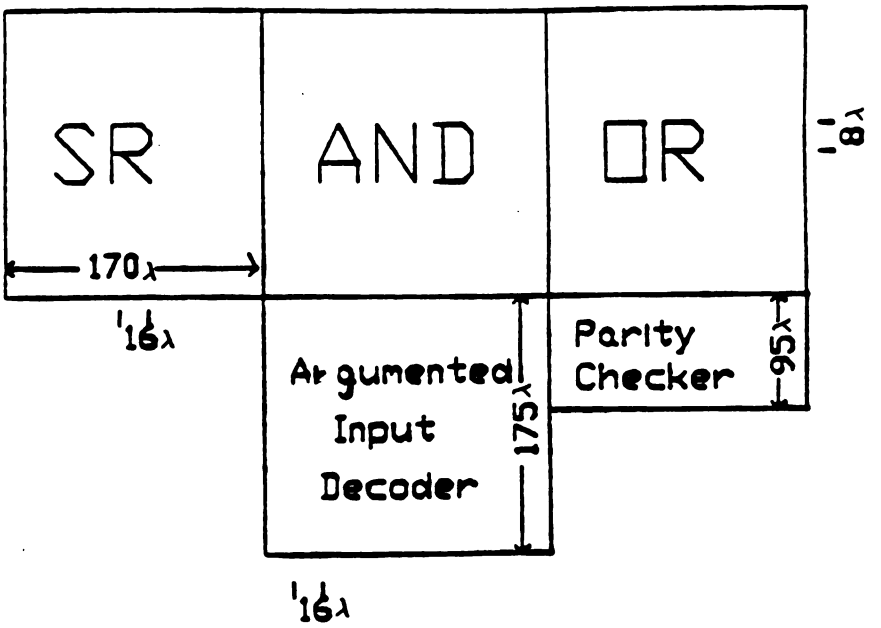


Figure 15. The floor plan of a testable PLA design with a shift register.

4.3. Implementation

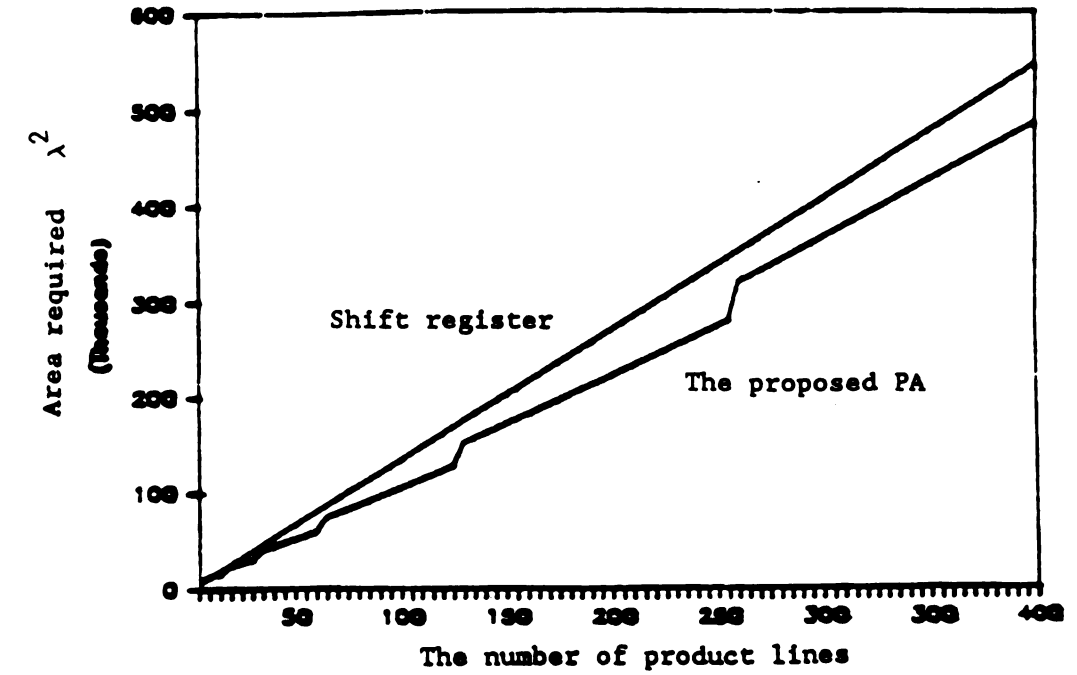
In order to demonstrate the effectiveness of the PA design for BIST PLA design, the implementation of the PA design into TRPLA [TFA85] is presented.

A. Area Overhead

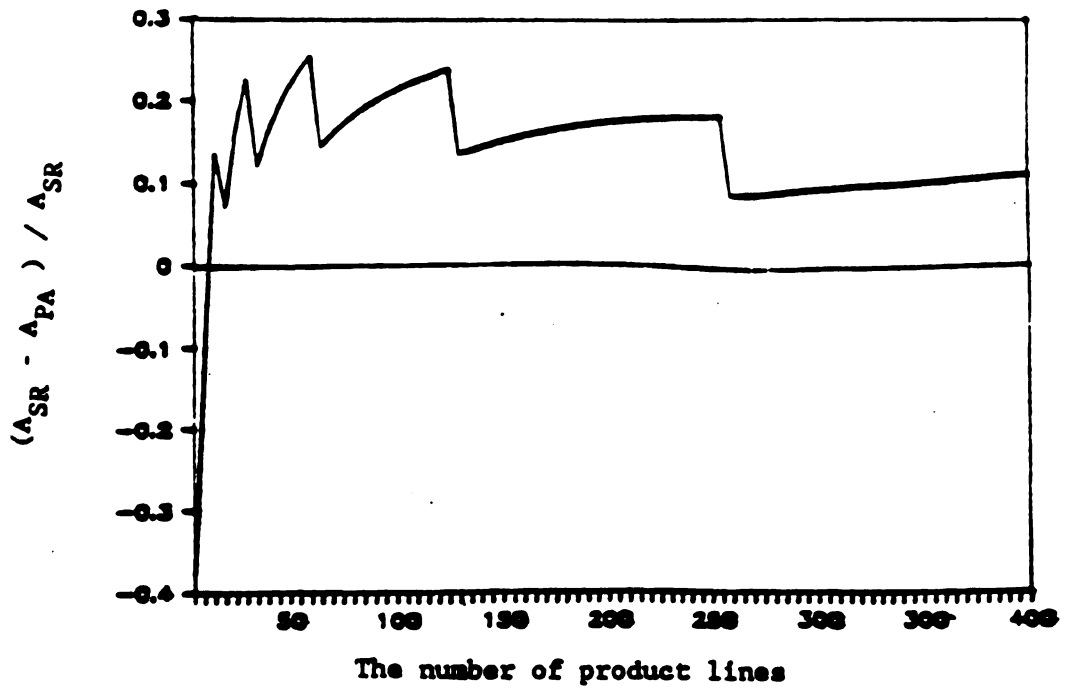
A floor plan of TRPLA is illustrated in Figure 15. The chip area for the shift register consumes $A_{SR} = 1360m\ \lambda^2$, or $170\lambda \times 8m\lambda$. Figure 16(a) plots the areas required for both PA and SR design versus the number of product lines. It is obvious that the PA design requires less chip area than that of shift register (SR) in a reasonable PLA design. The fraction of area reduction in the PA design calculated by $(A_{SR} - A_{PA}) / A_{SR}$ is illustated in Figure 16(b). The curves show that the area reduction can be up to 25%.

B. Test sequences

The test sequences employed in TRPLA design are listed in Figure 17. It has been proved that the test sequences can detect all single faults and almost all multiple faults [Fuj84]. In fact, the PA design has the same mechanism as the SR design, these test sequences can also be applied to the PA design without requiring any extra test



(a) Chip area.



(b) The fraction of area reduction in the PA design.

Figure 16. Chip area overhead.

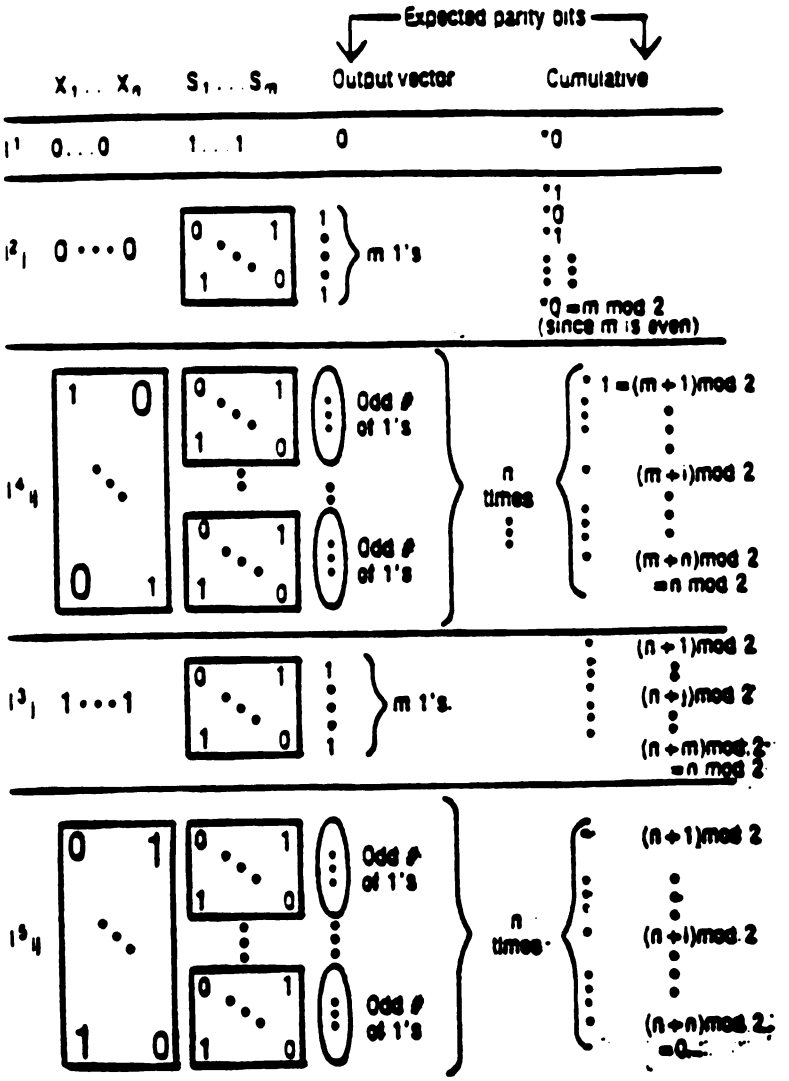


Figure 17. Cumulative parity comparison scheme.

sequences. The only problems remain are how to test the PA itself and whether or not extra test sequences are needed to test it.

Consider the stuck-at faults and/or bridging faults which occur on the product line(s) of the PAC. These are the same as the product line faults in the AND plane and can be detected by appling the test sequence I^1. On the other hand, if these types of faults occur on the input line(s) of the PAC, then they can be observed from the signal of the shift-out pin in the CSG. In the meanwhile, the crosspoint faults occur on the PAC may result the following error cases:

(1) The activation of the product lines is not in order, but one and only one product line is activated at a time;

(2) No product line is activated;

(3) Even number of product lines are activated; and

(4) Odd number of product lines are activated.

According to the activation mechanism in the PA design and the characteristics of the universal test sequences, as long as one and only one product line is activated, the order of activation is not really important. Therefore, the case (1) does not harm the activation mechanism. In other words, the proposed PA design can tolerate the crosspoint faults, as the case (1), in the PAC. The fault tolerant capability is one of the positive features.

If the parity bit detector gives the correct signal for the test sequences, I^1, I^2, and I^3, then the detector is sure that the activation mechanism be functioned properly. On the other hand, if no product line or multiple product lines are activated due to the crosspoint faults, then these faults can be detected as follows.

The single crosspoint fault may cause either no product line in the PAC to be activated, or two product lines to be activated simultaneously, for a certain code sequence generated from the CSG. For example, if a crosspoint is missed at the P_4 and A_3, it will change the function from (100) to (-00). When the code sequence (100) is applied, P_4 is the only activated product line. On the other hand, when the code sequence (000) is applied, the lines P_0 and P_4 are activated simultaneously. In either case, the even parity signal is detected, therefore, the use of the test sequence I^2 can detect that fault.

For the multiple faults, the combination of crosspoint faults may result in the error cases (2), (3), and (4). The crosspoint faults in the error cases (2) and (3) can be detected in a manner similiar to the case of single fault. Due to the odd parity design, the proposed PA design may fail to detect the faults if the crosspoint faults on these activated product lines pass both parity checkers. Specifically, a) to pass I^2_j (I^3_j) test, the crosspoints in the OR plane and in these odd number of product lines activated by the j-th pattern must

enable an odd number of output lines; and

b) to pass I^4_j (I^5_j) test, the number of crosspoints of X_i-line (X'_i-line) in these product lines activated by the j-th pattern must be odd, and the crosspoints in the activated product lines and the OR plane enable only odd number of product lines.

In fact, it has been shown that the probabilty of having these failures is very low and negligible [TFA85]. Therefore, it is not necessary to increase the test sequence in the use of PA design.

4.4 Discussion.

The salient features of the proposed PA design are (1) homogenous and regular structure; (2) less chip overhead; (3) no performance degradation during the normal operation due to the added hardware; (4) no additional I/O pins required; and (5) no extra test sequence needed. The proposed design can be implemented on any testable PLA designs using the shift registers so that the chip area overhead can be reduced.

CHAPTER V

Test Pattern Generation

Consider an AND-OR PLA with n input lines, p output lines, and m product lines. The functions realized by this PLA are represented as arrays L:(C,D) [Smi79], or cubes. A k-tuple $a=(a_1,a_2,..,a_k)$, where a_i is one of the items {0,1,x}, is defined to be a cube. Here the "don't care" term x takes value 0 or 1. The array L has two parts: an input part (C-array, n columns and m rows) and an output part (D-array, p columns and m rows). Each cube/row C_i of the C-array represents a product term of one or more functions realized by the PLA.

Example 1:

A simple schematic diagram of a PLA, as shown in Figure 18, implementing five 4-variable switching functions:

$$O_0 = \bar{I}_0 I_2 I_3 + I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3$$
$$O_1 = I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3 + \bar{I}_1 \bar{I}_3$$
$$O_2 = I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3 + I_0 I_2 I_3$$
$$O_3 = \bar{I}_0 I_2 I_3 + \bar{I}_1 \bar{I}_3$$
$$O_4 = I_0 I_2 I_3$$

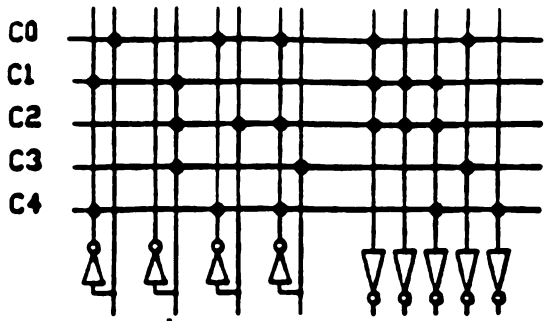
The cubical representation of this PLA is

	I_0	I_1	I_2	I_3		O_0	O_1	O_2	O_3	O_4
C_0:	0	x	1	1	D_0:	1	0	0	1	0
C_1:	1	0	x	x	D_1:	1	1	1	0	0
C_2:	x	0	0	1	D_2:	1	1	1	0	0
C_3:	x	0	x	0	D_3:	0	1	0	1	0
C_4:	1	x	1	1	D_4:	0	0	1	0	1

and the corresponding Karnaugh-map (K-map) is also shown in Figure 18. It should be noted that the notation C_i in the cubes of the K-map represents the 1-cube that is produced by the product term C_i.

Definitions:

1. **A crosspoint-irredundant PLA is one in which all the crosspoint faults are detectable.**
2. **A G-D-irredundant PLA is one in which all G and D faults are detectable.**
3. **Let a and b be any two cubes. If a has 1 in every minterm in which b has 1, then a is said to cover b. If neither a covers b, nor b covers a, then, a and b are said to be unordered.**
4. **A product term C_i is said to be non-isolated (with respect to a given output function), if there is at least one other product term C_j ($i \neq j$), in the functional specification, such that C_i and C_j cover one or more common minterms. Otherwise, it is said to be isolated product term.**



$O_0 = \bar{I}_0 I_2 I_3 + I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3$

$O_1 = I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3 + \bar{I}_1 \bar{I}_3$

$O_2 = I_0 \bar{I}_1 + \bar{I}_1 \bar{I}_2 I_3 + I_0 I_2 I_3$

$O_3 = \bar{I}_0 I_2 I_3 + \bar{I}_1 \bar{I}_3$

$O_4 = I_0 I_2 I_3$

cubical representation

	I_0	I_1	I_2	I_3
C_0:	0	x	1	1
C_0^1:	1	0	x	x
C_2^1:	x	0	0	1
C_2^2:	x	0	x	0
C_4^3:	1	x	1	1

	O_0	O_1	O_2	O_3	O_4
D_0:	1	0	0	1	0
D_0^1:	1	1	1	0	0
D_2^1:	1	1	1	0	0
D_2^2:	0	1	0	1	0
D_4^3:	0	0	1	0	1

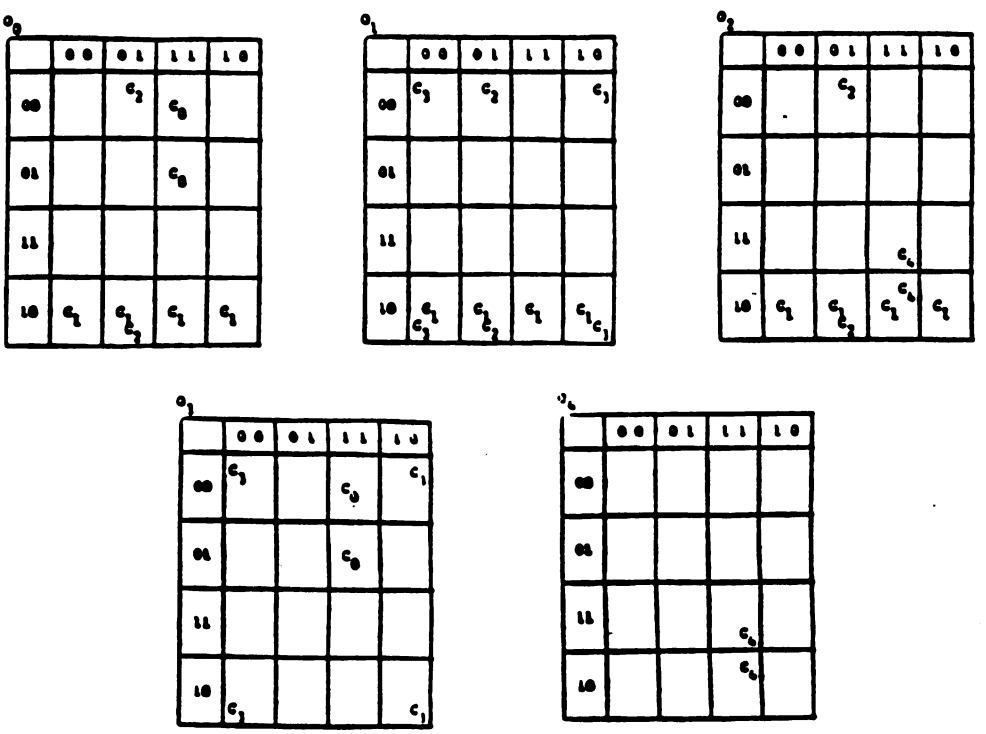


Figure 18 A schematic diagram of PLA and its K-maps.

5. For a given output function, a minterm covered by a product term is said to be free if it is not covered by any other product term of the function under consideration. Otherwise, it is said to be bound.
6. Two minterms covered by a product term are said to be adjacent if they differ in only one bit.
7. The number of bit positions in which two product terms differ is called the Hamming distance.
8. Let R_j, $j = 1,2,\ldots,p$, be the columns of D-array. A column R_k is said to be minimal if R_k does not cover any other columns. On the other hand, a column R_t is said to be maximal if R_t is not covered by any other columns. Note that neither minimal nor maximal column is unique.
9. Two rows of D-array, D_i and D_j, are said to bit-disjointed, if there does not exist a bit q such that $d_{iq}-d_{jq}=1$.

Notations:

Consider two product terms $C_i=(c_{i1},c_{i2},\ldots,c_{in})$ and $C_j=(c_{j1},c_{j2},\ldots,c_{jn})$. Let

$$C_i/\bar{c}_{jk}=(c_{i1},c_{i2},\ldots,\bar{c}_{jk}\ldots,c_{in}),$$

i.e., substituting the k-th bit of C_i by the complement of the k-th bit of C_j. Similarly,

$$C_i/\bar{c}_{ik}=(c_{i1},c_{i2},\ldots,c_{i(k-1)},\bar{c}_{ik},c_{i(k+1)},\ldots,c_{in})$$

5.1 Detecting Redundant Crosspoints from K-Maps

In this section, the objective is to derive a set of rules to detect the redundant crosspoints in both the AND and the OR planes. The removal of such redundant crosspoints will make the designed PLA G-D-irredundant. Therefore, all the G and D faults in the PLA can be detected. These detection rules are initially derived from the observations in the K-maps. A systematic derivation will be further discussed in the next section.

As mentioned in Chapter II, the "extra device" fault model in modern VLSI circuits has much less significance than other fault models normally considered for PLAs. Therefore, the test pattern generation process discussed in this chapter, is concentrated on both the Disappearance faults (D-fault) and Growth faults (G-faults).

As it is shown in Figure 2(a) and 2(c), a G-fault causes the additional minterms expand to its adjacent terms, while a D-fault makes some minterms disappear from the K-map.

Consider the irredundant crosspoints in the OR plane. As its name implied, the removal of an irredundant crosspoint will affect the output functions. In other words, some minterms are missing in the K-map. Conversely, the removal of a redundant crosspoint will not affect the output function at all and the K-map is unchanged. Based on this observation, it is easy to conclude that a crosspoint is not

redundant if its corresponding minterms are free. However, if the corresponding minterms are bound, the determination is discussed as follows.

Example 2:

Consider a PLA and its K-maps

	I_0	I_1	I_2		O_0	O_1	O_2
C_0:	0	1	x	D_0:	0	1	1
C_1:	0	0	1	D_1:	0	0	1
C_2:	0	1	0	D_2:	1	0	1
C_3:	1	0	x	D_3:	0	1	0

O_0:

	00	01	11	10
0				C_2
1				

O_1:

	00	01	11	10
0			C_0	C_0
1	C_3	C_3		

O_2:

	00	01	11	10
0		C_1	C_0	C_0 C_2
1				

From the above K-maps, it is easy to check that all minterms are free, except that the minterm (010) in O_2 is bound by the product terms C_0 and C_2. The redundancy is determined if there exists an order between the bound product terms. In other words, for an output function, if a product term associated with one of the bound terms is covered by any others' corresponding product terms, then the removal of such term won't affect the output function. Therefore, the corresponding crosspoints in the covered term are redundant.

Specifically, since the product term C_0=(01x) covers C_2=(010), the exclusive of the product term C_0 in O_2 may miss the minterm (011). On the other hand, the exclusive of the term C_2 in O_2, no minterm is missing. In this example, since C_0 covers C_2, the minterm (010) in O_2 is covered by C_0 regardless of the existence of C_2, hence, the crosspoint in C_2 and O_2, or d_{22}, is detected as redundant.

Rule RCD1: (Detecting the redundant crosspoints in the OR plane)

If a minterm in O_j is bound and a bound product term C_i is covered by other bound product terms, then the corresponding crosspoint, d_{ij} is redundant.

When an irredundant crosspoint occurs in the AND plane, then the removal of this crosspoint is equivalent to producing a G-fault. In other words, the additional minterms are created in the K-map when that crosspoint is removed. On the other hand, the created minterms due to the removal of the redundant crosspoints will be always bound by the other product terms. Based on this observation, if the corresponding adjacent minterms are free in, at least, one output function (or, K-map), then the corresponding crosspoint is irredundant. Otherwise, it is redundant.

Example 3:

Consider the same PLA in Example 2, except remove the redundant crosspoint d_{22} from the OR plane. The corresponding K-maps are

O_0:

	00	01	11	10
0				C_2
1				

O_1:

	00	01	11	10
0			C_0	C_0
1	C_3	C_3		

O_2:

	00	01	11	10
0		C_1	C_0	C_0
1				

From K-maps for both O_0 and O_1, the adjacent minterms of these minterms covered by C_0, C_2, and C_3, are free. According to the definition, the corressponding crosspoints are irredundant. On the other hand, consider the minterm (001) in the K-map for O_2, the free adjancent minterms at (000) and (101) guarantee that the corresponding crosspoints c_{10} and c_{12} are irredundant. However, due to the fact that the adjacent minterm at (011) is covered by C_0, it is concluded that the corresponding crosspoint c_{11} is redundant.

Rule RCD2: (Detecting the redundant crosspoints in the AND plane)

A crosspoint in the AND plane is redundant if the adjacent minterms of the corresponding minterm in all output functions (or, K-maps) are all covered by any other product terms.

With the redundancy detection process, the redundant crosspoints in both planes are removed to assure the PLA under test is a G-D-irredundant PLA.

5.2 Test Pattern Generation

In this section, the ways to select the minimum number of test patterns for detecting the G and D faults are discussed. As it is shown by Smith [Smi79], if T_s is the test set for detecting single G or D faults, then any detectable combination of G and D faults in a G-D irredundant circuit, is detected by T_s. Therefore, we only consider the test pattern generation for detecting single G and D faults.

In order to simplify the explanation of the test pattern generation process for both D-faults and G-faults, the basic concepts are introduced with K-maps.

A. Test Set for D-faults

Based on the following two observations: a D-fault makes some minterms disappear from the output functions (or, K-map), and the PLA under test contains no redundant crosspoint. A D-fault at the corresponding crosspoint is detected by simply examining if there exists a free minterm with respect to all output functions.

Specifically, from the K-maps, the test set for detecting a D-fault can be generated as follows:

Case DF1: if a product term is isolated, i.e. the minterms covered by the isolated product terms are free, then the product term is chosen as the test set;

Case DF2: if a product term is non-isolated, but there exists at least one common free minterm, then the common free minterms are chosen as the test set; and

Case DF3: if there is no common free minterm in a non-isolated product term, then the test set is formed by the minimum number of minterms which are free in all output functions.

Applying these three generation rules, the test set for D-faults in the K-maps of Figure 18 are generated in the following example. Let $PGD(D_i)$ be the test set for detecting the D-faults at D_i.

Example 4:

The PLA of Example 1 can be easily shown as G-D-irredundant by the redundancy detection rules. Since the product term C_0 is isolated as illustrated in the K-maps, according to the case DF1, the test set

is found to be $PGD(D_0)=C_0$, that is {(0011),(0111)}. In other words, either test pattern in these test sets can be applied to detect the D-faults at D_0. For the non-isolated product terms C_2, C_3, and C_4, by the case DF2, the common free minterms (0001), (00x0), (1111) can be chosen as the test sets for D_2, D_3, and D_4, respectively. Thus

$$PGD(D_2) = \{(0001)\}$$

$$PGD(D_3) = \{(0000),(0010)\}$$

$$PGD(D_4) = \{(1111)\}.$$

Although the non-isolated product term C_1 has no common free minterm, it can be found from the K-maps that the minterm (1000) is free in the output functions O_0 and O_2, but bound in O_1, while the minterm (1011) is free in O_0 and O_1, but bound in O_2. In other words, while the minterm (1000) can detect the D-faults at the corresponding crosspoints at O_0 and O_2, the blind point at O_1 can be detected by the minterm (1011). Therefore, with the use of these two test patterns the D-faults at D_1 can be detected. Alternately, another pair (1010) and (1011) can also be used as the test set. Therefore,

$$PGD(D_1) = \{\{(1011),(1000)\},\{(1011),(1010)\}\}.$$

The key to easily testable PLA design is the ability to select any arbitrary one product line during the test. If the only product line, say C_i, is activated, then the status of D_i is observable and the D-faults at D_i can be identified. Based on this concept, the test pattern generation problem is then turned to how to choose a test

pattern that can separate the product term C_i from other product terms.

As illustrated in the case DF1, if the product term C_i is isolated, then the minterms of C_i are free in all output functions of occurrence. This implied that the product term C_i has no common minterms with any others, or C_i is seperated from the others. In other words, the Hamming distance of any pair (C_i,C_j), for any other C_j's, is at least one. Therefore, the case DF1 is essentially equivalent to the case of Hamming distance of at least one.

Theorem 1:

If $d_H(C_i,C_j) \geq 1$, for all C_j's, then $PGD(D_i)=C_i$.

Lemma 1:

If there exists a test pattern that selects only the product line, C_i, then this pattern can detect the D-faults at D_i.

Proof:

Since only the product line C_i is selected, the status of the crosspoints at D_i is observable. Thus this test pattern can detect the D-faults at D_i.

Proof of Theorem 1:

Since $d_H(C_i,C_j) \geq 1$, for all C_j's, applying a test pattern of C_i will only select the line C_i and deselect the others. Thus, by Lemma 1, the test patterns of C_i form the set $PGD(D_i)$, i.e. $PGD(D_i)=C_i$.

Example 5:

Consider the same PLA as in the Example 1, it is easy to derive that

$$d_H(C_0,C_j)=1, \text{ for } j=1,2,3, \text{ and } 4,$$

Thus, the test set $PGD(D_0)=C_0=(0x11)$, or {(0111),(0011)}. This is the same as the test set generated in Example 4.

Consider the case of zero-Hamming distance. $d_H(C_i,C_j)=0$ implies that C_i and C_j contain at least one test pattern in common. In other words, applying the common test patterns may activate the product lines C_i and C_j simultanously. Therefore if the test patterns are chosen to separate the product term C_i from C_j, the common test patterns must be excluded. In fact, the determination of the exclusive test patterns depends on the crosspoints in the output functions. The test set generated for detecting the D-fault at D_i in the case of zero-Hamming distance should be derived from the test patterns generated for detecting D-fault at each crosspoint at D_i. For

notational simplicity, $d_H(C_i,C_j)=0$ is assumed for all C_j's in the PLA under test. Let R_j's, $j=1,2,..,p$, be denoted as the columns of the array D, and

$$S_{iq}= \{ C_j \mid d_H(C_i,C_j)=0 \text{ and } d_{iq} = d_{jq} = 1 \}.$$

Corollary 1.1:

If $S_{iq}=\{C_i\}$, then $PGD(d_{iq})=C_i$.

Proof:

$S_{iq}=\{C_i\}$ implies that either $d_H(C_i,C_j) \geq 1$, or d_{iq} is the only 1 in the column R_q. In the former case, by Theorem 1, $PGD(d_{1q})=C_i$. In the latter case, since C_i is the only product term in the output function O_q, it is obviously isolated, hence, from case DF1, $PGD(d_{iq})=C_i$.

If the set S_{iq} contains more than one product term, then the D-fault at d_{iq} may be masked by d_{jq}'s, unless C_i can be seperated from the other C_j's in S_{iq}. For the purpose of simplicity and clarity, the case of $S=\{C_i,C_j\}$ is first considered in Theorem 2, and the general case will be presented in Theorem 3.

Theorem 2:

If $S_{iq}=\{C_i,C_j\}$, then $PGD(d_{iq}) = \bigcup_k \{C_i/\bar{c}_{jk} \mid c_{ik}=x \text{ and } c_{jk} \neq x\}$.

Lemma 2:

If C_i is covered by C_j, then D_i and D_j are bit-disjointed.

Proof:

Assuming that D_i and D_j are not bit-disjointed, there exists a bit k such that d_{ik}-d_{jk}=1. Since C_i is covered by C_j, applying any test pattern to activate C_i would also activate C_j. This results in a redundant bit at d_{ik} and contradicts the assumption of irredundancy. Therefore, D_i and D_j are bit-disjointed.

In essence, the crosspoint at d_{iq} is detected as redundant if C_i is covered by C_j, and d_{iq}-d_{jq}=1. This is exactly the same as the redundancy detection rule RCD1.

Lemma 3:

If D_i and D_j are not bit-disjointed and $d_H(C_i,C_j)$=0, then there exists a bit k such that $c_{ik} = x$ and $c_{jk} \neq x$.

Proof:

Assume that there exists no such k, i.e. C_i is covered by C_j. By Lemma 2, D_i and D_j are bit-disjointed, contradict the assumption.

Lemma 4:

If S_{iq}=$\{C_i,C_j\}$, then there exists a bit k such that the test pattern from $C_i/\bar{c}_{jk}$ can detect the D-fault at d_{iq}.

Proof:

Since $S_{iq}=\{C_i,C_j\}$, i.e. $d_H(C_i,C_j)=0$ and $d_{iq}=d_{jq}=1$, or, D_i and D_j are bit-disjointed, by Lemma 3, there exists a bit k such that $c_{ik}=x$ and $c_{jk} \neq x$. Applying a test pattern from $C_i/\bar{c}_{jk}$ will activate only the line C_i and deactivate the others. By Lemma 1, the D-faults at d_{iq} can be detected by this pattern.

Proof of Theorem 2:

From Lemma 4, $C_i/\bar{c}_{jk}$ is one of the $PGD(d_{iq})$ at bit k. Therefore, $PGD(d_{iq})= U \{C_i/\bar{c}_{jk} \mid c_{ik} = x \text{ and } c_{jk} \neq x\}$.

Example 6:

Consider the following PLA,

C_1:	1	0	x	x		D_1:	1	1	1	0	0
C_2:	x	0	0	1		D_2:	1	1	1	0	0
C_3:	x	0	x	0		D_3:	0	1	0	1	0
C_4:	1	x	1	1		D_4:	0	0	1	0	1.

This PLA is the same as the PLA in Example 1, except the first product line is removed. It is easy to check that $d_H(C_1,C_j)=0$, for j=2,3,and 4, and $S_{11}=\{C_1, C_2\}$. The test set $PGD(d_{11})$ is generated as follows:

(1) Since $c_{13}=x$ and $c_{23}=0 \neq x$, the test patterns are selected from $C_1/\bar{c}_{23}=(c_{11}\ c_{12}\ \bar{c}_{23}\ c_{14})=(101x)$, or $\{(1010),(1011)\}$.

(2) Since c_{14}=x and c_{24}=1 ≠ x, the test patterns are selected from $C_1/\bar{c}_{24}$=(c_{11} c_{12} c_{13} $\bar{c}_{24}$)=(10x0), or {(1000),(1010)}.

By Theorem 2, the test set is the union of the test patterns generated in the both cases, or

$$PGD(d_{11})=TP_1=(101x)U(10x0)=\{(1000),(1010),(1011)\}.$$

Theorem 3:

If $S_{iq}=\{C_j \mid d_H(C_i,C_j)=0 \text{ and } d_{iq}=d_{jq}=1\}$, and $s = |S_{iq}| > 1$, then

$$PGD(d_{iq}) = TP_1 \cap TP_2 \cap \ldots \cap TP_s$$

where each TP_k, as shown in Theorem 2, is the test set generated for the pair (C_i,C_k).

Prior to the proof of Theorem 3, we consider Lemma 5.

Lemma 5:

If TP_1 and TP_2 are any two sets of test patterns generated for the pairs (C_i,C_j) and (C_i,C_t), respectively, then TP_1 and TP_2 have common test pattern(s).

Proof:

By Theorem 2, $TP_1 = U \{ C_i/\bar{c}_{jk} \mid c_{ik}=x \text{ and } c_{jk} \neq x \}$ and

$TP_2 = U \{ C_i/\bar{c}_{tr} \mid c_{ir}=x \text{ and } c_{tr} \neq x \}$.

Consider $C_i/\bar{c}_{jk}$ and $C_i/\bar{c}_{tr}$ subsets of TP_1 and TP_2, respectively. If k < r, then

$$Tp = (c_{i1},c_{i2},..,\bar{c}_{jk},..,\bar{c}_{tr},..,c_{in})$$

is a subset of both $C_i/\bar{c}_{jk}$ and $C_i/\bar{c}_{tr}$. In other words, TP_1 and

TP_2 have common test patterns. The same result can be obtained for $k > r$. On the other hand, for $k = r$, if $c_{jk} = \bar{c}_{tr}$ is assumed, then the "don't care" term c_{ik} is a logical OR of c_{jk} and c_{tr}. Since for all other bits, c_{iy} is covered by both c_{jy} and c_{ty}. Thus C_i is covered by $C_j \cup C_t$, resulting that the bit d_{iq} is redundant. Therefore, $c_{jk}=c_{tr}$, i.e., $C_i/\bar{c}_{jk}$ is the common subset of both TP_1 and TP_2.

Proof of Theorem 3:

Since each TP_j is a test set for the pair (C_i,C_j), this set consists of the test patterns which can seperate C_i from C_j. Therefore, the test pattern used to seperate C_i from others will be the intersection of the test sets TP_1, TP_2,..., and TP_s. From Lemma 5, it can be shown that this intersection is not empty.

Example 7:

Consider the PLA in Example 6. Since $S_{12}=\{C_1,C_2,C_3\}$ and s=2, there exist two don't cares in C_1. Therefore, the $PGD(d_{12})$ is generated as follows:

(1) From Example 6, $TP_1=(101x)U(10x0)=\{(1000),(1010),(1011)\}$.

(2) Similarly, for (C_1,C_3), $TP_2=(10x1)=\{(1001),(1011)\}$.

By Theorem 3, $PGD(d_{11}) = TP_1 \cap TP_2 = \{(1011)\}$.

Similarly, $PGD(d_{12})=(10x0)=\{(1000),(1010)\}$.

The test set $PGD(d_{iq})$ consists of the test patterns which are used to detect the D-fault at d_{iq}, where d_{iq}=1. Thus, the test set $PGD(D_i)$ is a collection of the test sets that consist of a test pattern from each $PGD(d_{iq})$, for all q. However, due to the relationship among the test sets $PGD(d_{iq})$'s, the generation process for the test set $PGD(D_i)$ can be simplified as follows.

Lemma 6:

If the column R_q covers R_j, then $PGD(d_{iq})$ is a subset of $PGD(d_{ij})$.

Proof:

Since R_q covers R_j, in other words, every pair of 1-valued entities in R_j will also be in R_q, then the test patterns generated for d_{iq} are always for d_{ij}, i.e., $PGD(d_{iq})$ is a subset of $PGD(d_{ij})$.

Example 8:

Since D_1 contains only three crosspoints at d_{10}, d_{11}, and d_{12}, the test set $PGD(D_1)$ is determined by the test sets $PGD(d_{1j})$, j=0, 1, and 2. Among the corresponding columns R_0, R_1 and R_2 ,it is found that the columns R_1 and R_2 are maximum, i.e. $PGD(d_{10})$ is a subset of both $PGD(d_{11})$ and $PGD(d_{12})$. Therefore, $PGD(D_i)$ is determined by these two subsets. Specifically, from Example 8,

$PGD(d_{11})=\{(1011)\}$

and $PGD(d_{12})=(10x0)=\{(1000),(1010)\}$.

then the element of the test set $PGD(D_i)$ is formed by selecting a pattern from both $PGD(d_{11})$ and $PGD(d_{12})$, or

$PGD(D_1)=\{\{(1011),(1000)\},\{(1011),(1010)\}\}$

This is exactly the same as the test set generated in the case DF3.

In fact, if the generated test sets, $PGD(d_{iq})$'s, are the same for each d_{iq} in D_i, i.e., $PGD(D_i)=PGD(d_{iq})$, then this is equivalent to the case DF2.

The test set for detecting the D-faults can be summarized in the following theorem,

Theorem 4:

Consider $S_i=\{C_j \mid d_H(C_i,C_j)=0\}$, and $s = |S_i|$.

(i) If $s = 1$, then $PGD(D_i)=C_i$, or

(ii) If $s > 1$, then the test set

$$PGD(D_i)=\{ \{p_1,..,p_k\} \mid \text{each } p_j,\ j=1,2,..,k, \text{ is selected from } PGD(d_{ij}) \text{ for all maximum columns } R_j \}. \quad (1)$$

Proof:

If s=1, then either $S_i=\{C_i\}$ or $d_H(C_i,C_j)\geq 1$. By Theorem 1, $PGD(D_i)=C_i$. On the other hand, if s>1, then the test set $PGD(D_i)$ is generated from the test sets $PGD(d_{iq})$'s that correspond to the maximal columns. The test set $PGD(D_i)$ is expressed as Equation (1).

Based on Theorem 4, the test pattern generation for D-fault is summarized in Algorithm PGD.

Algorithm PGD:

```
Step 1: (Test pattern generation)
        DO i=1 to m  (m is the number of product lines)
        BEGIN
           IF s=|S_i|=1, THEN  PGD(D_i)=C_i   ELSE
           BEGIN
              Determine the maximum columns R_iq's
              PGD(D_i) is generated by equation (1)^iq
           END
        END

Step 2: (Test pattern compaction)
        Eliminate the duplicated test patterns from PGD(D_i) for all
        i.
```

Example 9:

Consider the PLA in Example 1. Applying the Algorithm PGD, the test set for D-faults are generated as follows:

$PGD(D_0)$ = (0x11) = {(0011),(0111)},

$PGD(D_1)$ = {{(1011),(1010)}, {(1011),(1000)}},

$PGD(D_2)$ = {(0001)},

$PGD(D_3)$ = (00x0) = {(0000),(0010)}, and

$PGD(D_4)$ = {(1111)}.

Thus, {(0011),(1011),(1010),(0001),(0000),(1111)} is one of the test sets.

B. Test Set for G-faults

Let $PGG(c_{ik})$ be the test set for a G-fault at the crosspoint $c_{ik} \neq x$. As shown in the K-maps, a G-fault at c_{ik} may cause the additional logics to expand toward the corresponding direction, $\bar{c}_{ik}$, or to the adjacent minterms $C_i/\bar{c}_{ik}$. In fact, these adjacent minterms must be free. Otherwise, the corresponding crosspoints are redundant. Consequently, if an additional term is presented in one of these adjacent minterms, a G-fault at c_{ik} is detected. Specifically, the test set for detecting the G-fault at c_{ik} is generated as follows:

Case GF1: If the term $C_i/\bar{c}_{ik}$ is isolated with respect to any given output function, then $C_i/\bar{c}_{ik}$ is chosen as the test set for c_{ik}.

Case GF2: If the term $C_i/\bar{c}_{ik}$ is non-isolated, then the free minterms of $C_i/\bar{c}_{ik}$ are chosen as the test set for c_{ik}.

Example 10:

Consider the same G-D irredundant PLA and its K-maps (Figure 18). From the K-map for O_3, the minterms $C_0/\bar{c}_{00}$=(1x11) are all free. In other words, $C_0/\bar{c}_{00}$ is isolated with respect to O_3. Thus, by the case GF1, these minterms can be chosen as the test set to detect G-fault at c_{00}, or $PGG(c_{00}) = C_0/\bar{c}_{00} = (1x11)$. On the other hand, from the K-maps for O_0, O_1, and O_2, although the non-isolated minterms

$C_1/\bar{c}_{00}$=(00xx) has some bound minterms, it still contains several free minterms, such as (0000) and (0010) in O_0, (0011) in O_1, and (0000), (0010), and (0011) in O_2. Thus, PGG(c_{10})= {(0000),(0010),(0011)}.

Similarly, the remaining PGG(cik)'s are generated as shown in Table 2.

Table 2. Test Set for G-faults.

PGG(C_{ik})

	k: 0	1	2	3
i: 0	(1x11)	--	(0x01)	(0x10)
1	(00x0) (001x)	(11xx)	--	--
2	--	(x101)	(0011)	(0000)
3	--	(x1x0)	--	(x0x1)
4	(0x11)	--	(1x01)	(1x10)

Similar to the test set generation process for D-faults, the concept of Hamming distance is also applied here. The case GF1 is equivalent to the following theorem.

Theorem 5:

If $d_H(C_i/\bar{c}_{ik}, C_j) \geq 1$, for all C_j's, then $PGG(c_{ik})=C_i/\bar{c}_{ik}$.

Proof:

If $d_H(C_i/\bar{c}_{ik}, C_j) \geq 1$, for all C_j's, then the minterms $C_i/\bar{c}_{ik}$ are all free in all output functions. Consequently, $PGG(c_{ik})=C_i/\bar{c}_{ik}$.

Note that, if the crosspoint d_{iq} is the only one in the output function O_q, then the K-map with respect to O_q may contain only the product term C_i. Obviously, the adjacent minterms of C_i in the K-map are all free. In this special case, the test set is chosen as follows.

Theorem 6:

If d_{iq} is the only crosspoint in O_q, then $PGG(c_{ik})=C_i/\bar{c}_{ik}$, for all c_{ik}.

Proof:

Since the minterms $C_i/\bar{c}_{ik}$, for all c_{ik}, are all free, they can be chosen as test patterns, thus, $PGG(c_{ik})= C_i/\bar{c}_{ik}$.

Consider the case of $d_H(C_i/\bar{c}_{ik},C_j)=0$, this is equivalent to the case GF2. For notational simplicity, $d_H(C_i/\bar{c}_{ik},C_j)=0$ is assumed for all C_j's in the PLA under test. R_j's, j=1,2,..,p, are again denoted as the columns of array D. The $PGG(c_{ik})_q$ is denoted as the test set generated with respect to the output function O_q (or column R_q) for a G-fault at c_{ik}. This test set can be generated in a manner similar to Theorem 3.

Theorem 7:

Let $T_{ik} = \{ C_j \mid d_H(C_i/\bar{c}_{ik},C_j) = 0 \}$. If $t = |T_{ik}| > 0$, then

$$PGG(c_{ik})_q = TP_1 \cap TP_2 \cap \ldots \cap TP_t$$

where TP_j is the test set generated for the pair $(C_i/\bar{c}_{ik},C_j)$.

proof: It is the same as Theorem 3 by looking $C_i/\bar{c}_{ik}$ as C_i and t as s, respectively.

Example 11:

Consider the test set $PGG(c_{10})$ of the PLA in Figure 18. It can be easily check that $d_H(C_1/\bar{c}_{10},C_j)=0$, for j=0, 2, and 3. The cubical notation is rewritten as follows:

						O_0	O_1	O_2	O_3	O_4
$C_1/\bar{c}_{10}$:	0	0	x	x	D_1:	1	1	1	0	0
C_0 :	0	x	1	1	D_0:	1	0	0	1	0
C_2 :	x	0	0	1	D_2:	1	1	1	0	0
C_3 :	x	0	x	0	D_3:	0	1	0	1	0 .

In O_0, the test sets for the pairs $(C_1/\bar{c}_{10}, C_0)$ and $(C_1/\bar{c}_{10}, C_2)$ are TP_1= {(000x),(00x0)} and TP_2= {(001x),(00x0)}, respectively. Thus, $PGG(c_{10})_0 = TP_1 \cap TP_2$={(000x),(00x0)}∩{(001x),(00x0)}={(00x0)}. From the K-map for O_0, the adjacent minterms (0000) and (0010) of C_1 are free. They can be chosen as the test set regardless other adjacent minterms (0001) and (0011) are bound by C_2 and C_0, respectively.

Similarly, from either Theorem 7 or the K-maps, we can generate

$PGG(c_{10})_1$ = {(001x),(00x0)}∩ {(00x1)} = {(0011)} for O_1, and

$PGG(c_{10})_2$ = {(001x),(00x0)} for O_2.

Because each individual test pattern in $PGG(c_{ik})_q$ can detect the D-fault at c_{ik}, the test set $PGG(c_{ik})$ is then a collection of all elements in $PGG(c_{ik})_q$. Therefore,

$PGG(c_{10})$= $PGG(c_{10})_0$ U $PGG(c_{10})_1$ U $PGG(c_{10})_2$ = {(001x),(00x0)},

and we conclude that the test set $PGG(c_{ik}) = \bigcup_q \{ PGG(c_{ik})_q \}$.

It is inefficient and impractical to derive all poosible test sets for the corresponding outputs without simplification. In fact, from the above example,

$$PGG(c_{10}) = \{(001x),(00x0)\} = PGG(c_{10})_2.$$

This is due to the fact that the column R_2 covers both columns R_0 and R_1.

Lemma 8:

If column R_q covers R_j, then $PGG(c_{ik})_q$ is a subset of $PGG(c_{ik})_j$.

Proof:

Since the set $PGG(c_{ik})_r$ is the generated test set with respect to the output function O_r. The test set is generated by intersecting the test sets of pairs. If R_q covers R_j, in other words, the bits in R_j are also in R_q, then the more intersection is performed, the smaller set is obtained. Therefore, $PGG(c_{ik})_q$ is a subset of $PGG(c_{ik})_j$.

Based on the relationship derived in Lemma 8, the test set can be simplified as

$$PGG(c_{ik}) = \bigcup_q \{ PGG(c_{ik})_q \mid R_q\text{'s are the minimum columns} \}. \quad (2)$$

Theorem 8:

Consider $T_{ik} = \{ C_j \mid d_H(C_i/\bar{c}_{ik}, C_j)=0 \}$, and $t = |T_{ik}|$,

(i) if $t = 0$, then $PGG(c_{ik})=C_i/\bar{c}_{ik}$, or

(ii) if $t \geq 1$, then $PGG(c_{ik})$ is expressed in (2).

Proof:

If t=0, then $d_H(C_i/\bar{c}_{ik}, C_j) \geq 1$, by Theorem 5, $PGG(c_{ik})=C_i/\bar{c}_{ik}$. If $t \geq 1$, by Lemma 8, $PGG(c_{ik})$ can be expressed as the union of the test sets generated with respect to the minimum columns.

Corollary 8.1:

If d_{iq} is the only crosspoint in column R_q, then

$$PGG(c_{ik})=C_i/\bar{c}_{ik}.$$

Proof:

If d_{iq} is the only crosspoint in column R_q, then R_q is minimum, by Theorems 6 and 7, $PGG(c_{ik})=PGG(c_{ik})_q=C_i/\bar{c}_{ik}$.

Based on the above discussions, the test generation for G-faults is summarized in Algorithm PGG.

Example 12:

Consider the PLA in Example 1, applying the Algorithm PGG, the test set for D-fauls at each c_{ik} are the same as in Table 2. The test compaction process will eliminate the duplicated patterns. Therefore, a test set for both D-faults and G-faults in the PLA of Figure 18 is

{(0000),(0001),(0011),(0110),(1010),(1011),(1101),(1111)}.

Algorithm PGG:

Step 1: EIGEN(i)=false, i=1,2,..,m (m is the number of product lines)

Step 2: DO j=1 to p (p is the number of output lines)
BEGIN (Theorem 6)
IF d_{iq} is the only crosspoint in O_q, THEN
BEGIN
EIGEN(i)=true.
DO k=1 to n (n is the number of input lines)
BEGIN
IF $c_{ik} \neq x$ THEN $PGG(c_{ik})=C_i/\bar{c}_{ik}$.
END
END
END

Step 3: DO i=1 to m
BEGIN (Theorem 8)
IF (EIGEN(i)=false) THEN
BEGIN
DO k=1 to n
BEGIN
IF ($c_{ik} \neq x$) THEN
BEGIN
IF t=$|T_{ik}|$ = 0 THEN
$PGG(c_{ik})=C_i/\bar{c}_{ik}$
ELSE
BEGIN
determine the minimum columns R_q's
$PGG(c_{ik})$ is dereived as the equation(2).
END
END
END
END
END

Step 4: (Test Pattern Compaction)
Eliminate the duplicated test patterns from $PGG(c_{ik})$ and $PGD(D_i)$.

5.3. Simulation Results

The Proposed test pattern generation algorithm has been implemented on a VAX 11/780 in FORTRAN. In order to demonstrate the effectiveness of the proposed algorithm, the ten PLAs in [BoM84] have been simulated and a comparison of the number of test patterns required with other techniques [FuK81][SKF81][Kha83][SaT82][BoH84] are given in Table 3. The test length for each example shown in Table 3 is derived according to the heuristic that the test set is comprised by choosing every first test pattern of the test sets $PGD(D_i)$ and $PGG(c_{ik})$ and eliminating the duplicated test patterns. Therefore, the test length presented in Table 3 for the proposed algorithm can be improved if a better heuristic algorithm for test generation and compaction process is applied.

To collect experimental data, we also used testable versions of 49 PLAS that were used earlier in collecting data on efficacy of a PLA minimization program called ESPRESSO [BHM84]. After the PLA raw data are processed by ESPRESSO, the minimized PLAs with the test lengths are listed in Table 4, and these minimized PLA data are supposed to be irredundant. In fact, some redundant PLAs, as the entries marked with '*' in Table 4, are detected by our program. For example, 29 redundant crosspoints have been detected in the PLA 'cps' which has 29 input lines, 162 product lines, and 109 output lines.

Table 3. Comparison of the number of test patterns

Name	Fujiwara	Saluja	Khakbaz	Decoder	McCluskey	Chang
Master	104	902	515	594	540	51
New alu	102	871	496	572	520	83
bar new	95	598	398	528	462	62
recur	44	168	101	117	90	16
traffic	36	113	74	88	72	9
alu test	119	1105	650	792	684	87
cond	83	549	338	408	312	58
bar	87	530	350	435	337	64
rimp	119	1028	626	780	663	104
Cerber	159	1927	1102	1300	1150	160

Table 4. The number of test patterns required in some PLAs

Name	Ni	Np	No	Test Pattern	Name	Ni	Np	No	Test Pattern
adr4	8	75	5	90	root	8	57	5	101
alu1	12	19	8	8	sqn	7	38	3	68
alu2	10	68	8	73	sqr6	6	50	12	46
alu3	10	66	8	70	ti	47	213	72	346 *
ala	10	25	12	57	tial	14	579	8	1179 *
bc0	26	179	11	451*	vg2	25	110	8	178
bca	26	180	46	1445	wim	4	9	7	8
bcb	26	156	39	1270	x1dn	27	110	6	194
bcc	26	137	45	1143	x6dn	39	81	5	209 *
bcd	26	117	38	977	x9dn	27	120	7	194
chkn	29	140	7	505	z4	7	59	4	68
co14	14	14	1	106	in3	35	74	29	140 *
cps	24	162	109	919*	in4	32	212	20	394 *
dc1	4	9	7	13	in5	24	62	14	214
dc2	8	39	7	71	in6	33	54	23	183
dist	8	120	5	175*	in7	26	54	10	82
dk27	9	10	9	17	misg	56	69	23	43
dk48	15	21	17	34	mish	94	82	43	17
ex1	4	7	7	9	mlp4	8	127	8	164
f51m	8	76	8	74	opa	17	79	69	223
gary	15	107	11	345*	radd1	8	75	5	90
in0	15	107	11	347	rckl	32	32	7	529
in1	16	104	17	478	rd53	5	31	3	32
in2	19	135	10	342	rd73	7	127	3	128
					risc	8	28	31	39

Ni: the number of input lines.

Np: the number of product lines.

No: the number of output lines.

CHAPTER VI

Conclusion

Two testable PLA designs for both function-independent and function-dependent tests, have been presented.

The key to easily testable PLA design is the ability to select any arbitrary one product line during the test. This key concept has been implemented to design a product line activator in Chapter IV. It has been shown that the proposed design has the following salient features: (1) homogenous and regular structure; (2) less chip overhead; (3) no performance degradation during the normal operation due to the added hardware; (4) no additional I/O pin required; and (5) no extra test sequence needed.

The above key concept is also used to the test pattern generation in Chapter V. One of the major contributions in this approach is the introduction of the concept of Hamming distance to the test pattern generation. The test pattern generation problem is, therefore, turned in to the problem of how to choose a test pattern so that a product term can be seperated from others.

Based on the proposed algorithm, a software program has been implemented on a VAX 11/780 in Fortran. This program provides not only the generated test set, but also the information of redundant crosspoints in the PLA under test.

Since the "extra device" fault model in modern VLSI circuits has been less significant than other fault models normally considered for PLAs [KhB85], in Chapter V, the emphasis of the test pattern generation is on the detection of G and D faults. Also, as it has been shown, if T_s is the test set for detecting single G and D faults, then any detectable combination of G and D faults in a G-D irredundant PLA, is detected by T_s [Smi79]. Therefore, the proposed test generation for the single fault detection can essentially be applied for the multiple fault detection.

Although the proposed approach concentrated on the G and D faults, the same principle can be extended to the shrinkage and appearance faults, if necessary. However, this extension is held only if the combination of G and D faults and the combination of S and A faults do not occur simultaneously [Smi79].

As mentioned earlier, the motivation of doing this work is to reduce the hardware overhead for the test purpose in the design of fault-tolerant PLAs. Since the extra hardware overhead may offset the yield improvement, thus, the test generation approach that requires no hardware overhead may be superior to the testable design approach for this purpose. The development of an efficient test generation process with the minimal test length is of significient importance that leads to a future research direction.

In essence, the proposed pattern generation provides a potential to derive the near-minimal test length. Unlike the heuristic process applied to generate the test set in the existing generation

algorithms, the proposed algorithm is capable of generating all possible test patterns for each fault. Therefore, the derivation of the minimal test length can be accomplished by a method applied to derive the minimal covering set of fault-detection tests for the combinational circuits. Table 5 shows that all possible test patterns for each fault of the PLA in Example 1 are listed. The test set generated in Example 12 is essentially the minimal covering set of this table.

One of the most important issues in the design of fault-tolerant PLAs is fault-location. The faults must be located so that the spares can be efficiently allocated to repair the partially defective chips [WeL87]. The fault location problem had been overlooked for years until the fault-tolerant PLA being recently proposed. Research efforts have been devoted to the fault location problem.

It is possible to develop a fault location algorithm based on the proposed test generation approach. Again, since all possible test pattern(s) are generated, if the fault-location experiments [Koh78] developed for the combinational circuits are implemented, it is possible to generate the test set and set schedule to locate the fault which leads to an another future research direction.

Table 5. Test set for Figure 18.

	D 0	D 1	D 2	D 3	D 4	C 00	C 02	C 03	C 10	C 11	C 21	C 22	C 23	C 31	C 33	C 40	C 42	C 43
0000				1					1				1					
0001			1				1								1			
0010				1				1	1									
0011	1								1			1			1	1		
0100														1				
0101							1				1							
0110								1						1				
0111	1															1		
1000		1																
1001															1		1	
1010		1																1
1011		1				1							1					
1100										1				1				
1101										1	1						1	
1110										1				1				
1111					1	1				1								

LIST OF REFERENCES

[AbF86] Abraham, J. A. and W. K. Fuchs, "Fault and Error Models for VLSI", Proceedings of IEEE, Vol. 74, No. 5, pp. 639-654, May 1986.

[BHM84] Brayton, R.K., Hachtel, G.D., McMullen and A.L. Sangiovanni-Vincentelli, **Logic Minimization Algorithm for VLSI Synthesis**, Kluwer Academic Publishers, Hingham, MA, 1984.

[BoM84] Bozorgui-Nesbat, S. and E. J. McCluskey, "Lower Overhead Design for Testability of Programmable Logic Arrays", 1984 International Test Conference, pp. 856-865.

[CCH79] Cenker, R.P., Clemons, D.G., Huber, W.R., Petrizzi, J.B., Procyk, F.J., and G.M. Trout, "A Fault-tolerant 64k Dynamic RAM", IEEE Trans. on Electr. Dev., Vol. ED-26, No. 6, pp. 853-860, June 1979.

[Cha78] Cha, C. W., " A Testing Strategy for PLAs", Proc. 15th Design Automation Conference, pp. 326-334, 1978.

[DaM81] Daehn, W. and J. Mucha, "A Haredware Approach to Self-testing of Large Programmable Logic Arrays", IEEE Trans. on Computers, Vol.C-30, pp. 829-833, Nov. 1981.

[EiL80] Eichelberger, E. B. and E. Lindbloom, "A Heuristic Test-pattern Generator for Programmable Logic Array", IBM J. Res. & Dev., pp. 15-22, Jan. 1980.

[FKH80] Fujiwara, H., Kinoshita, K. and H. Ozaki, "Universal Test Sets for Programmable Logical Array", Proc. International Symposium on Fault-tolerant Computing, pp. 137-142, 1980.

[FuH86] Fung, H.S. and S. Hirschhorn, "An Automatic DFT System for the Slic Silicon Compiler", IEEE Design and Test, pp. 45-47, Feb. 1986.

[Fuj84] H. Fujiwara, "A New PLA Design for Universal Testability", IEEE Trans. on Computers, Vol. C-33, No. 8, pp. 745-750, Aug. 1984.

[FuK81] Fujiwara, H. and K. Kinoshita, "A Design of Programmable Logic Arrays with Universal Test", IEEE Trans. on Computers, Vol. C-30, No. 11, pp. 823-828, Nov. 1981.

[HaR85] Ha, D.S. and S.M. Reddy, "On the Design of Testable Domino PLAs", 1985 International Test Conference, pp. 567-573, 1985.

[HJA84] Hua, K.A., Jou, J.Y. and J.A. Abraham, "Built-In Tests for VLSI Finite-State Machines", Digest, Proc. 14th International Symposium on Fault-Tolerant Computing, pp. 292-297, 1984.

[Kha83] Khakbaz, J., "A Testable PLA Design with Low Overhead and High Fault Coverage", Proc. 13th International Symposium on Fault-Tolerant Computing, pp. 426-429, 1983.

[KhB85] Khakbaz, J. and S. Bozorgui-Nesbat, "Mimimizing Extra Hardware for Fully Testable PLA Design", International Conference on Computer Aided Design, pp. 102-104, 1985.

[KhM81] Khakbaze, J. and E. J. McCluskey, "Concurrent Error Detection and Testing for Large PLAs", CRC Tech. Rep. 81-14, Stanford Univ., Oct. 1982.

[Koh78] Kohavi, Z., **Switching and Finite Automata Theory**, McGraw-Hill, 1978.

[KoP86] Koren, I. and D.K. Pradhan, "Yield and Performance Enhancement through Redundancy in VLSI and WSI Multiprocessor Systems", Proceedings of the IEEE, Vol. 74, No. 5, pp. 699-711, May 1986.

[MAD82] Mak, G. P., Abraham, J. M. and E. S. Davidson, "The Design of PLAs with Concurrent Error Detection", Proc. 1982 International Test Conference, pp. 303-310.

[MeC80] C. Mead and L. Conway, **Introduction to VLSI system**, Addison-Wesley, 1980.

[Min84] Min, Y., "A PLA Design for Ease of Test Generation", Proc. 14th International Symposium on Fault-Torelant Computing, pp. 436-442, June 1984.

[Moo86] Moore, W.R., " A Review of Fault-Tolerant Techniques for the Enhancement Integrated Circuit Yield", Proceedings of the IEEE, Vol. 74, No. 5, pp. 684-689, May 1986.

[NeM83] J. Newkirk and R. Mathens, **The VLSI Designer's Library**, Addison-Wesley, 1983.

[OsH79] Ostapko D. L. and S. J. Hong, "Fault Analysis and Test Generation for Programmable Logic Array", IEEE Trans. on Computers, Vol. C-28, pp. 617-626, Sep. 1979.

[SaS86] Sami, M. and R. Stenfanelli, "Reconfigurable Architectures for VLSI Processing Arrays", Proceedings of the IEEE, Vol. 74, No. 5, pp. 699-711, May 1986.

[SaT82] Sato, T. and Y. Tohma, "A New Configuration of PLA with Functional Independent Test", Tech. Rep., Dept. of Computer Science, Tokyo Inst. of Technology, Tokyo, Japan, Oct. 1982.

[Sch78] Schuster, S.E., "Multiple Word/Bit Line Redundancy for Semiconductor Memories", IEEE J. Solid-State Circuits, SC-13, No. 5, pp. 698-703, 1978.

[SGM83] Somenzi F., Gai S., Mezzalama M. and P. Prinetto, "A New Integrated System for PLA Testing and Verification", IEEE 20th Design Automation Conference, pp. 57-63, 1983.

[SGM84] Somenzi, F., Gai S., Mezzalama M. and P. Prinetto, "PART:Programmable Array Testing Based on a Partitioning Algorithm", IEEE Trans. on Computer Aided Design, Vol. CAD-3, No. 2, pp. 142-149, April 1984.

[SKF81] Saluja, K.K., Kinoshita, K. and H. Fujiwara, "A Multiple Fault Testable Design of Programmable Logic Arrays", Proc. 11th International Symposium on Fault-Torelant Computing, pp. 44-46, 1981.

[SMD80] Stapper, C.H., Mclaren, A.N., and M. Dreckmann, "Yield Model for Productivity Optimization of VLSI Memory Chips with redundancy and parially good product", IBM J. Res. Dev., Vol.24, pp. 398-409, May 1980.

[Smi79] Smith, J. E., "Detection of Faults in Programmable Logic Arrays", IEEE Trans. on Computers, pp. 845-853, Nov. 1979.

[SoG86] Somenzi, F. and S. Gai, "Fault Detection in Programmable Logic Arrays", Proceedings of the IEEE, pp. 655-667, Vol. 74, No. 5, May 1986.

[SoP80] Son, K. and D. K. Pardhan, "Design of Programmable Logic Arrays for Testability", Proc. 1980 International Test Conference, pp. 163-166.

[Tas84] Tamir, Y. and C. H. Sequin, "Design and Application of Self-Testing Comparators Implemented with MOS PLA's", IEEE Trans. on Computer, Vol. C-33, No. 6, pp. 493-505, June 1984.

[TFA85] Treuer, R., Fujiwara, H. and V.K. Agrawal, "Implementing a Built-In Self-Test PLA Design", IEEE Design & Test, pp. 37-48, April 1985.

[WCV86] Wey, C.L., Chang, T.Y., and M.K. Vai, "On the Design of Fault-Tolerant Programmable Logic Arrays", Proc. of International Computer Symposium, Tainan, 1986, pp. 298-304.

[WeL86] Wey, C.L., and F. Lombardi, "On the Repair of Programmable Logic Arrays (RPLA)", Proc. 1986 IEEE International Symposium on Circuits and syatems, pp. 649-652, San Jose, CA. May 5-7, 1986.

[WVL86] Wey, C.L., Vai, M. K., and F. Lombardi, "On The Design of A Reduntant Programmable Logic Arrays (RPLA)", IEEE J. of Solid-State Circuits. (in press).

[Zhu86] Zhu, Xi-an, "A Knowledge-Based System for Testable Design Methodology Selection", Tech. Rep. CRI-86-23 U.S.C., Ph.D. dissertation.