

THE APPLICATION OF SPECTRAL TECHNIQUES  
TO THE DETECTION OF  
SINGLE AND MULTIPLE STUCK-AT FAULTS  
IN IRREDUNDANT COMBINATIONAL NETWORKS

by

Patrick Kam Lui

A thesis  
presented to the University of Manitoba  
in partial fulfillment of the  
requirements for the degree of  
MASTER OF SCIENCE  
in  
DEPARTMENT OF COMPUTER SCIENCE

Winnipeg, Manitoba, 1982

THE APPLICATION OF SPECTRAL TECHNIQUES  
TO THE DETECTION OF  
SINGLE AND MULTIPLE STUCK-AT FAULTS  
IN IRREDUNDANT COMBINATIONAL NETWORKS

BY

PATRICK KAM LUI

A thesis submitted to the Faculty of Graduate Studies of  
the University of Manitoba in partial fulfillment of the requirements  
of the degree of

MASTER OF SCIENCE

© 1983

Permission has been granted to the LIBRARY OF THE UNIVER-  
SITY OF MANITOBA to lend or sell copies of this thesis, to  
the NATIONAL LIBRARY OF CANADA to microfilm this  
thesis and to lend or sell copies of the film, and UNIVERSITY  
MICROFILMS to publish an abstract of this thesis.

The author reserves other publication rights, and neither the  
thesis nor extensive extracts from it may be printed or other-  
wise reproduced without the author's written permission.

(c) Patrick Kam Lui, 1982

## ABSTRACT

In this presentation, the fault diagnosis capabilities of Rademacher-Walsh spectral coefficients are investigated. The principal results which have appeared in the literature are reviewed. With this basis algorithms are developed for the derivation of minimal or near-minimal fault detection signatures for all single or multiple stuck-at faults in a single-output irredundant combinational network. The problem of fault location is also considered, and algorithms are described for deriving minimal or near-minimal fault location signatures that can diagnose single stuck-at faults to specified degree of diagnostic resolution. Although the fault diagnosis problem is attacked from a theoretical standpoint, many of the developed results are of practical importance.

## ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. Jon C. Muzio for his supervision throughout my thesis work, and to Jimmy C.H. Ho and Dr. David M. Miller for their many helpful comments. I am also thankful to Dr. Miller, Dr. Wooil Moon and Dr. Muzio for their careful examinations of my thesis copies. Last but not least, I wish to thank Dr. Hans W. Laale, Dr. Muzio and Peter K.L. Shum, without whose constant encouragement this thesis would never have been completed.

DEDICATION

To my family and friends

## CONTENTS

ABSTRACT . . . . .	ii
ACKNOWLEDGMENTS . . . . .	iii
DEDICATION . . . . .	iv

<u>Chapter</u>	<u>page</u>
I. INTRODUCTION . . . . .	1
II. THE RADEMACHER-WALSH TRANSFORM OF BOOLEAN FUNCTIONS	6
Introduction . . . . .	6
The Rademacher-Walsh Transform . . . . .	10
[0,1]-coding and [+1,-1]-coding . . . . .	14
logical interpretation of the spectrum . . . . .	16
Fast transform - the butterfly diagram . . . . .	19
Some special properties of a spectrum . . . . .	21
Parity of a spectrum . . . . .	22
Spectra of subfunctions - Tokmen's theorem . . . . .	23
Spectra of redundant functions . . . . .	26
Conclusion . . . . .	28
III. FAULT DETECTION - BACKGROUND . . . . .	30
Introduction . . . . .	30
Fault modelling of combinational logic networks . . . . .	33
Redundant networks . . . . .	34
Traditional and spectral testing approaches . . . . .	36
Traditional approach . . . . .	36
Spectral approach . . . . .	38
Traditional vs spectral . . . . .	40
Fault collapsing and minimal checkpoint . . . . .	41
Fault collapsing . . . . .	42
Minimal checkpoints . . . . .	46
Remark . . . . .	46
Fault location . . . . .	47
Conclusion . . . . .	49
IV. SYNDROME TESTING OF SINGLE STUCK-AT FAULTS . . . . .	51
Introduction . . . . .	51
Spectral conditions for syndrome-testability . . . . .	53
Unate lines . . . . .	58
Syndrome testable networks . . . . .	62

Tree networks . . . . .	63
NILE networks . . . . .	64
OIL networks . . . . .	65
Syndrome-testable design . . . . .	66
Testing general irredundant networks . . . . .	71
A multiple-output network example . . . . .	74
Conclusion . . . . .	76
V. SPECTRAL TESTING OF SINGLE STUCK-AT FAULTS . . . . .	77
Introduction . . . . .	77
Spectral conditions for testability . . . . .	78
Tree networks . . . . .	80
Fault detection signatures . . . . .	82
Minimal Spectral Signature . . . . .	82
Near minimal spectral signatures . . . . .	84
Fault location signatures . . . . .	90
Conclusion . . . . .	94
VI. SPECTRAL TESTING OF MULTIPLE STUCK-AT FAULTS . . . . .	96
Introduction . . . . .	96
Testability of input multiple faults . . . . .	98
A signature for input multiple faults . . . . .	101
Tree networks . . . . .	105
Internally fanout-free networks . . . . .	111
General networks . . . . .	124
Conclusion . . . . .	127
VII. SUMMARY AND FUTURE RESEARCH TOPICS . . . . .	129
BIBLIOGRAPHY . . . . .	135

### LIST OF FIGURES

<u>Figure</u>	<u>page</u>
2.1. Truth table of $f(x_3, x_2, x_1) = x_3x_2 + x_1$ . . . . .	7
2.2. The spectral transform requirement . . . . .	8
2.3. labelling coefficients of an example 3-variable function . . . . .	13
2.4. The topology of a fast transform for generating the spectrum of a 3-variable function . . . . .	20



3.1. Summary of fault equivalence relationships for  
basic gates . . . . . 45

4.1. functional model for an internal line g . . . . . 57

## Chapter I

### INTRODUCTION

One of the unfavourable outcomes of the widening application of digital computing systems is the increasing importance of ensuring that a computer system is operating correctly. For example, computer malfunctions in real-time applications such as process control and military command can prove to be disastrous. While fault-tolerant design[5] is usually used to mask faults in applications where the requirement for correct operation is stringent, it is important for any computing system to be periodically tested to determine if it is functioning properly. If any malfunction is detected, the faulty unit must be located and repaired or replaced.

This presentation is concerned only with the testing of the digital network components of the computing system rather than testing at the system level. In fact, we shall only consider single-output irredundant combinational networks.

The malfunctioning of a digital network may be due to an intermittent fault or a solid one. Although intermittent faults do occur in practice, they are very difficult to be tested for because their behaviours can vary with time. In the present presentation, we shall restrict our attention to

solid faults, which always exhibit themselves in the same manner.

During testing, the network under test(NUT) is often viewed as a black box with input and output terminals, and we attempt to detect faults by applying a sequence of input stimuli to the NUT and verifying the output responses. Clearly, any solid fault that causes a NUT to malfunction may be tested by applying all possible input combinations and verifying all output responses. Such an exhaustive approach is unappealing because of the progressive increases of testing time and data as the number of input terminals becomes large. To simplify testing, a fault model is assumed and the NUT is tested only for faults in the fault model. The most popular fault models to-date are the single and multiple stuck-at fault models, where one or more lines in the network are stuck at one of their two logical values of 0 and 1. It is well known that most solid faults that occur in practice can be represented by the stuck-at fault models.

Under the traditional testing approach, a subset rather than all of the input combinations are applied to the NUT and the corresponding output responses verified. This complete test set is selected so that the occurrence of any fault from the fault model will cause at least one input combination in the set to produce an erroneous output. The derivation of such a test set can be expensive and a large amount of data must be handled at test time.

A more recent approach chooses to apply all possible input combinations and compress the output response vector. The main advantage of this "response data compression"[15] approach over the traditional one is the reduction of data for the testing process. Spectral testing[38,39] is a testing technique based on response data compression. The output of the NUT, like the Boolean function it realizes, can be transformed to the Rademacher-Walsh spectral domain[24] where it is represented by a set of integers called the spectrum or spectral coefficients. A subset of these coefficients is selected to form a spectral signature[38,39] which can be used to detect or locate a fault from the fault model considered. During testing, all input combinations are applied and the coefficient values in the signature are collected and verified at the network output.

In this presentation, we shall describe the derivation of spectral signatures for irredundant combinational networks under the single or multiple stuck-at fault assumption. Chapters II and III are background chapters. Chapter II describes the Rademacher-Walsh transform of a Boolean function and discusses some properties of the spectrum instrumental to the development of results in later chapters. Chapter III introduces the basic concepts and terminology in fault diagnosis. Both of these chapters are essential for a good understanding of the remaining portion of this thesis.

Savir[45] has termed a fault syndrome-testable if its presence causes the network to realize a function with a different number of minterms from that of the intended function. In chapter IV, the published work on syndrome testing of single stuck-at faults are reviewed[38,39,45,51] and some improvements are made. A syndrome testable implementation of any irredundant Boolean function is also proposed.

Muzio and Miller[38,39] generalized Savir's syndrome testing method to that of spectral testing which makes use of coefficients other than the syndrome, the zero order coefficient, in the spectrum. Chapter V attempts to formalize the results presented in [38,39] by describing several single-fault detection signatures. The discussion is extended to the derivation of a spectral solution to the fault location problem, and Chang's method of distinguishability criteria[7] is modified to obtain a near-minimal single-fault location signature that can diagnose faults to a specified degree of diagnostic resolution.

Chapter VI extends the analysis of single faults to that of multiple faults, and represents the majority of research work performed for this presentation. A functional signature is developed to cover all input multiple faults. It is shown that a majority of irredundant functions, including any tree-realizable ones, have a signature length of one (coefficient). This functional signature is then shown to cover

many other multiple faults. With this signature as a basis, algorithms are developed to obtain a near-minimal spectral signature for detecting all multiple faults in any irredundant combinational network. Finally in chapter VII, the principal results that have been developed are summarized. The chapter concludes with an outline of some possible future research problems.

## Chapter II

### THE RADEMACHER-WALSH TRANSFORM OF BOOLEAN FUNCTIONS

#### 2.1 INTRODUCTION

A Boolean variable(switching variable) is one which can be assigned any one of the two values 0 and 1. A Boolean function(switching function)  $f(x_n, \dots, x_1)$  on the  $n$  input Boolean variables  $x_n, \dots, x_1$  is a mapping which associates each of the  $2^n$  possible assignments of  $x_n, \dots, x_1$  to any one of 0 and 1[28]. An assignment of  $x_n, \dots, x_1$  is said to be an input combination to the function, and the value after the mapping is termed the output value of the function corresponding to that input combination.

A Boolean function may be represented by a switching expression based on switching algebra[28], or by a truth table which tabulates its outputs for all possible input combinations. Consider as an example the following truth table for the Boolean function  $f(x_3, x_2, x_1) = x_3x_2 + x_1$  :

binary input			function output
$x_3$	$x_2$	$x_1$	$f(x_3, x_2, x_1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 2.1: Truth table of  $f(x_3, x_2, x_1) = x_3x_2 + x_1$

The binary input combinations are usually ordered in ascending order with respect to the binary number system. If this ordering is observed, the column vector of function outputs is termed the characteristic sequence(CHS) of the function.

The truth table representation is rather restrictive, in the sense that each tabulated output entry gives us no information about the function output for a different input combination. This restriction prohibits us from recognizing certain characteristics of interest for the entire function by looking at just a few numbers. Indeed, for a  $n$ -variable function, the complete CHS of  $2^n$  entries is necessary to fully define the function.



In view of the above restriction, one may want to re-express a Boolean function as another set of numbers each of which contains a higher information content than the two values of 0 and 1. This suggests some sort of mathematical transform operation, such as matrix multiplication, to the output CHS. The new set of numbers formed from such a transform is called the spectrum or spectral coefficients of the function. The methods of deriving and applying these numbers is referred to as spectral techniques[24]. Figure 2.2 illustrates the requirement for this transformation.

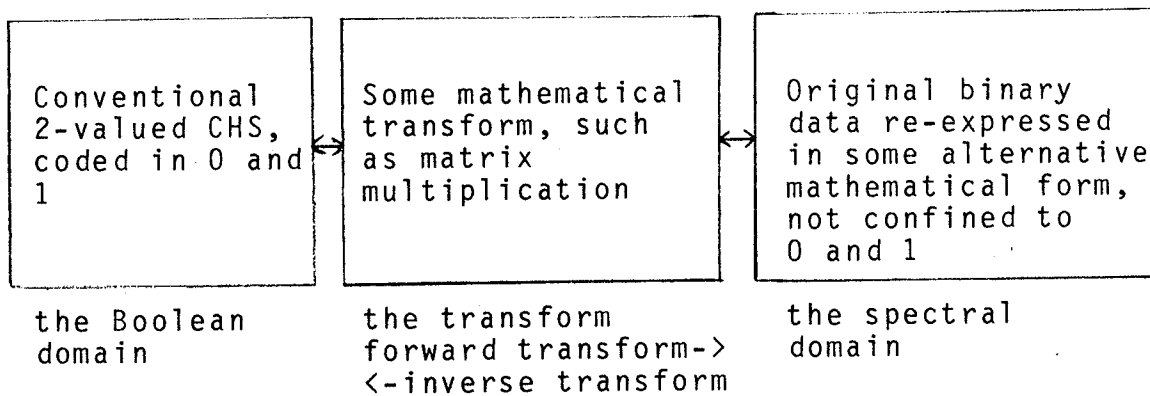


Figure 2.2: The spectral transform requirement

The most suitable transforms for this purpose are orthogonal transforms[24], which use non-singular (square) matrices. They are chosen mainly because :

- 1) Matrix multiplication for such a transform may be performed with  $N \times \log_2 N$  or fewer arithmetical opera-

tions for an  $N \times N$  matrix, compared with  $N^2$  generally necessary.

- 2) If the matrix can be constructed from +1's and -1's, then maximum computational simplicity can be achieved (i.e. only additions/subtractions, no multiplication is needed).
- 3) If the matrix and its inverse can be made identical except for some constant normalizing factor, then the same computational procedure may be used for the forward or the inverse transform.
- 4) Because of the orthogonality of such a transform, the  $2^n$  spectral coefficients obtained for a  $n$ -variable function have dissimilar information content from one another, so that each individual coefficients reflects some particular characteristics of the whole function.

There are several variations of the orthogonal matrix transforms[22,32], the one that we shall be using is based on those of Rademacher[42] and Walsh[54]. In this chapter, we shall describe the Rademacher-Walsh transform[24] of Boolean functions. We begin by defining the Rademacher-Walsh transform matrix. Next the transform procedure together with a labelling scheme of the individual coefficients after the transform are described. The spectra obtained from two different coding schemes of the CHS (the [0,1]-coding and the [+1,-1]-coding) are then compared, and the logical meaning

of each spectral coefficient is interpreted. An efficient algorithm for evaluating the transform is then given. Finally, three special properties of the spectrum, namely the parity, the spectra of subfunctions, and the spectra of redundant functions, are described.

## 2.2 THE RADEMACHER-WALSH TRANSFORM

The Rademacher-Walsh transform is an orthogonal matrix transform. It is so named because the rows of the transform matrix are made up of Rademacher-Walsh functions[24]. The Rademacher-Walsh transform matrix of order  $n$ , denoted by  $T^n$ , is a  $2^n$  by  $2^n$  square matrix defined recursively as :

$$T^n = \begin{bmatrix} T^{n-1} & T^{n-1} \\ T^{n-1} & T^{n-1} \end{bmatrix} ; \quad T^0 = [1] \quad . . . . . (2.1)$$

By definition,  $T^n$  consists of +1 and -1 entries only. The Rademacher-Walsh matrices of orders  $n = 1, 2$  and 3 are :

$$T^1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$T^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$T^3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

We shall denote the  $(i+1)$ -th column (row) of  $T^n$ ,  $i=0(1)2^n-1$ , by the column (row) vector  $t_{.i}^n$  ( $t_{i.}^n$ ). From now on, if it is clear by context what the order of the transform matrix is, we shall drop the superscript  $n$  and use  $T$ ,  $t_{.i}$ ,  $t_{i.}$  instead.

Several important properties of  $T$  are :

1) Symmetry

$$t_{i.} = (t_{.i})^t \quad (\text{or } T = T^t) \quad \dots \dots \dots (2.2.1)$$

where superscript  $t$  denotes transpose.

2) Orthogonality

$$(t_{i.}, t_{j.}) = \begin{cases} 2^n & i=j \\ 0 & i \neq j \end{cases} \quad \dots \dots \dots (2.2.2)$$

where  $(u,v)$  is the "inner product" of the two vectors  $u$  and  $v$  (i.e. the sum of multiplying entries in  $u$  with corresponding entries in  $v$ ).

3) Inverse

$$T^{-1} = \left(\frac{1}{2}\right)^n T \quad \dots \dots \dots (2.2.3)$$

Properties 1) and 2) are clear by definition. Property 3) follows directly from 1) and 2).

Let us proceed to describe the transform procedure. Let the  $2^n \times 1$  column vector  $\underline{f}$  be the CHS of a Boolean function  $f(x_n, \dots, x_1)$ , then the forward transform is mathematically given by

$$T^n \underline{f} = R \quad \dots \dots \dots (2.3)$$

The resultant column vector  $R$  is the Rademacher-Walsh spectrum of  $f$ . since  $T^n$  is non-singular, a unique inverse transform matrix exists (cf.(2.2.3)), and  $R$  uniquely identifies  $f$ . Applying the transform (2.3) to the Boolean function in figure 2.1, we have

$$\begin{array}{c}
 \left[ \begin{array}{cccccccc}
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
 \end{array} \right]
 \begin{array}{c}
 \left[ \begin{array}{c}
 0 \\
 1 \\
 0 \\
 1 \\
 0 \\
 1 \\
 1 \\
 1
 \end{array} \right]
 =
 \begin{array}{c}
 \left[ \begin{array}{c}
 5 \\
 -3 \\
 -1 \\
 -1 \\
 -1 \\
 -1 \\
 1 \\
 1
 \end{array} \right]
 \end{array}
 \end{array}$$

$T^3 \quad \underline{f} = R$

The inverse transform of (2.3) is given by

$$[T^n]^{-1} R = \underline{f}$$

and by (2.2.3), we have

$$\left(\frac{1}{2}\right)^n T^n R = \underline{f} \quad \dots \dots \dots (2.4)$$

In order to single out a spectral coefficient from among the  $2^n$  coefficients in the spectrum, each coefficient is assigned a unique label. Three commonly used labelling schemes are illustrated in Figure 2.3 for the previous example of a 3-variable function.

binary input			output	spectrum	labelling schemes		
$x_3$	$x_2$	$x_1$	$f$	$R$	I	II	III
0	0	0	0	5	$r_{000}$	$r_0$	$r_0$
0	0	1	1	-3	$r_{001}$	$r_1$	$r_1$
0	1	0	0	-1	$r_{010}$	$r_2$	$r_2$
0	1	1	1	-1	$r_{011}$	$r_3$	$r_{12}$
1	0	0	0	-1	$r_{100}$	$r_4$	$r_3$
1	0	1	1	-1	$r_{101}$	$r_5$	$r_{13}$
1	1	0	1	1	$r_{110}$	$r_6$	$r_{23}$
1	1	1	1	1	$r_{111}$	$r_7$	$r_{123}$

Figure 2.3: labelling coefficients of an example 3-variable function

In labelling scheme I, the ascending  $n$ -digit binary number sequence is used to label the coefficients. Contrary to what this might have suggested, there is no special rela-

tionship between a spectral coefficient and the input combination identical to its label. The labels in scheme II are just the decimal equivalences of those in scheme I. In scheme III, the position(s) from right to left (and starting from 1) of the 1's in the binary label from scheme I constitutes the label. We shall use scheme III exclusively in this presentation. As we shall see later in this chapter, this labelling scheme associates each spectral coefficient with a set of input variables in such a way that each coefficient describes a well-defined characteristic of the function with respect to the input variables in the coefficient label.

### 2.3 [0,1]-CODING AND [+1,-1]-CODING

In the preceding description, the output CHS consists of 0's and 1's. One can recode the CHS by replacing 0's with +1's and 1's with -1's, and then perform the same transform (2.3) to obtain the spectrum for [+1,-1]-coding. Most literature on spectral techniques uses the [+1,-1]-coding. However, the majority of published papers on fault detection by spectral or similar methods chose the [0,1]-coding. This is because the resulting expressions developed for fault detection purposes are simpler in [0,1]-coding than in [+1,-1]-coding. Thus we shall also adopt the [0,1]-coding scheme in this presentation. Since the spectra obtained using the two different codings are equivalent and interchangeable, it is easy to convert the results to be present-

ed in  $[0,1]$ -coding into their equivalences in  $[+1,-1]$ -coding.

Let us describe how the conversion can be performed. Define

- $\underline{f}$  as the CHS of  $f(x_n, \dots, x_1)$  in  $[0,1]$ -coding,
- $\underline{g}$  as the CHS of  $f(x_n, \dots, x_1)$  in  $[+1,-1]$ -coding,
- R as the spectrum obtained from  $\underline{f}$ , with entries  $r_0, r_1, r_2, r_{12}, \dots, r_{1..n}$ ,
- S as the spectrum obtained from  $\underline{g}$ , with entries  $s_0, s_1, s_2, s_{12}, \dots, s_{1..n}$ .

then the following results are obvious by definition :

Result 2.1

$$1) S = T^n \underline{g} \dots \dots \dots (2.5)$$

$$\underline{g} = \left(\frac{1}{2}\right)^n T^n S \dots \dots \dots (2.6)$$

$$2) \underline{g} = 1 - 2\underline{f} \dots \dots \dots (2.7)$$

$$s_\alpha = \begin{matrix} 2^n - 2r_\alpha, & \alpha=0 \\ -2r_\alpha, & \alpha \neq 0 \end{matrix} \dots \dots \dots (2.8)$$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

From now on, we shall use R  $(r_0, r_1, r_2, r_{12}, \dots, r_{1..n})$  to denote a spectrum from  $[0,1]$ -coding, and S  $(s_0, s_1, s_2, s_{12}, \dots, s_{1..n})$  from  $[+1,-1]$ -coding. Unless otherwise specified, the unqualified terms "CHS", "spectrum",



"spectral coefficients" etc. are assumed to be for [0,1]-coding.

#### 2.4 LOGICAL INTERPRETATION OF THE SPECTRUM

The transform matrix  $T^n$  is made up of rows each of which is a Rademacher-Walsh function[24]. They are labelled as  $R_0, R_1, R_2, R_{12}, \dots, R_{1..n}$  going down  $T^n$ . In fact, the labelling schemes for the spectral coefficients as described in section 2.2 arises from this Rademacher-Walsh function labelling, since

$$R_\alpha \underline{f} = r_\alpha \dots \dots \dots (2.9)$$

If we convert all the +1,-1 values of  $R_\alpha$  into the more familiar context of 0's and 1's, and interpret it as the CHS of a n-variable function  $f(x_n, \dots, x_1)$ , then it can be shown that

- 1)  $R_0$  is the CHS of the constant zero function,
- 2)  $R_i, i=1,2,\dots,n$ , is the CHS of the function  $x_i$ ,
- 3)  $R_\alpha, \alpha=12,13,\dots,1..n$ , is the CHS of the function formed from the Exclusive-or of the inputs contained in  $\alpha$ .

One easily verifies the case for a 3-variable function to obtain the following relationship :

R-W function	equivalent Boolean function
$R_0$	0
$R_1$	$x_1$
$R_2$	$x_2$
$R_{12}$	$x_1 \oplus x_2$
$R_3$	$x_3$
$R_{13}$	$x_1 \oplus x_3$
$R_{23}$	$x_2 \oplus x_3$
$R_{123}$	$x_1 \oplus x_2 \oplus x_3$

Consider again the transform of the general  $n$ -variable function  $f(x_n, \dots, x_1)$ . Suppose  $\underline{g}$  is the CHS of  $f$  in  $[+1, -1]$ -coding. If we look at the mathematics of our  $+1, -1$  transform and consider what happens in generating the final spectral coefficients, we shall see that the matrix multiplication (2.5) is effectively computing the number of agreements minus the number of disagreements of the values in the column vector  $\underline{g}$ , which represents the given function  $f$ , with the rows of the transform matrix  $T^n$ . Hence we may define the resultant values of the spectral coefficients as [24] :

$$\begin{aligned}
 s_0 &= (\text{number of } +1\text{'s in } \underline{g}) - (\text{number of } -1\text{'s in } \underline{g}) \\
 &= (\text{number of false minterms in } f) \\
 &\quad - (\text{number of true minterms in } f) \quad \dots \dots (2.10)
 \end{aligned}$$

$$s_\alpha = (\text{number of agreements between } \underline{g} \text{ and } R_\alpha) - (\text{number of disagreements between } \underline{g} \text{ and } R_\alpha)$$

Breaking down the latter into the primary(first order) coefficients  $s_1$  to  $s_n$  and the remaining (higher order) secondary coefficients, we have

$$s_i, i=1, \dots, n = \begin{aligned} & \text{(number of agreements between} \\ & \text{f and the input variable } x_i) \\ & - \text{(number of disagreements between} \\ & \text{f and the input variable } x_i). \dots (2.11) \end{aligned}$$

and

$$s_\alpha, \alpha=12, \dots, 1..n = \begin{aligned} & \text{(number of agreements between} \\ & \text{f and the XOR of the variables} \\ & \text{contained in } \alpha) \\ & - \text{(number of disagreements between} \\ & \text{f and the XOR of the variables} \\ & \text{contained in } \alpha) \dots (2.12) \end{aligned}$$

The less obvious interpretation of the spectrum R from the [0,1]-coded CHS  $\underline{f}$  is :

$$\begin{aligned} r_0 &= \text{(number of 1's in } \underline{f}) \\ &= \text{(number of minterms realized by f)} \dots (2.13) \end{aligned}$$

$$r_i, i=1, \dots, n = \begin{aligned} & \text{(number of 1's in } \underline{f} \text{ when } x_i=0) \\ & - \text{(number of 1's in } \underline{f} \text{ when } x_i=1) \dots (2.14) \end{aligned}$$

$$r_\alpha, \alpha=12, \dots, 1..n = \begin{aligned} & \text{(number of 1's in } \underline{f} \text{ when} \\ & \text{the XOR of the variables} \\ & \text{in } \alpha \text{ yields 0)} \\ & - \text{(number of 1's in } \underline{f} \text{ when} \\ & \text{the XOR of the variables} \\ & \text{in } \alpha \text{ yields 1)} \dots (2.15) \end{aligned}$$

Thus each spectral coefficient value  $s_\alpha$  gives a numerical measure of how "like" the function output is to its inputs, or appropriate combinations of its inputs. Similarly,  $r_\alpha$  measures how "unlike" the function output is to its inputs

or combinations of inputs. The ranges of  $s_\alpha$  and  $r_\alpha$  are as follows :

$$1) -2^n \leq s_\alpha \leq 2^n \quad \text{for all } \alpha \quad . . . . . (2.16)$$

$$2) \quad 0 \leq r_0 \leq 2^n$$

$$-2^{n-1} \leq r_\alpha \leq 2^{n-1} \quad \text{for all } \alpha \neq 0 \quad . . . . . (2.17)$$

## 2.5 FAST TRANSFORM - THE BUTTERFLY DIAGRAM

If the spectral coefficient values for a simple Boolean function are to be computed by hand, it will become apparent that the computation involves a number of product terms which are common to several coefficients. We can make use of this fact to reduce the number of product terms to be summed to  $n \times 2^n$  instead of the normal  $2^n \times 2^n$ . This reduction can be achieved by means of a fast transform. For example, consider a 3-variable Boolean function with CHS (a b c d e f g h)<sup>t</sup>. The direct forward transform is given by the matrix multiplication (2.3) as :

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_{12} \\ r_3 \\ r_{13} \\ r_{23} \\ r_{123} \end{bmatrix}$$

$T^3 \quad \underline{f} = R$

The corresponding fast transform is graphically represented in figure 2.4, which is often referred to as the butterfly diagram[22,48,53].

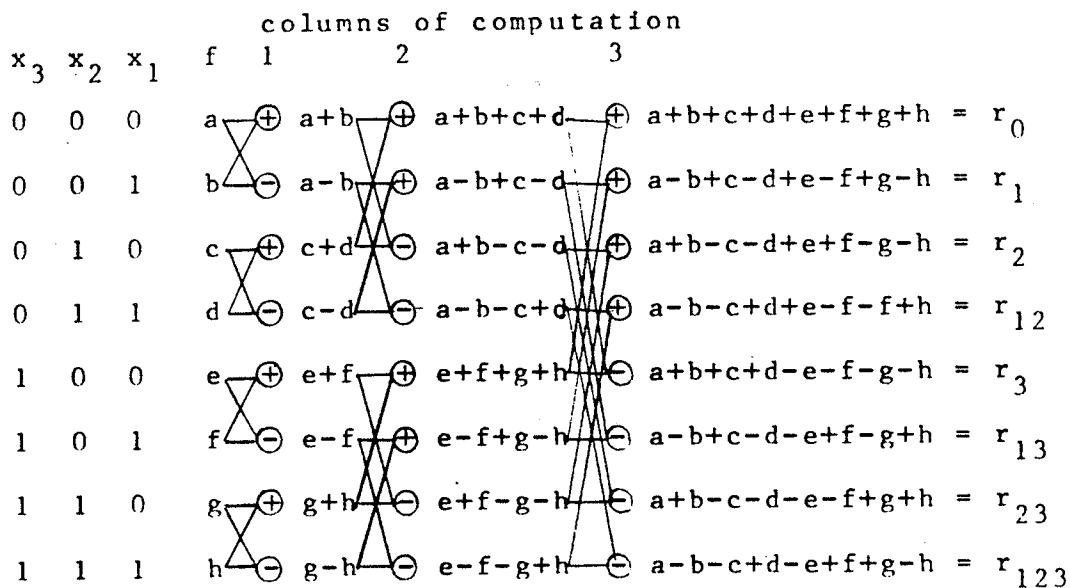


Figure 2.4: The topology of a fast transform for generating the spectrum of a 3-variable function

By virtue of the recursive definition of  $T^n$  given in (2.1), we can show that for  $n=2$ , we delete the bottom half of the schematic in figure 2.4 and need only 2 instead of 3 columns of computation. For  $n=4$ , we duplicate this schematic and add a fourth column of computation, and so on. The +'s and -'s operations in each column follows a simple order -- for the  $i$ th column, we have  $2^{i-1}$  +'s followed by  $2^{i-1}$  -'s, and this pattern repeats itself down the column if necessary. Thus for a butterfly diagram of order  $n$ , there should be  $2^n$  rows and  $n$  columns (of computation), with the first column consisting of alternating +'s and -'s, and the last column consisting of  $2^{n-1}$  +'s followed by  $2^{n-1}$  -'s. Since there are  $2^n$  product terms involved in each of the  $n$  computational columns, the total number of product terms computed is  $n \times 2^n$  instead of  $2^n \times 2^n$ .

## 2.6 SOME SPECIAL PROPERTIES OF A SPECTRUM

In this section, we shall describe three special properties of the spectrum of a Boolean function. We first show that the coefficients in a spectrum are either all odd or all even. We then state Tokmen's theorem[51] which relates the spectra of a function with its subfunctions. Lastly, a redundant function is defined and its spectrum described. These properties are instrumental in the development of many results in later chapters.

### 2.6.1 Parity of a spectrum

As previously mentioned, the  $2^n$  coefficients in a spectrum are dissimilar, in the sense that they measure the "likeness" of the represented function to some dissimilar functions. However, the coefficients are not independent from one another. One property common to all  $2^n$  coefficient values is their parity, as stated in the following theorem :

Theorem 2.1 The spectral coefficients  $r_0$  to  $r_{1..n}$  of a  $n$ -variable Boolean function  $f(x_n, \dots, x_1)$  is either all odd or all even.

Proof  $r_0$  is the number of 1's in  $\underline{f}$ , the CHS of  $f$ . Now to obtain  $r_\alpha$  ( $\alpha \neq 0$ ) using (2.14) or (2.15), the 1's in  $\underline{f}$  are partitioned into 2 disjoint subsets, and  $r_\alpha$  computed as the difference of their respective sums. If  $r_0$  is odd, these two sums have different parities and their difference must be odd, so that  $r_\alpha$  is also odd. On the other hand, if  $r_0$  is even, these two sums have equal parity and  $r_\alpha$  must also be even.  $\Omega$

A function is said to be an odd (even) function if its spectrum has odd (even) parity.

### 2.6.2 Spectra of subfunctions - Tokmen's theorem

Let  $f(x_n, \dots, x_1)$  be a Boolean function, we define its subfunction by fixing the last  $n-m$  inputs  $x_n, \dots, x_{m+1}$  respectively at  $u_{n-m}, \dots, u_1$  ( $u_i \in [0,1]$ ) as

$$f_u(x_m, \dots, x_1) = f(u_{n-m}, \dots, u_1, x_m, \dots, x_1)$$

where  $0 \leq u \leq \beta = 2^{n-m}-1$  and  $u$  is an integer such that

$$u = \sum_{i=1}^{n-m} u_i 2^{i-1}$$

Conversely,  $f$  is said to be a superfunction of  $f_u$  by adding the  $n-m$  extra inputs  $x_n, \dots, x_{m+1}$ . the terms subspectrum and superspectrum are similarly defined.

Given the spectrum  $R$  of  $f$ , we can compute from it the subspectrum  $R_u$  of  $f_u$  as follows. We first partition  $R$  into  $(\beta+1)$  subvectors  $R^0, R^1, \dots, R^\beta$ , i.e.

$$R = [(R^0)^t \ (R^1)^t \ \dots \ (R^\beta)^t]^t$$

each subvector  $R^i$  will then contain all those coefficients involving only a fixed subset of  $x_n, \dots, x_{m+1}$ . Thus  $R^0$  contains coefficients that do not involve any of  $x_n, \dots, x_{m+1}$ ;  $R^1$  contains those that involve  $x_{m+1}$  but none of  $x_n, \dots, x_{m+2}$ , etc.; and finally,  $R^\beta$  contains coefficients involving all of  $x_n, \dots, x_{m+1}$ . Then  $R$  and  $R_u$  are related by the following theorem by Tokmen[51] :



Theorem 2.2 [51]

$$[R_0 \ R_1 \ \dots \ R_\beta] = \left(\frac{1}{2}\right)^{n-m} [R^0 \ R^1 \ \dots \ R^\beta] T^{n-m} \quad \dots \quad (2.18)$$

or conversely

$$[R^0 \ R^1 \ \dots \ R^\beta] = [R_0 \ R_1 \ \dots \ R_\beta] T^{n-m} \quad \dots \quad (2.19)$$

Proof See [51].  $\Omega$

We shall illustrate the theorem for the previous example function  $f(x_3, x_2, x_1) = x_3 x_2 + x_1$ . Fixing  $x_3$ , we have  $f_0(x_2, x_1) = x_1$ ;  $f_1(x_2, x_1) = x_2 + x_1$ . Let  $\underline{f}_0, \underline{f}_1$  be the CHS of  $f_0$  and  $f_1$  respectively, construct the following truth table :

$x_2$	$x_1$	$\underline{f}_0$	$\underline{f}_1$
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	1

Applying the forward transform (2.3) to  $\underline{f}_0$  and  $\underline{f}_1$ , we have

$$\begin{aligned} R_0 &= T^2 \underline{f}_0 = [2 \ -2 \ 0 \ 0]^t \\ R_1 &= T^2 \underline{f}_1 = [3 \ -1 \ -1 \ -1]^t \quad \dots \quad (2.20) \end{aligned}$$

Now from figure 2.3,

$$R = [5 \ -3 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1]^t = [(R^0)^t \ (R^1)^t]^t$$

so that

$$R^0 = [5 \ -3 \ -1 \ -1]^t \ ; \ R^1 = [-1 \ -1 \ 1 \ 1]^t$$

and applying Tokmen's theorem in (2.19) :

$$\begin{aligned} [R_0 \ R_1] &= \left(\frac{1}{2}\right) [R^0 \ R^1] T^1 \\ &= \frac{1}{2} \begin{bmatrix} 5 & -1 \\ 3 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 3 \\ -2 & -1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} \end{aligned}$$

as in (2.20).

Similarly, we can also verify the theorem by fixing two variables  $x_3$  and  $x_2$ .

By re-arranging the variables in  $f(x_n, \dots, x_1)$ , the theorem may be used to relate the spectrum of  $f$  with those of its subfunctions formed by fixing any subset of the  $n$  variables. This theorem is the backbone for much of the analysis to follow, and we shall make frequent use of it.

### 2.6.3 Spectra of redundant functions

A function  $f(x_n, \dots, x_1)$  is said to be redundant in a variable  $x_i$  if  $f$  is logically invariant under the complementation of  $x_i$ , i.e.

$$f(x_n, \dots, x_i, \dots, x_1) = f(x_n, \dots, \bar{x}_i, \dots, x_1)$$

Consider the spectrum  $R$  of  $f$ . If we partition  $R$  into 2 subvectors so that

$$R = [(R^0)^t \ (R^1)^t]^t,$$

then it is easy to see that  $R^0$  contains only coefficients not involving  $x_n$ , and  $R^1$  contains only coefficients involving  $x_n$ . The following theorems are easy to prove :

Theorem 2.3 A function  $f(x_n, \dots, x_1)$  is redundant in an input variable  $x_i$  if and only if all coefficients involving  $x_i$  are zero.

Proof Without loss of generality, assume that  $x_n$  is redundant. Then the subfunctions  $f_0$  and  $f_1$ , formed by fixing  $x_n$  at 0 and 1 respectively, are equal. Thus  $R_0 = R_1$ . By (2.19), we have

$$[R^0 \ R^1] = [R_0 \ R_1] T^1$$

for which

$$R^1 = R_0 - R_1 = \underline{0}, \text{ the zero vector.}$$

But the coefficients in  $R^1$  are all those involving  $x_n$ , hence all spectral coefficients involving  $x_n$  are zero. Conversely, suppose all coefficients involving  $x_n$  are zero, i.e.

$$R^1 = R_0 - R_1 = \underline{0}$$

which implies  $R_0 = R_1$ , hence  $\underline{f}_0 = \underline{f}_1$ , and  $f$  is redundant in  $x_n$ .  $\Omega$

Theorem 2.4 If  $f(x_n, \dots, x_1)$  is redundant in  $k$  of its variables, then all the coefficients in its spectrum are divisible by  $2^k$ .

Proof Without loss of generality assume  $x_n, \dots, x_{m+1}$  are redundant. Then following the notations in subsection 2.6.2, we have

$$f_0 = f_1 = \dots = f_\beta$$

so that

$$R_0 = R_1 = \dots = R_\beta.$$

From theorem 2.2, we have

$$[R^0 \ R^1 \ \dots \ R^\beta] = [R_0 \ R_1 \ \dots \ R_\beta] T^k$$

hence

$$R^0 = 2^k R_0$$

$$R^i = \underline{0}, \text{ for all } i > 0 \text{ since } t_{.i}^k \text{ (} i > 0 \text{) has equal number of +1's and -1's.}$$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

A redundant function is one which is redundant in one or more of its variables. It can be redefined by an irredundant subfunction with all the redundant variables removed. Given

the CHS and the spectrum of any Boolean function, it is obviously easier to detect any redundancy by looking at the spectrum than at the CHS.

## 2.7 CONCLUSION

In this chapter, we have described in detail the background material on spectral techniques. We shall conclude the chapter with a summary of the advantages of spectral methods and a discussion on their applications.

As we have mentioned, a spectral coefficient has a higher information content about the function than an entry in the CHS. Furthermore, certain operations, which are cumbersome using the conventional Boolean domain, are greatly simplified under the spectral transform. Two such example operations are the detections of redundancy (cf.2.6.3) and symmetry[24]. The well-established mathematics behind the transform also makes it easy to formulate or formalize theories and algorithms, and provides the flexibility necessary for their generalizations. Because of these and other advantages, spectral techniques have wide applications in the analysis of switching functions and the synthesis of digital networks realizing them. Of particular importance are the classification of logic functions[9,10,24], digital network design[10,11,23,24], fault detection[38,39,50] and multi-valued logic[25,51].

In the next chapter, we shall cover the background material on fault detection.

## Chapter III

### FAULT DETECTION - BACKGROUND

#### 3.1 INTRODUCTION

Fault detection of a combinational logic network realizing a Boolean function is the task of determining whether the network is operating correctly. A fault of a logic network is a physical defect within the network which can cause the network to malfunction. Faults can roughly be classified into two categories, according to their behaviour. An intermittent fault is one whose behaviour varies with time. It can be present in some interval of time, and absent or change its nature in others. On the other hand, a solid(permanent) fault always exhibits itself in the same manner. Since intermittent faults are very difficult to test, we shall consider the testing of solid faults only.

For some network, the presence of certain solid faults does not change the function realized. It will be shown later in this chapter that such a situation may arise as a result of redundancy, in the sense that certain components or connections may be eliminated from the network without altering the function realized. Such a network is said to be redundant. Redundant networks are very difficult to be test-

ed, and we shall restrict our attention to irredundant networks only.

Clearly, any solid fault that causes an irredundant network to malfunction can be tested by applying all possible input combinations to the network under test(NUT) and checking all outputs against the CHS of the intended function. This approach is unappealing because of the progressive increases of testing time and data as the number of inputs becomes large. A better approach is to assume that only a certain class of faults can occur, and derive a simpler test procedure that can detect the presence of any fault from this fault model. The most common solid fault models considered are the single stuck-at, the multiple stuck-at, and the bridging fault models.

There are two approaches to fault detection under the fault model assumption. In the traditional approach, a set of test input/output vectors are derived. During testing, each input combination from the list is applied to the NUT and the output is verified. The list is selected in such a way that the presence of any fault from the fault class will cause at least one of the outputs to change its value when its corresponding input combination is applied. Under a more recent approach, frequently referred to as response data compression, a list of input combinations are applied to the NUT, and the outputs(responses) are compressed to form at-



tributes. These attributes of the responses rather than the responses themselves are then checked. Usually, the input test list consists of all possible input combinations. Methods such as transition count testing[20], transition testing[15], signature testing[14], attribute testing[52], syndrome testing[45,46] and parity testing[6] are based on the response data compression approach. The last three methods are in fact special cases of testing by verifying Rademacher-Walsh coefficients, or spectral testing, a technique that is the subject of investigation of our present work.

To ensure proper operation of a logic network, we must be able to detect faults when they have occurred. Also, we must be able to pinpoint the locations of faults accurately enough to permit repair. The problem of locating the faults in a faulty network is known as fault location(fault isolation). Fault detection and fault location are two closely related problems, collectively referred to as the fault diagnosis problem.

In this chapter, we shall introduce some basic concepts and terminology on fault diagnosis. We shall first describe briefly the three mentioned fault models. We then discuss in detail the problem of redundancy. Next we describe and compare the traditional and spectral techniques of fault diagnosis. Two stuck-at fault analysis concepts -- fault collapsing[49] and minimal checkpoints[4] -- are introduced. Finally, the problem of fault location is considered.

### 3.2 FAULT MODELLING OF COMBINATIONAL LOGIC NETWORKS

A logic network is combinational if its output can be expressed as a Boolean function of its inputs  $x_n, \dots, x_1$ , such that the output value is dependent only on the present assignments of the inputs  $x_n, \dots, x_1$ . A combinational logic network may be thought of as an assembly of basic electronic components called logic gates. A logic gate is an elementary switching device whose task is to perform logical operations on the gate input signals. We shall assume that our logic networks are composed of multiple (gate) input, single (gate) output AND, OR, NAND, NOR, NOT, XOR, XNOR basic gates only. The definitions and logic symbols of these basic gates are given in most text books on switching theory(eg.[3,31]) and will not be described here.

The inputs, outputs, and interconnections between gates are called the lines of the network. Any line that is not an input line is an internal line.

Solid faults in a logic network can usually be modelled as logical faults on its lines only. The most popular fault model is the single stuck-at fault model, in which a single line in the network is assumed to be stuck at one of its two logical levels -- stuck-at-zero(s-a-0) or stuck-at-one(s-a-1). A more general model assumes multiple stuck-at faults, where several lines in the network can be simultaneously stuck. Clearly, a single (stuck-at) fault is a special mul-

multiple (stuck-at) fault. Although the single fault model is often assumed to be adequate for fault analysis, even the multiple fault model is not guaranteed to cover all causes of solid faults[3:pp.5-6]. Other models such as the bridging fault model[37] have been proposed to supplement the stuck-at models. However, most faults found in practice can be modelled as a stuck-at fault, and the stuck-at fault models remain the most effective fault models. We shall restrict our attention to the stuck-at models.

In a network with  $p$  lines, there are  $2p$  possible single faults and  $3^p - 1$  possible multiple faults (including single faults). Thus multiple fault analysis is more difficult than its single fault counterpart, because of the much larger number of faults involved. For networks that have been operating correctly and going through testing constantly, the single fault assumption is reasonably valid. However, for densely packed LSI or VLSI circuits, or chips that are undergoing initial testing, a multiple fault model is more realistic.

### 3.3 REDUNDANT NETWORKS

A line in a network is said to be redundant if no output function of the network is changed as a result of cutting the line[13]. Cutting an input to a gate has the same effect on its output as assigning a fixed value to that input such that the output is independent of the fixed input. For exam-

ple, cutting an input line of an AND or NAND gate is equivalent to assigning the fixed value of 1 to that line. Similarly, cutting an input to an OR or NOR gate is equivalent to setting it to 0. Since cutting a redundant connection does not affect the output of the network, it follows that the equivalent s-a-0 or s-a-1 fault will not affect any output for all input combinations, and hence is undetectable by verifying the network output values or some attributes of them. Such a network is said to be redundant.

Though redundancy, in general, complicates the fault diagnosis problem, it may be introduced deliberately to the network to serve a useful purpose. For example, Huffman[21] and McCluskey[35] have shown that a network realizing a Boolean function can be made free of certain types of combinational hazards by the introduction of redundant terms. Another example is the use of redundant hardware to improve the reliability of logic networks in the fault-tolerant design[5] of logic systems.

In this presentation, we shall restrict our attention to irredundant networks.

### 3.4 TRADITIONAL AND SPECTRAL TESTING APPROACHES

In this section, we shall sketch the traditional and spectral approaches to fault detection of irredundant networks. We shall not describe the two approaches in detail, as this has been done for the traditional approach in the cited references, and shall be done for the spectral approach in later chapters. The main aim here is to introduce and then compare these approaches. The single and multiple (stuck-at) fault models are assumed, but the discussion can easily be extended to cover the bridging fault model as well.

#### 3.4.1 Traditional approach

Under the traditional approach, a test for a fault is the application to the NUT of an input combination which, at the presence of the fault, will produce an incorrect output. We shall call such an input combination and the corresponding expected output value a test vector for the fault, and the fault is said to be covered by the test vector. In general, a particular input/output combination may be a test vector for several faults, and a fault may be covered by several test vectors.

A set of test vectors is a complete test set for a fault model of the NUT if any fault in the model is covered by at least one test vector in the set. An absolutely minimal complete test set is a complete test set with the least possible number of test vectors in it. A near minimal complete

test set is one which may not be, but is believed to be close to, an absolutely minimal complete test set.

The derivation of a (absolutely) minimal complete test set usually consists of two stages. They are called the test generation stage and the test minimization stage.

During test generation, the set of all test vectors which cover a particular fault in the fault model is found. This is done for every fault in the fault model. Test generation algorithms such as Boolean difference[29,47], path sensitizing[2], Roth's D-algorithm[44] and fault simulation[8] are well known.

During test minimization, a minimal complete test set is selected from the sets of test vectors derived for each of the faults during test generation. The problem of finding such a minimal test set is exactly analogous to that of the prime implicant covering problem[34], and methods of solutions[18,34,41] to the latter are also applicable to the former.

Usually, the results obtained during test generation are recorded in the form of a fault table[13:pp.46ff], which has a row for each test vector and a column for each fault. An entry  $t_{ij}$  is 1 if the  $i$ th test vector covers the  $j$ th fault, and 0 otherwise. The minimal set of test vectors is then obtained by finding the smallest set of rows such that every

column of the fault table has a 1 in at least one of the rows in the set. The Quine-McCluskey prime implicant table technique[34] may directly be applied to the fault table for this purpose.

During testing, the test inputs from the complete test set are applied in turn to the NUT, and the output values verified. If there exists at least one erroneous output, the network is faulty, otherwise it is assumed to be fault free.

#### 3.4.2 Spectral approach

Under the spectral approach, a spectral test for a fault is the application of all possible input combinations to the NUT and the collection at the network output a spectral coefficient value (of the realized function)  $r_\alpha$  which, at the presence of the fault, will be different from that of the intended function. The fault is said to be  $r_\alpha$ -testable( $r_\alpha$ -detectable). Alternatively,  $r_\alpha$  is said to cover the fault. In general, a particular coefficient can cover several faults and a fault can be covered by several coefficients.

A set of coefficients is a spectral signature for a fault model of the NUT if any fault in the model is covered by at least one coefficient in the signature. The signature is said to cover the fault model. An (absolutely) minimal spectral signature is a signature with the shortest possible

length (i.e. least number of coefficients). A near minimal spectral signature is not necessarily, but is believed to be close to, minimal. If a (spectral) signature is a functional attribute (i.e. it is determined from the intended function and is totally independent of the layout of the NUT), it is termed a functional signature.

Similar to the derivation of a minimal complete test set, the derivation of a minimal spectral signature consists of the test generation and test minimization stages. During test generation, a fault table may again be used to record the results. Each row of the table corresponds to a coefficient and each column to a fault. An entry  $t_{ij}$  is assigned a 1 if fault  $j$  is covered by coefficient  $i$ . During test minimization, the fault table minimization problem is solved in exactly the same way as in the traditional approach, thereby yielding a minimal signature. As we shall see in later chapters, the minimization stage is not always needed, since a minimal or near minimal signature can often be obtained during test generation.

During testing, all input combinations are applied to the NUT and the values of the spectral coefficients in the signature are collected by attaching extra hardware[38] to the network output. If these values are different from the expected ones, the NUT is faulty. Otherwise it is assumed to be fault free.



### 3.4.3 Traditional vs spectral

The major advantages of the spectral approach over its traditional counterpart are :

- 1) No network dependent test input combinations need be generated or applied at test time, as required in the traditional approach.
- 2) All possible input combinations are easily generated by an n-bit counter, where n is the number of inputs to the NUT, at reasonable speed. (eg. In the testing of a 20-input network, one pass through the  $2^{20}$  input combinations at 1MHz requires about one second.)
- 3) For the fault models studied, the signature to be verified is often obtained from the function being realized and the topological class of the NUT, a detailed knowledge of the network layout, required in the traditional approach, is usually not necessary.
- 4) The number of values to be verified is usually smaller than that in the traditional approach.
- 5) The human effort in manual testing is greatly reduced from that involved in the traditional approach, thus the testing procedure is less error prone.

The spectral approach does have some drawbacks :

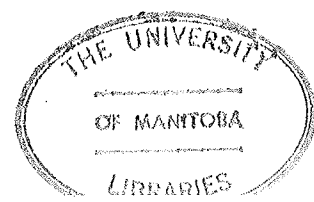
- 1) Testing time can be very long for very large networks.
- 2) Extra hardware is required to collect the spectral coefficients at the network output.

Compared with the advantages gained, the above disadvantages are not of great significance.

In a recent paper by Savir, the concept of constrained syndrome testing[46] is proposed. Savir's concept may be generalized to that of constrained spectral testing, under which a subset of the inputs to the NUT are held fixed while all possible combinations are applied to the remaining inputs. Some spectral coefficients of the subfunction thus realized are checked against those of the expected subfunction. From the preceding description, we can recognize that traditional and spectral approaches are the two extremes in the spectrum of constrained spectral testing -- that is, a traditional test is a constrained test where all inputs are fixed, and a spectral test is a constrained test where none of the inputs are fixed. Although we shall not consider constrained testing in this presentation, it is intuitively clear that a general theory on constrained spectral testing may be developed to relate the traditional and spectral testing approaches together.

### 3.5 FAULT COLLAPSING AND MINIMAL CHECKPOINT

Although the fault table minimization approach as described in subsection 3.4.1 is neat in concept, it is too lengthy for most networks of practical interest. A  $n$ -input network with  $p$  lines would require a fault table of  $2^n$  rows (one for each test vector or spectral coefficient), and  $2p$  (fault)

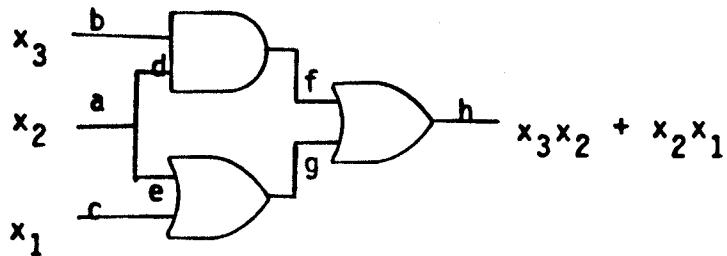


columns under the single fault model, or  $3^P-1$  columns under the multiple fault model. Numerous approaches have been proposed to reduce the size of the fault table. Fault collapsing and minimal checkpoint are two such approaches which attempt to reduce the number of faults in the fault model before test generation. The reduction in faults is achieved by considering Fault equivalence[36]. Two faults are said to be equivalent(indistinguishable) if they individually lead to the same faulty function. Clearly, a traditional or spectral test for a fault will also cover any of its equivalent faults.

The above reduction techniques were originally developed under the traditional approach. They can be carried over to the spectral approach. In this section, we shall describe these two techniques in reasonable detail. More thorough descriptions are given in the original papers[4,49].

### 3.5.1 Fault collapsing

We begin our discussion with some definitions. The fanin of a gate is the number of (gate) inputs it accepts. The fanout of a line is the number of gates it is fed into (each line is assumed to be fed into a gate at most once). A line with a fanout of 2 or more is a fanout line. Otherwise, it is fanout-free.

Example 3.1

In example 3.1, lines b-h are fanout-free lines, while line a is a fanout line with a fanout of 2. The fanout line is often termed the fanout stem (eg. line a), and the lines it fans out to the fanout branches (eg. lines d and e).

Fault collapsing[49] is the process of combining faults by means of implication relationships derived from a study of the NUT. Classes of equivalent faults are found before test generation, and a representative member from each class need then be considered for fault diagnosis. There are three separate stages of fault collapsing. The first and third stages are concerned with fault equivalence, which results in a bidirectional test-fault implication relationship[49] between two indistinguishable faults covered by the same set of tests. The second stage is concerned with unidirectional test-fault implication relationship[49] between two faults tested under the traditional approach. A fault implies a second fault if a test vector covering the first must also

cover the second. Since all input combinations are applied during spectral testing, the second stage is not directly applicable to our analysis. Therefore, we shall only describe the first and third stages of fault collapsing.

First stage fault collapsing corresponds to equivalent single stuck-at faults. For example, a s-a-0 at an input of an AND gate is equivalent to a s-a-0 at the (gate) output.

Third stage fault collapsing relates equivalent single and multiple faults. For example, a s-a-1 at the output of an AND gate is equivalent to all the (gate) inputs simultaneously s-a-1.

Figure 3.1 summarizes the fault equivalence relationships for the fanout stems/branches and the basic gates, assuming that any gate other than a NOT gate has a fanin of two (it can easily be extended for basic gates with fanins greater than two). The results for XOR and XNOR gates were not considered in the original paper.

The application of fault collapsing to a network consists of forming all possible expressions like those of figure 3.1 and combining these expressions by means of transitivity. This results in classes of equivalent faults and a member from each class is selected for fault analysis purposes. If the single fault model is assumed, only first stage fault collapsing is needed. If the multiple fault model is used, both stages should be applied.

## Stage 1

Type	description
AND	any input s-a-0 $\Leftrightarrow$ output s-a-0
OR	any input s-a-1 $\Leftrightarrow$ output s-a-1
NAND	any input s-a-0 $\Leftrightarrow$ output s-a-1
NOR	any input s-a-1 $\Leftrightarrow$ output s-a-0
NOT	input s-a-1 $\Leftrightarrow$ output s-a-0
	input s-a-0 $\Leftrightarrow$ output s-a-1

## Stage 3

Type	description
fanout	stem s-a-0 $\Leftrightarrow$ all branches s-a-0
	stem s-a-1 $\Leftrightarrow$ all branches s-a-1
AND	output s-a-1 $\Leftrightarrow$ both inputs s-a-1
OR	output s-a-0 $\Leftrightarrow$ both inputs s-a-0
	output s-a-1 $\Leftrightarrow$ both inputs s-a-1
NAND	output s-a-0 $\Leftrightarrow$ both inputs s-a-1
	output s-a-1 $\Leftrightarrow$ both inputs s-a-0
NOR	output s-a-1 $\Leftrightarrow$ both inputs s-a-0
	output s-a-0 $\Leftrightarrow$ both inputs s-a-1
XOR	output s-a-0 $\Leftrightarrow$ both inputs s-a-0
	output s-a-1 $\Leftrightarrow$ both inputs s-a-1
XNOR	output s-a-1 $\Leftrightarrow$ one input s-a-0 and one input s-a-1
	output s-a-1 $\Leftrightarrow$ both inputs s-a-0
	output s-a-0 $\Leftrightarrow$ both inputs s-a-1
	output s-a-0 $\Leftrightarrow$ one input s-a-0 and one input s-a-1

Legend :  $\Leftrightarrow$  equivalence relationship

Figure 3.1: Summary of fault equivalence relationships for basic gates

### 3.5.2 Minimal checkpoints

If we base our fault analysis on the multiple fault model, it becomes possible to represent all the multiple faults by specifying a minimal number of lines in the NUT such that any multiple fault in the network becomes equivalent to a multiple fault among these lines only. This minimal set of lines is called the minimal checkpoints of the network. The following theorem is due to Bossen and Hong[4] :

Theorem 3.1[4] The minimal checkpoints in a network are

- 1) all the network inputs that are fanout-free
- 2) all the fanout branches

Any multiple fault in the network is equivalent to a multiple fault involving these lines only.

Proof [4]  $\square$

### 3.5.3 Remark

Minimal checkpoints and fault collapsing techniques may be used together in multiple fault analysis to achieve fault reduction with minimal effort. The checkpoints in the NUT are first identified. Fault collapsing may then be applied to faults among the checkpoints only.

### 3.6 FAULT LOCATION

Fault location is the problem of locating a fault down to a particular module of a faulty network. Usually, a fault location test procedure is carried out in order to pinpoint a faulty module that can be replaced, or to reveal a design flaw at design or early production stage of a chip. Roughly speaking, the quantity of fault location information provided by a fault diagnosis test procedure is called the diagnostic resolution of the procedure. Thus a fault detection test procedure has zero diagnostic resolution, and a fault location test procedure with maximal diagnostic resolution would enable us to locate a fault down to the faulty lines and identify the exact nature of the fault. This is not usually possible because of the presence of equivalent faults.

Because of the large number of faults in the multiple fault model, and consequently the large sizes of equivalent fault classes, each of which may contain faults from several modules of interest, the fault location problem is very complex under the multiple fault assumption. Thus it is more sensible to assume a single fault model, where equivalent faults are usually localized to a small part of the entire network.

A fault location test procedure must be able to distinguish faults from different modules of interest, that is, the results of the procedure for faults from different mod-



ules must be different. The fault table method as described in subsection 3.4.1 can be extended to obtain a minimal fault location test set or a minimal fault location spectral signature of any given diagnostic resolution. This extension is due to Poage[40] and was also described by Kautz[26]. It begins with the construction of a difference table(fault location table). The difference table is constructed from the original fault (detection) table, plus a new column for each pair of faults that have to be distinguished. An entry  $t_{ij}$  is a 1 if test  $i$  produces different results in the individual presence of the pair of faults. We say that test  $i$  distinguishes the fault pair. Any fault table minimization algorithms can then be used to obtain a minimal fault location test set/signature. The information as to what result obtained during testing corresponds to which fault(s) is recorded in form of a table called fault dictionary[8].

As mentioned before, a fault table minimization procedure can be prohibitively lengthy. A difference table is larger, and hence of greater computational complexity. To counteract this problem, several methods have been proposed which simplify the process at the expense of obtaining only a near minimal solution. Two such methods are introduced below.

Galey, Norby and Roth[16] described the method of test set intersection, in which the columns in the difference table are intersected to obtain a reduced table. The outcome

of this method depends very much on the order of intersection and the computation can still be very lengthy.

A second method, called the method of distinguishability criteria, is due to Chang[7] (see also [8]). This method assumes that only a complete or partial fault table (with certain rows missing) is available, and has the ability to select test sets/signatures to specified degrees of diagnostic resolution. The basic idea of the method is to compute weights which reflect the abilities of tests (test vectors or spectral coefficients) to distinguish faults. Tests are then selected systematically on the basis of their weights. After a test is picked, the weights are re-computed for the remaining tests to reflect their ability to differentiate the yet undistinguished fault pairs.

Both of the above methods are not guaranteed to produce minimal solutions. But the later is believed to be better in terms of computational efficiency and closeness to a minimal solution. We shall describe a modified version of Chang's method in more detail in chapter V.

### 3.7 CONCLUSION

In this chapter, we have covered much of the background on fault diagnosis. Fault testing concepts and terminology have been introduced. They are essential for a good understanding of the rest of this presentation, which details the applica-

tion of spectral techniques to the fault diagnosis of irredundant combinational networks under the single or multiple stuck-at fault model.

## Chapter IV

### SYNDROME TESTING OF SINGLE STUCK-AT FAULTS

#### 4.1 INTRODUCTION

The syndrome[45], weight[52], or  $r_0$  of a Boolean function is the number of minterms it realizes (i.e. number of 1's in its CHS). In syndrome testing, all input combinations are applied to the network under test(NUT), and the syndrome of the function realized by the NUT is collected at the network output, usually by means of a counter. If the collected syndrome is different from that of the intended function, the NUT is declared faulty. Otherwise it is assumed to be fault free.

A fault in a network is said to be syndrome-testable( $r_0$ -testable) if its presence can be detected by verifying the syndrome of the network. Syndrome testing is most suitable for single (stuck-at) faults. This is because network lines belonging to a certain topological class are inherently syndrome-testable for both stuck-at-zero(s-a-0) and stuck-at-one(s-a-1) faults. For brevity, we shall term a line that is syndrome-testable for both single faults a syndrome-testable line. Similarly, a syndrome-testable network has all its lines syndrome-testable.

The idea of syndrome testing was first proposed by Tzidon, Berger and Yoeli[52], who also showed that not all networks are syndrome-testable. Savir[45] then considered the design of syndrome-testable networks by modifying syndrome-untestable ones. Markowsky[33] went on to prove that any irredundant network can be modified to become syndrome-testable. Savir later proposed constrained syndrome testing[46] to detect syndrome-untestable single faults. Muzio and Miller[38,39] translated all these results to the Rademacher-Walsh domain, and generalized them to coefficients other than  $r_0$  (i.e. the syndrome).

In this chapter, we shall describe syndrome testing of irredundant combinational networks under the single fault assumption. We first develop the spectral conditions for syndrome-testability. Next we show that all irredundant unate lines are syndrome-testable. Simple tree, NILE, and certain OIL networks are then shown to be syndrome-testable network classes. A syndrome-testable OIL-network implementation of any irredundant function is proposed, and an algorithm for deriving a test procedure for any syndrome-untestable network is given. Finally, the syndrome-testability of a multiple-output network is also investigated.

Most of the material outlined above are attributed to Tzidon et.al.[52], Savir[45], and Muzio and Miller [38,39]. The remaining of this chapter attempts to formalize their

results and present them in a more unified fashion. Some generalizations have also been made to their findings, primarily to networks with XOR/XNOR gates and to syndrome-testable OIL-network design.

#### 4.2 SPECTRAL CONDITIONS FOR SYNDROME-TESTABILITY

In this section, we shall develop the spectral conditions required for a single stuck-at fault in a NUT to be syndrome( $r_0$ )-testable, as was done in [39]. We divide the problem into two situations -- input lines and internal lines.

Consider first a single fault on an input line. Let  $f(x_n, \dots, x_1)$  be the intended function. Define, as in subsection 2.6.2, its subfunctions

$$f_u(x_m, \dots, x_1) = f(u_{n-m}, \dots, u_1, x_m, \dots, x_1)$$

where  $0 \leq u \leq \beta = 2^{n-m} - 1$  and  $u$  is an integer such that

$$u = \sum_{i=1}^{n-m} u_i 2^{i-1}$$

Let  $R$  be the spectrum of  $f$  and  $R_u$  the spectrum of  $f_u$ . Partition  $R$  into  $\beta$  subvectors :

$$R = [(R^0)^t \ (R^1)^t \ \dots \ (R^\beta)^t]^t$$

then by theorem 2.2, we have

$$[R_0 \ R_1 \ \dots \ R_\beta] = \left(\frac{1}{2}\right)^{n-m} [R^0 \ R^1 \ \dots \ R^\beta] T^{n-m} \quad \dots \dots (4.1)$$

or conversely

$$[R^0 \ R^1 \ \dots \ R^\beta] = [R_0 \ R_1 \ \dots \ R_\beta] T^{n-m} \quad \dots \dots (4.2)$$

Putting  $m = n-1$ , we have the simplified formulae :

$$R_0 = \frac{1}{2}(R^0 + R^1) \quad \dots \dots \dots (4.3)$$

$$R_1 = \frac{1}{2}(R^0 - R^1) \quad \dots \dots \dots (4.4)$$

$$R^0 = R_0 + R_1 \quad \dots \dots \dots (4.5)$$

$$R^1 = R_0 - R_1 \quad \dots \dots \dots (4.6)$$

Without loss of generality, assume that the input line  $x_n$  is stuck, so that the network realizes some function  $\overset{*}{f}$  distinct from  $f$ . Let the faulty spectrum be  $\overset{*}{R}$ . Suppose further that  $x_n$  is s-a-0, then  $\overset{*}{f}$  will realize  $f_0$  twice -- once when  $x_n$  is supposed to be zero, and once when  $x_n$  is supposed to be one. Therefore

$$\overset{*}{R}_0 = \overset{*}{R}_1 = R_0$$

and from (4.5) and (4.6), we have

$$\overset{*}{R}^0 = \overset{*}{R}_0 + \overset{*}{R}_1 = 2R_0 = R^0 + R^1 \quad \dots \dots \dots (4.7)$$

$$\overset{*}{R}^1 = \overset{*}{R}_0 - \overset{*}{R}_1 = \underline{0}, \text{ the zero vector} \quad \dots \dots \dots (4.8)$$

By definition,

$$R^0 = [r_0 \ r_1 \ r_2 \ r_{12} \ \dots \ r_{1..(n-1)}]^t$$

$$R^1 = [r_n \ r_{1n} \ r_{2n} \ r_{12n} \ \dots \ r_{1..n}]^t \quad \dots \dots \dots (4.9)$$

hence from (4.7),

$$r_0^* = r_0 + r_n \quad \dots \dots \dots (4.10)$$

and the fault is  $r_0$ -testable if and only if

$$\begin{aligned} r_0 \neq r_0^* &= r_0 + r_n \\ \text{i.e. } r_n &\neq 0 \quad \dots \dots \dots (4.11) \end{aligned}$$

Similarly, for  $x_n$  s-a-1, we have

$$R_0^* = R_1^* = R_1$$

giving

$$R^0 = R_0^* + R_1^* = 2R_1 = R^0 - R^1 \quad \dots \dots \dots (4.12)$$

$$R^1 = \underline{0} \quad \dots \dots \dots (4.13)$$

Again, the fault is  $r_0$ -testable if and only if

$$r_n \neq 0 \quad \dots \dots \dots (4.14)$$

Theorem 4.1[39] An input line is  $r_0$ -testable for both s-a-0 and s-a-1 faults if and only if  $r_i \neq 0$ .

Proof Directly from (4.11) and (4.14).  $\Omega$

Corollary 4.1  $r_0$  will either cover both or neither of the two single faults involving an input  $x_i$ .  $\Omega$



Corollary 4.2 If  $r_0$  is odd, then a single fault on any input line will change  $r_0$  to even. Thus a parity check on  $r_0$  of the function realized by the NUT is a sufficient test for all input single faults.

Proof From (4.7), (4.8), (4.12) and (4.13), any input stuck-at fault will change the parity of the spectrum to even.  $\Omega$

Observe from the preceding analysis that an input stuck-at fault will cause the network to realize a faulty function  $f^*$  that is redundant in the stuck input. Moreover  $f^*$  is obtainable from the intended function  $f$  (since  $f^*$  realizes a subfunction of  $f$  twice). Hence the syndrome-testability condition is independent of the layout of the NUT. For a fault on an internal line, the opposite is true, and its testability condition involves coefficients from a new, network-dependent spectrum. We introduce in figure 4.1 a functional model of the NUT having an internal line  $g$ , where  $f(x_n, \dots, x_1) = h(g(x_n, \dots, x_1), x_n, \dots, x_1)$ .

When operating correctly,  $h$  reduces to  $f$  -- a function of  $n$  variables. However, when  $g$  is stuck,  $h$  behaves as a function of  $n+1$  variables, i.e.

$$h = h(x_{n+1}, x_n, \dots, x_1)$$

with one input  $x_{n+1}$  stuck. The two possibilities are

$$h_0(x_n, \dots, x_1) = h(0, x_n, \dots, x_1) \quad \text{for } g \text{ s-a-0}$$

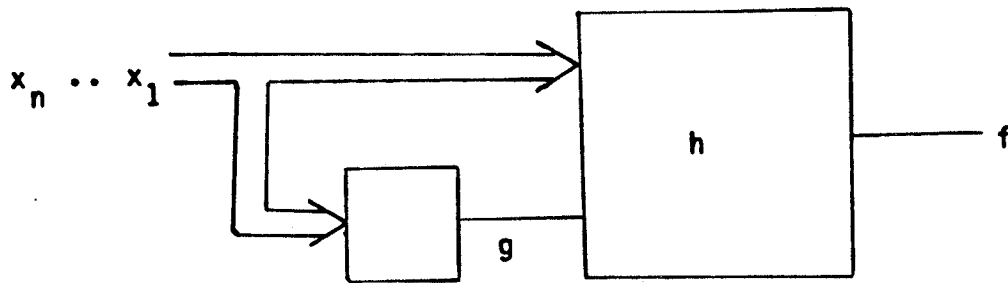


Figure 4.1: functional model for an internal line  $g$

$$h_1(x_n, \dots, x_1) = h(1, x_n, \dots, x_1) \quad \text{for } g \text{ s-a-1}$$

Let  $R$  denotes the sepctrum of  $h$ . From (4.3) and (4.4)

$$R_0 = \frac{1}{2}(R^0 + R^1)$$

$$R_1 = \frac{1}{2}(R^0 - R^1)$$

Again let  $\hat{R}$  be the faulty spectrum when  $g$  is stuck. The two possiblilities are

$$\hat{R} = R_0 = \frac{1}{2}(R^0 + R^1) \quad \text{for } g \text{ s-a-0} \quad \dots (4.15)$$

$$\hat{R} = R_1 = \frac{1}{2}(R^0 - R^1) \quad \text{for } g \text{ s-a-1} \quad \dots (4.16)$$

But the fault is  $r_0$ -testable if and only if  $r_0 \neq \hat{r}_0$ , giving the following theorem :

Theorem 4.2[39] A single fault on an internal line is  $r_0$  testable if and only if

$$r_0 \neq \frac{1}{2}(\bar{r}_0 + \bar{r}_{n+1}) \quad \text{for } g \text{ s-a-0} \quad \dots (4.17)$$

$$r_0 \neq \frac{1}{2}(\bar{r}_0 - \bar{r}_{n+1}) \quad \text{for } g \text{ s-a-1} \quad \dots (4.18)$$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

### 4.3 UNATE LINES

Using theorems 4.1 and 4.2, one can identify all syndrome-testable lines in a network by evaluating for each line the spectral conditions for syndrome-testability. However, there is a certain topological class of irredundant lines, called unate lines, which are necessarily syndrome-testable (for single faults). Therefore it is not always necessary to evaluate the testability conditions for all lines.

A function  $f(x_n, \dots, x_1)$  is said to be positive (negative) in the variable  $x_j$  if there exists a minimum sum-of-product switching expression for it in which  $x_j$  appears only in uncomplemented (complemented) form. It is said to be unate in  $x_j$  if it is either positive or negative in  $x_j$ . That is, we can write

$$f = Ax_j + B \quad \text{or} \quad f = A\bar{x}_j + B$$

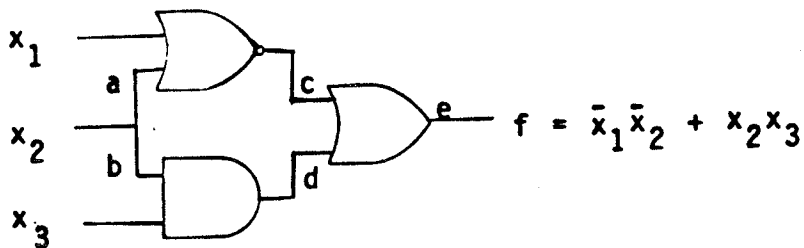
where  $A$  and  $B$  are independent of  $x_j$ . A function that is unate in all its variables is termed a unate function.

The unateness of a network with respect to one of its input lines may be interpreted in the same manner as that of the realized function with respect to one of its input variables. Moreover, the network is said to be unate in an internal line  $g$  if we can express the network output  $f$  as

$$f = Ag + B \quad \text{or} \quad f = A\bar{g} + B$$

where  $A$  and  $B$  are independent of  $g$ . Such input or internal lines are called unate lines.

Example 4.1 For the network



$x_1$ ,  $x_3$  are unate input lines, but not  $x_2$ . The internal line  $a$  is unate since

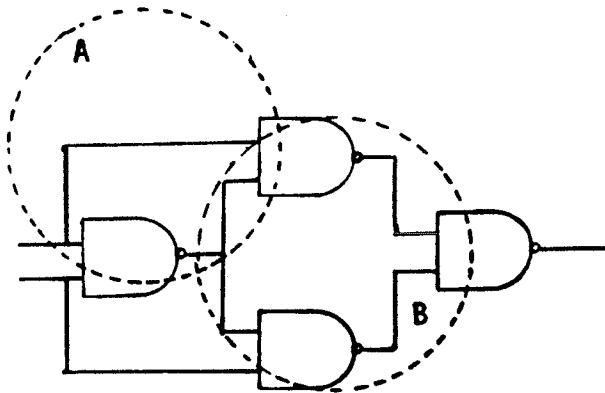
$$f = \bar{x}_1 \bar{a} + x_3 a = A\bar{a} + B$$

where  $A$ ,  $B$  are independent of the line  $a$ . Similarly, lines  $b$ - $e$  are unate.  $\Omega$

Two paths that emanate from a common line (the fanout stem) and reconverge at some forward point (the reconverging

point) are said to be reconverging paths[2], or collectively a loop. The inversion parity[27] along a loop is the parity of the sum of the number on inversions of the two reconverging paths.

Example 4.2 The following is a NAND implementation of the XOR gate :



Loop A has odd parity (number of inversions = 1), while loop B has even parity (number of inversions = 2).  $\Omega$

The next two theorems enable us to identify unate lines in any given network.

Theorem 4.3[45,52] In a network composed of AND, OR, NAND, NOR, NOT, XOR, XNOR basic gates, if there is a unique path from a line  $g$  to the network output, and this path does not contain any XOR or XNOR gates, then  $g$  is unate.

Proof Since only vertex gates are involved in the path originating at  $g$  and terminating at the network output, the network will either be positive or negative in  $g$ , depending on the inversion parity of the path. Hence  $g$  is unate.  $\Omega$

Theorem 4.4[45] In a network composed of AND, OR, NAND, NOR, NOT, XOR, XNOR gates, if all paths from a line  $g$  to the network output contains only loops with even parity, and no path from  $g$  to the network output passes through an XOR/XNOR gate, then  $g$  is unate.

Proof Since any two reconverging paths have equal inversion parity, it follows that any reconverging point is unate in  $g$ , hence the network output is also unate in  $g$ .  $\Omega$

XOR/XNOR gates are excluded from the paths of  $g$  in the above two theorems because, as illustrated by example 4.2, XOR/XNOR gates contain "hidden" odd parity loops which cause non-unateness.

Theorem 4.5[45,52] In a combinational network, any irredundant unate line  $g$  is syndrome-testable.

Proof Assume without loss of generality that the network output  $f(x_n, \dots, x_1)$  is positive in  $g$ , then we can write

$$f = Ag + B$$

where  $A$ ,  $B$ ,  $g$  are functions of  $x_n, \dots, x_1$  and  $A$ ,  $B$  are independent of  $g$ . A s-a-0 on line  $g$  eliminates all the minterms associated with  $Ag$  (there must be at least one such minterm, otherwise  $g$  is redundant), and does not create any new one. Since the syndrome is a minterm count, it is changed. To prove  $g$  s-a-1 is syndrome-testable, we use a contrapositive

approach. Assume that  $g$  s-a-1 is syndrome-untestable. Thus all minterms covered by  $A\bar{g}$  are already covered by  $f$ . Therefore,  $f$  can be represented by

$$f = Ag + A\bar{g} + B = A + B$$

which implies  $g$  is redundant, a contradiction.  $\Omega$

Corollary 4.3[39] In a irredundant combinational network composed of AND, OR, NAND, NOR, NOT, XOR, XNOR basic gates, the candidate syndrome-untestable lines are

- 1) lines which fan out to reconverging paths with unequal inversion parity
- 2) inputs to XOR/XNOR gates
- 3) lines that feed lines of types 1) or 2) directly or indirectly.

Proof By theorems 4.3 and 4.4, all other lines are unate.  $\Omega$

#### 4.4 SYNDROME TESTABLE NETWORKS

If every line in an irredundant network is syndrome-testable for single faults, the network is said to be syndrome-testable. It is obvious that any irredundant network that contains only unate lines is syndrome-testable. Simple tree networks and NILE networks are two classes of such networks. Moreover, an OIL network, which can have non-unate input lines, can also be syndrome-testable if it realizes a Boolean function with some special property. In this section, we shall describe the above network classes and investigate their syndrome-testability.

#### 4.4.1 Tree networks

A tree(fanout-free) network is one which has no fanout lines. Thus each line in the network feeds into at most one gate. A tree network that is composed of AND, OR, NAND, NOR, NOT gates only is termed a simple tree network. Simple tree networks can only realize unate functions. However, not all unate functions are simple-tree realizable (eg. implementing the majority function  $f(x,y,z) = xy + xz + yz$  requires the fanout of at least two input lines).

Theorem 4.6[52] Given any irredundant simple tree network, then

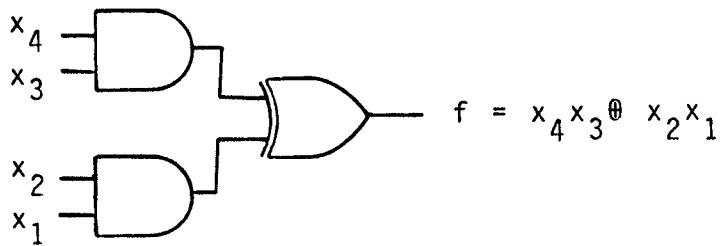
- 1) the syndrome is odd
- 2) the network is syndrome-testable for all multiple stuck-at faults, and a parity check of the syndrome is a sufficient test.

Proof See [52] theorem 3.1. See also theorem 6.2 and corollary 6.5 in chapter VI.  $\Omega$

The syndrome-testability of a tree network that has XOR/XNOR gates is dependent only on the function it realizes. The next example shows one that is syndrome-testable for all multiple faults. We shall defer the proof of testability till chapter VI (see example 6.2), where we consider multiple faults.

Example 4.3 The following tree network





with spectrum R of the the intended function

$$\begin{array}{cccccccc}
 r_0 & r_1 & r_2 & r_{12} & r_3 & r_{13} & r_{23} & r_{123} \\
 6 & -2 & -2 & 2 & -2 & -2 & -2 & -2
 \end{array}$$

$$\begin{array}{cccccccc}
 r_4 & r_{14} & r_{24} & r_{124} & r_{34} & r_{134} & r_{234} & r_{1234} \\
 -2 & -2 & 2 & 2 & 2 & 2 & 2 & 2
 \end{array}$$

is syndrome-testable for all multiple faults. In fact the parity of  $r_0/2$  is a sufficient test.  $\Omega$

#### 4.4.2 NILE networks

A NILE network[52] is a network composed of AND, OR, NAND, NOR, NOT gates only, in which the parity of the Number of Inverters along any Loop is Even. Any unate function can be realized by a NILE network (eg. two-level sum-of-product realization), and simple tree networks are special NILE networks.

Theorem 4.7[52] Any irredundant NILE network is syndrome-testable (for single faults).

Proof Every line in the network is unate. By theorem 4.5, any irredundant unate line is syndrome-testable.  $\Omega$

Corollary 4.4 All primary coefficients  $r_1$  to  $r_n$  of any irredundant unate function are non-zero.

Proof Any unate function is NILE-realizable. All input lines of a NILE network are unate and hence  $r_0$ -testable. From theorem 4.1, it follows that all primary coefficients are non-zero.  $\Omega$

#### 4.4.3 OIL networks

An OIL network[52] is one where Only Input Lines may fanout to a loop with odd inversion parity. This definition implies that XOR and XNOR gates may appear in an OIL network, provided that they are fed only by (network) input lines. Not all functions are unate and NILE-network realizable. However, any function can easily be, and usually is, implemented by an OIL network (eg. 2-level sum-of-product realization, including PLA implementation). Thus it is of practical importance to find a convenient test procedure for them.

Theorem 4.8[38] Any irredundant OIL network realization of a function with non-zero primary coefficients is syndrome-testable.

Proof Clearly, all internal lines are unate and hence syndrome-testable. Since  $r_1$  to  $r_n$  are non-zero, it follows from

theorem 4.1 that all input lines are also syndrome-testable.

Ω

Corollary 4.5[52] Any irredundant OIL network realizing a function with an odd number of minterms is syndrome-testable.

Proof Since  $r_0$  is odd, it follows from theorem 2.1 that all primary coefficients are odd and non-zero. Ω

#### 4.5 SYNDROME-TESTABLE DESIGN

Since half of the functions in the  $n$ -variable Boolean function space (the  $n$ -space) are odd, and there are some even ones that have all primary coefficients non-zero, we can conclude that the majority of functions in the  $n$ -space are realizable by syndrome-testable OIL networks. The exact number of such functions in the  $n$ -space is a combinatorial problem that we have not yet solved. In any event, there are  $n$ -variable functions that have at least one of  $r_1$  to  $r_n$  zero, and cannot be directly implemented by syndrome-testable OIL networks.

One way of overcoming this problem is to realize instead a superfunction of such a "nasty" function, such that the superfunction has one extra variable  $x_{n+1}$  and its primary coefficients  $r_1$  to  $r_{n+1}$  are all non-zero. This superfunction is therefore realizable by a syndrome-testable OIL network.

During normal operation, the extra control input  $x_{n+1}$  is held fixed so that the network realizes the "nasty" function. During testing,  $x_{n+1}$  is also exercised, and the network becomes syndrome-testable.

Savir[45] proposed a tedious procedure for modifying any existing syndrome-untestable network to become syndrome-testable. His procedure may yield unsatisfactory results[38].

For OIL network, a syndrome-testable design may be achieved using the straight-forward approach advocated by Susskind[50] and Carter[6]. Observe that the "nasty" function  $f(x_n, \dots, x_1)$  must be even since one or more primary coefficients are zero. If we consider[50] the superfunction

$$g(x_{n+1}, \dots, x_1) = \bar{x}_{n+1}^* f + x_{n+1}^* x_n^* \dots x_1^* \quad . \quad (4.19)$$

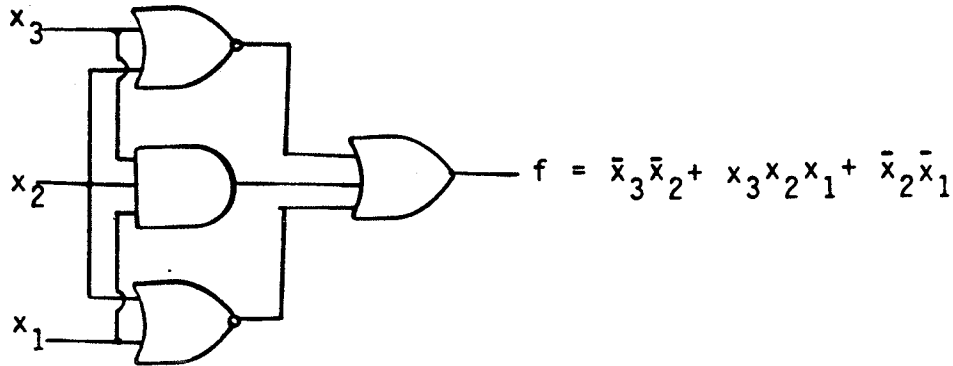
where  $x_i^* \in [x_i, \bar{x}_i]$ , and  $x_{n+1}^* x_n^* \dots x_1^*$  is any arbitrary  $(n+1)$ -variable minterm, then  $g$  realizes one more minterm than  $f$  and hence is odd and has non-zero primary coefficients. A perhaps better odd superfunction is

$$g(x_{n+1}, \dots, x_1) = f + x_{n+1}^* x_n^* \dots x_1^* \quad . \quad . \quad (4.20)$$

where  $x_n^* \dots x_1^*$  is not a minterm of  $f$ , and the syndrome of  $g$  is twice that of  $f$  plus 1. The OIL network design of  $g$  may be obtained by modifying that of  $f$ , adding one extra control input and extra AND/OR gates and connections according to one of (4.19) and (4.20).

The above design approach is not the best possible. Notice that any superfunction of  $f$  with non-zero primary coefficients will do. Even if we insist on an odd superfunction which is more readily obtainable from  $f$ , there are  $2^{2^n-1}$  to choose from ( $2^{2^n-1}$  is the number of odd  $n$ -variable functions, and the  $(n+1)$ -variable superfunction is a combination of two  $n$ -variable functions --  $f$  and any odd function). Although it is not feasible to analyze all these superfunctions in order to find the best one in terms of ease of realization, picking one arbitrarily is obviously not the best approach. If instead we form the odd  $n$ -variable subfunction (for the normally unused value of the control input) by adding or removing an odd number of minterms realized by the nasty function  $f$  in such a way that the resulting sum-of-product form of the (odd) superfunction is as close to  $f$  as possible, we may not need so much extra hardware as in Susskind and Carters' proposed design. The next example should further clarify our discussion.

Example 4.4 The following OIL network



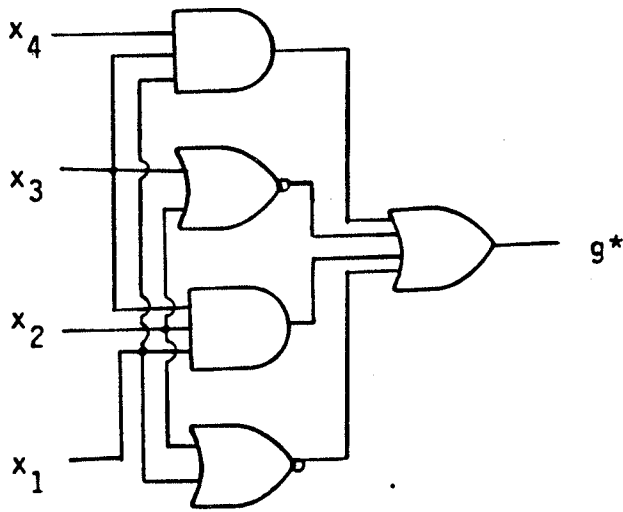
is not syndrome-testable because the spectrum of  $f$  is

$r_0$	$r_1$	$r_2$	$r_{12}$	$r_3$	$r_{13}$	$r_{23}$	$r_{123}$
4	0	2	2	0	0	2	-2

which implies  $x_1$  and  $x_3$  are syndrome untestable. If we make use of (4.20) and add for example the minterm  $x_4\bar{x}_3x_2x_1$ , we end up with the odd superfunction (with  $r_0 = 2 \times 4 + 1 = 9$ ):

$$\begin{aligned} g^* &= f + x_4\bar{x}_3x_2x_1 \\ &= \bar{x}_3\bar{x}_2 + x_3x_2x_1 + \bar{x}_2\bar{x}_1 + x_4\bar{x}_3x_1 \end{aligned}$$

and an OIL network realizing  $g^*$  requires one extra AND, one extra NOT, and 4 extra lines as compared to that realizing  $f$ :



However, if we look at the Karnaugh map of  $f$  and try duplicating it to accommodate an extra input  $x_4$  :

	$x_2 x_1$			
$x_3$	00	01	11	10
0	1	1		
1	1		1	

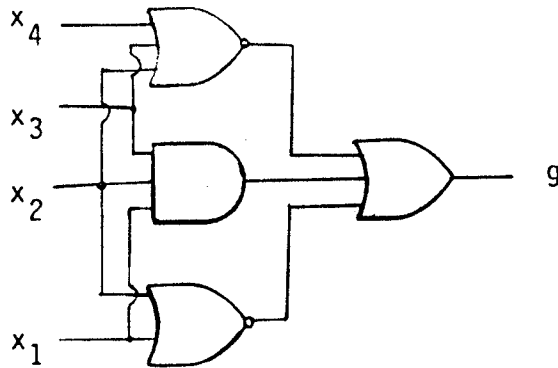
$$f = \bar{x}_3 \bar{x}_2 + x_3 x_2 x_1 + \bar{x}_2 \bar{x}_1$$

	$x_2 x_1$			
$x_4 x_3$	00	01	11	10
00	1	1		
01	1		1	
11	1		1	
10	1	x		

x -- minterm removed

$$g = \bar{x}_4 \bar{x}_3 \bar{x}_2 + x_3 x_2 x_1 + \bar{x}_2 \bar{x}_1$$

Removing the minterm  $x_4 \bar{x}_3 \bar{x}_2 x_1$  after the duplication leaves us with  $g$ , an odd superfunction that can be realized by adding only one extra line ( $x_4$ ) and no extra gating at all :



In both of the modified networks ( $g$  and  $g^*$ ),  $x_4$  is fixed at the logical value zero during normal operation, and exercised during syndrome testing.  $\Omega$

Thus any irredundant function is realizable by a syndrome-testable network, sometimes at the expense of an extra control input and some extra hardware.

#### 4.6 TESTING GENERAL IRREDUNDANT NETWORKS

In general, a network may not be syndrome-testable, and there can be many reasons for not adopting the syndrome-testable OIL network design as described in the previous section :

- 1) The network may have been built and currently under use.



- 2) The price of an extra control input may be too high for the network environment.
- 3) Some functions can be more cheaply implemented using XOR or XNOR gates, which may result in networks with internal syndrome-untestable lines.

There can be several approaches to testing syndrome-untestable networks. We mentioned Savir's constrained syndrome testing[46] as one such approach. Muzio and Miller[39] suggested that coefficients other than  $r_0$  (the syndrome) should be used for the test. We shall leave the discussion of this testing approach to the next chapter. Here, we propose a straight-forward method of testing by supplementing the syndrome with traditional test vectors. We shall describe this method in the form of an algorithm.

Algorithm 4.1 Deriving a test procedure for single faults in an irredundant network by coupling syndrome and traditional testing techniques.

- Step 1. Locate all candidate syndrome-untestable lines using corollary 4.3.
- Step 2. Use stage 1 of fault collapsing (see subsection 3.5.1) to reduce the candidate syndrome-untestable faults into equivalent classes. Any fault that is equivalent to a fault on a syndrome-testable line is discarded.

Step 3. For a representative member of each of the remaining fault classes, evaluate its syndrome-testability using the appropriate spectral condition in theorem 4.1 or 4.2.

Step 4. If no faults are truly syndrome-untestable, the network is syndrome-testable. Otherwise derive a minimal complete test set for all such faults, as described in subsection 3.4.1.

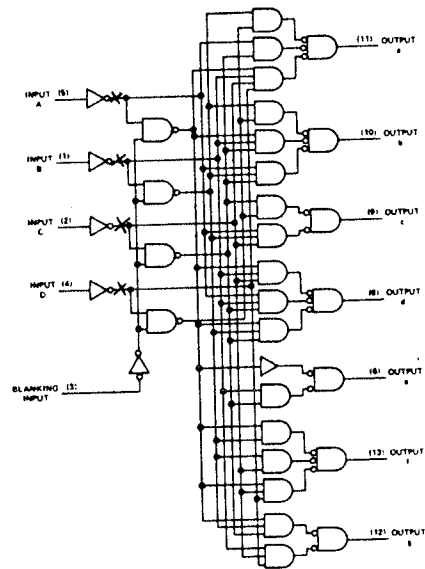
Step 5. During testing, apply syndrome test first. If this fails to reveal any fault, the derived test vectors are applied in turn until a fault is discovered or the test list is exhausted and network assumed to be fault free.  $\Omega$

For networks with a small number of candidate syndrome-untestable faults, the derivation of tests using algorithm 4.1 is much more efficient than deriving a complete test set for all single faults, and requires only a small test set. Thus the algorithm is believed to be very practical for network testing.

#### 4.7 A MULTIPLE-OUTPUT NETWORK EXAMPLE

So far, we have assumed that our NUT has a single output. However, many practical networks consist of multiple outputs. This actually simplifies the derivation of tests, because a line that is not unate with respect to one output may be unate with respect to another. The next example shows an industrial network with four candidate syndrome-untestable lines marked by X's. Since each of these lines has a unique path to at least one of the seven outputs, the entire network can be tested for all single faults by verifying the seven output syndromes.

Example 4.5 A multi-syndrome-testable multiple-output BCD  
seven-segment Decoder/Driver[56]



#### 4.8 CONCLUSION

In this chapter, we have described the syndrome testing technique which tests faults by counting the number of min-terms of the function realized by the NUT. This technique has been applied to single fault testing, with the syndrome-testable condition of a fault expressed in the spectral domain. Unate lines in an irredundant network have been shown to be syndrome-testable, and syndrome-testable tree, NILE and OIL networks have been described. A syndrome-testable OIL-network realization of any irredundant function has been proposed, together with an algorithm for testing syndrome-untestable networks. An example has been used to illustrate syndrome testing of multiple-output networks.

Compared to the traditional testing approach, syndrome testing is better because it simplifies test generation and minimization stages, and the testing procedure is easy to apply. However, syndrome testing requires extra counters[38] and longer testing time. These disadvantages are usually insignificant as compared to the advantage of greatly reducing the number of reference values to be stored and verified.

In the next chapter, we shall describe testing by verifying a set of spectral coefficients of the realized function. This testing approach is a direct generalization of syndrome-testing, which only makes use of  $r_0$ .

## Chapter V

### SPECTRAL TESTING OF SINGLE STUCK-AT FAULTS

#### 5.1 INTRODUCTION

The concept of  $r_\alpha$ -testability is a direct generalization of syndrome-testability by Muzio and Miller[38,39]. A fault in a network is said to be  $r_\alpha$ -testable( $r_\alpha$ -detectable) if its presence will cause the network to realize a function with a  $r_\alpha$  value different from that of the intended function. Thus a syndrome-testable fault is a  $r_0$ -testable fault. The definitions of  $r_\alpha$ -testable line,  $r_\alpha$ -testable network are analogous to those for the syndrome.

Recall from subsection 3.4.2 that in spectral testing, a spectral signature for a set of faults in a network under test(NUT) is a set of spectral coefficients at least one of which covers any fault from the fault set. During testing, all input combinations are applied to the NUT and the value of every coefficient in the signature is collected at the network output and verified. Alternatively, all input combinations are applied several times and the coefficients are verified one at a time. The hardware required for the collection is described in [38].

In this chapter, we shall describe spectral testing of single stuck-at faults in irredundant combinational networks. The spectral conditions for  $r_\alpha$ -testability of single faults are first developed. Algorithms are then described for obtaining a minimal or a near minimal fault detection or fault location signature for a NUT. It is assumed that the cost of testing is identical for each of the coefficients selected. In practice, the cost may vary slightly with the order of the coefficient.

Most material to be presented in this chapter is a formalization of Muzio and Millers' work[38,39]. The spectral solution to the fault location problem has never been described, but some ideas will be borrowed from Poage[40] and Chang[7], whose papers on fault location were based on the traditional test set approach.

## 5.2 SPECTRAL CONDITIONS FOR TESTABILITY

In this section, we shall follow [39] and develop the spectral conditions required for single faults to be  $r_\alpha$ -testable. This has actually been done implicitly in section 4.2 on syndrome-testability. Therefore the description given here will be kept brief. Again, input and internal lines are considered separately.

Consider first a single fault on an input line. Let  $f(x_n, \dots, x_1)$  be the Boolean function that the NUT is supposed

to realize, and  $R$  be its spectrum. Let  $\hat{f}$  be the faulty function and  $\hat{R}$  the faulty spectrum due to  $x_n$  stuck-at-zero(s-a-0). From (4.7) and (4.8) :

$$\hat{R}^0 = \hat{R}_0 + \hat{R}_1 = 2R_0 = R^0 + R^1 \quad \dots \dots \dots (5.1)$$

$$\hat{R}^1 = \hat{R}_0 - \hat{R}_1 = \underline{0}, \text{ the zero vector} \quad \dots \dots \dots (5.2)$$

Let  $n \neq \alpha$  where  $\alpha$  is a coefficient label. Then the coefficients in  $R^0$  and  $R^1$  are of the form  $r_\alpha$  and  $r_{\alpha n}$  respectively. Suppose we want to test for  $x_n$  s-a-0 using  $r_\alpha$ , this requires

$$\begin{aligned} r_\alpha \neq \hat{r}_\alpha &= r_\alpha + r_{\alpha n} \quad \text{from (5.1)} \\ \text{i.e. } r_{\alpha n} &\neq 0 \quad \dots \dots \dots (5.3) \end{aligned}$$

To test for the same fault using  $r_{\alpha n}$ , this requires

$$\begin{aligned} r_{\alpha n} \neq \hat{r}_{\alpha n} &= 0 \\ \text{i.e. } r_{\alpha n} &\neq 0 \quad \dots \dots \dots (5.4) \end{aligned}$$

Similarly for  $x_n$  s-a-1, we derive from (4.12) and (4.13) that the fault is  $r_\alpha$ - and  $r_{\alpha n}$ -testable if and only if  $r_{\alpha n} \neq 0$ .

Theorem 5.1[39] An input line  $x_i$  is  $r_\alpha$ -testable and  $r_{\alpha i}$ -testable for both s-a-0 and s-a-1 if and only if  $r_{\alpha i} \neq 0$ .  $\Omega$

Corollary 5.1 If  $R$  is odd, then a single fault on any input line is covered by any of the coefficients, since the fault will change all their values from odd to even.  $\Omega$



For an internal line  $g$ , we have from (4.15) and (4.16) :

$$\tilde{R} = R_0 = \frac{1}{2}(R^0 + R^1) \quad \text{for } g \text{ s-a-0} \quad \dots (5.5)$$

$$\tilde{R} = R_1 = \frac{1}{2}(R^0 - R^1) \quad \text{for } g \text{ s-a-1} \quad \dots (5.6)$$

from which we deduce the following theorem :

Theorem 5.2[39] A single fault on an internal line  $g$  is  $r_\alpha$ -testable if and only if

$$r_\alpha \neq \frac{1}{2}(\bar{r}_\alpha + \bar{r}_{\alpha(n+1)}) \quad \text{for } g \text{ s-a-0} \quad \dots (5.7)$$

$$r_\alpha \neq \frac{1}{2}(\bar{r}_\alpha - \bar{r}_{\alpha(n+1)}) \quad \text{for } g \text{ s-a-1} \quad \dots (5.8)$$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

Notice that theorems 5.1 and 5.2 and corollary 5.1 are generalizations of theorems 4.1 and 4.2 and corollary 4.2 respectively.

### 5.3 TREE NETWORKS

We have shown that all unate lines are  $r_0$ -testable. However, we are at present unable to determine the existence of such  $r_\alpha$ -testable line classes for  $\alpha \neq 0$ . Thus the general  $r_\alpha$ -testability of a single fault must be explicitly determined from the spectral conditions given in theorem 5.1 or 5.2, or from fault simulation[8] techniques such as explicit enumeration[38] in which the faulty spectrum is obtained by examining explicitly the effect of the fault on the network

output. The fault is  $r_\alpha$ -testable if (the faulty)  $r_\alpha^*$  has a different value from (the expected)  $r_\alpha$ .

The spectral coefficient  $r_0$  of  $f(x_n, \dots, x_1)$  is "symmetric" among the  $n$  variables, in the sense that it has no biased information towards a particular (proper) subset of the variables. Another "symmetric" coefficient in the spectrum is  $r_{1..n}$  (r-all[50], height<sup>1</sup>) which has been shown by Susskind[50] to be an important tool for testing input multiple stuck-at faults. We stated in theorem 4.6 that a simple tree network (without XOR/XNOR gates) can be tested for all multiple faults by checking the parity of the output syndrome. It follows from theorem 2.1 that simple trees are  $r_\alpha$ -testable for any  $\alpha$ . The next theorem covers trees with XOR/XNOR gates. It is included here for the sake of completeness of the present chapter, and we shall leave its proof until chapter VI, where we consider multiple faults.

Theorem 5.3 For an irredundant tree network composed of AND, OR, NAND, NOR, NOT, XOR, XNOR gates

- 1)  $r_{1..n}$  is non-zero,
- 2) the network is  $r_{1..n}$ -testable for all multiple faults.

Proof See theorems 6.3 and 6.4, and corollary 6.6.  $\Omega$

-----  
<sup>1</sup> See section 6.4.

## 5.4 FAULT DETECTION SIGNATURES

a direct application of  $r_\alpha$ -testability is to testing syndrome-untestable networks. A set of coefficients are selected to form a spectral signature which covers every single fault in the NUT. Such a signature must exist since the NUT is irredundant so that every single fault is detectable. The NUT may then be tested by verifying this signature.

In this section, we shall describe algorithms for deriving a minimal spectral signature, and two near-minimal spectral signatures for detecting all single faults in an irredundant network.

### 5.4.1 Minimal Spectral Signature

A Minimal Spectral Signature(MSS) for a set of faults is one which covers every fault in the set and contains the least possible number of coefficients. The derivation of a MSS for all single faults in a NUT is similar to that of a minimal complete test set, as described in subsection 3.4.1. Before test generation, first stage fault collapsing may again be used to reduce the number of faults that need to be considered. The complete algorithm is outlined below :

Algorithm 5.1 A MSS for all single faults in an irredundant network

- Step 1. Use first stage fault collapsing to reduce the faults into equivalent fault classes.

Step 2. Test generation : For each coefficient  $r_\alpha$ , determine the classes of faults that it covers. Only a representative member from each class needs to be considered. If all the fault classes are covered by  $r_\alpha$ , then the NUT is  $r_\alpha$ -testable, and algorithm terminates. Otherwise, add a new row for  $r_\alpha$  in the fault table which has a column for each fault class, and fill in the testability entries.

Step 3. Test minimization : Solve the resulting table minimization problem to obtain a MSS.  $\Omega$

During test generation (step 2 in algorithm 5.1), the testability condition may be determined explicitly using theorem 5.1 or 5.2, or using fault simulation techniques. Choosing the former means that a new network-dependent spectrum must be evaluated for each internal line, and this can be a computationally expensive process. Furthermore, an  $n$ -input  $p$ -line network requires a fault table of  $2^n$  rows and  $2p$  columns, and the test minimization cost increases exponentially with respect to  $n$ . Thus this algorithm is unlikely to be efficient for large size networks.

#### 5.4.2 Near minimal spectral signatures

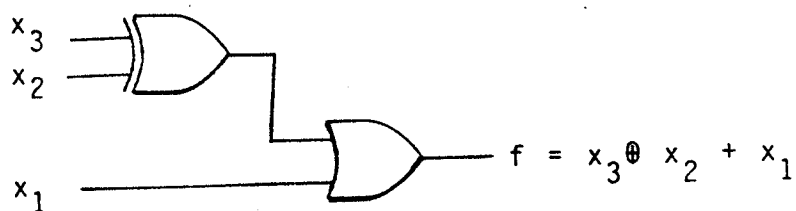
Muzio and Miller[39] suggested that  $r_0$  should always be included in the spectral signature, because a majority of the lines in practical networks are often unate and hence  $r_0$ -testable. We shall formalize their idea with an algorithm to derive a Near minimal Spectral Signature which includes  $r$ -Zero(NSSZ). A NSSZ is  $r_0$  plus a MSS for the set of all  $r_0$ -untestable faults.

Algorithm 5.2 A NSSZ for all single faults in an irredundant network

- Step 1. Identify all candidate  $r_0$ -untestable lines in the NUT using corollary 4.3.
- Step 2. Use first Stage fault collapsing to reduce the faults on these lines into equivalent fault classes. Any fault that is equivalent to a fault on an  $r_0$ -testable line is also  $r_0$ -testable and should not be considered further.
- Step 3. Evaluate the  $r_0$ -testability of a representative member from every fault class. Remove the fault classes that are covered by  $r_0$ .
- Step 4. Carry out steps 2 and 3 of algorithm 5.1 on the  $r_0$ -untestable fault classes to obtain a MSS for the  $r_0$ -untestable faults. Include  $r_0$  in this signature.  $\Omega$

An NSSZ will at worst be one coefficient longer than a corresponding MSS, and usually a considerable amount of computation is saved because the resulting fault table is expected to have fewer columns than the table for the full algorithm for a MSS. The reduction is especially significant for  $r_0$ -untestable OIL networks, where only input lines can be  $r_0$ -untestable. Notice that a NSSZ for an OIL network is a functional signature, i.e. one that is totally independent of the network layout. We shall use the abbreviation FMSSZ for such a functional NSSZ. The abbreviation FNSSZ is not used because the signature is minimal with respect to all functional signatures that includes  $r_0$ .

Example 5.1 For the  $r_0$ -untestable tree network (also an OIL network)



the spectrum of the intended function is

$r_0$	$r_1$	$r_2$	$r_{12}$	$r_3$	$r_{13}$	$r_{23}$	$r_{123}$
6	-2	0	0	0	0	-2	-2

By theorem 5.3, a MSS is  $[r_{123}]$ . Alternatively, we can derive one following algorithm 5.1 :

Step 5.1.1 fault collapsing

class	faults
1	$x_3/0$
2	$x_3/1$
3	$x_2/0$
4	$x_2/1$
5	$x_1/0$
6	$x_1/1, a/1, b/1$
7	$a/0$
8	$b/0$

Step 5.1.2 Built a fault table using some appropriate test generation technique. Notice that the two stuck at faults of an input line may be considered together because a coefficient that covers one will automatically cover the other.

coefficients	fault classes				
	1,2	3,4	5,6	7	8
$r_0$	0	0	1	1	1
$r_1$	0	0	1	1	1
$r_2$	1	0	0	0	0
$r_{12}$	1	0	0	0	0
$r_3$	0	1	0	0	0
$r_{13}$	0	1	0	0	0
$r_{23}$	1	1	1	1	1
$r_{123}$	1	1	1	1	1

Since  $r_{23}$  covers all faults, a MSS is  $[r_{23}]$ .

Now if instead we use algorithm 5.2 to obtain a NSSZ, we perform the followings :

Step 5.2.1  $x_3$  and  $x_2$  are identified as the candidate  $r_0$ -untestable lines.

Step 5.2.2 No fault collapsing is possible.

Step 5.2.3  $x_3/0$ ,  $x_3/1$ ,  $x_2/0$ ,  $x_2/1$  are all found to be  $r_0$ -untestable



Step 5.2.4 Build a fault table :

coefficients	$x_3/0, x_3/1$	$x_2/0, x_2/1$
$r_1$	0	0
$r_2$	1	0
$r_{12}$	1	0
$r_3$	0	1
$r_{13}$	0	1
$r_{23}$	1	1
$r_{123}$	1	1

Since  $r_{23}$  covers all  $r_0$ -untestable lines, a NSSZ (FMSSZ) is  $[r_0, r_{23}]$

The results for this tree network example may be summarized as :

MSS :  $[r_{23}]$  or  $[r_{123}]$

NSSZ :  $[r_0, r_{23}]$  or  $[r_0, r_{23}]$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

Another functional signature for all single faults in an OIL network is a (functional) Minimal Covering Signature which includes r-Zero(FMCSZ)[38]. It is  $r_0$  plus a minimal set of non-zero coefficients such that every  $r_0$ -untestable input is involved in at least one of them, and hence covered by the signature. The algorithm for obtaining a FMCSZ is similar to that of a FMSSZ(NSSZ) and will not be given.

There is no need to explicitly determine the  $r_\alpha$ -testability of each input line -- a coefficient whose label does not span the input is treated as if it does not cover the single faults on it. Only non-zero coefficients need to be considered and the resulting fault table can be considerably smaller. Thus a FMCSZ is more readily obtainable than a corresponding FMSSZ. For a function with small number of inputs ( $< 8$ ), it can usually be identified by inspection of the spectrum.

Example 5.2 Given  $f(x_4, x_3, x_2, x_1) = \bar{x}_4 \bar{x}_2 \bar{x}_1 + \bar{x}_4 x_3 x_1 + x_4 \bar{x}_3 \bar{x}_2$  with non-zero spectral coefficients :

$r_0$	$r_{12}$	$r_{13}$	$r_{23}$	$r_4$	$r_{24}$	$r_{124}$	$r_{34}$	$r_{134}$	$r_{234}$
6	-2	2	2	2	-4	-2	-4	2	-2

Two functional signatures are :

FMSSZ :  $[r_0, r_{34}]$

FMCSZ :  $[r_0, r_{12}, r_{13}]$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

A FMCSZ for an  $n$ -variable function has at most  $n$  coefficients (as in the case when all except the second-order coefficients  $r_{12}$  to  $r_{(n-1)n}$  and  $r_0$  are zero). By definition, a FMSSZ cannot be longer than the corresponding FMCSZ. Thus an upper bound for the length of a MSS for detecting all

single faults in an irredundant OIL network is  $n$ . It is always possible to reduce the length to 1 by realizing instead a "well-behaved" superfunction of  $n+1$  variables using an approach similar to that of section 4.5.

### 5.5 FAULT LOCATION SIGNATURES

Perhaps the most important aspect of spectral signature testing is its application to fault location, a problem which syndrome testing alone is not suitable for. As described in section 3.6, fault location is the problem of locating a fault down to a particular module of a faulty network. A fault location signature is a set of coefficients such that the faulty signature values corresponding to faults that we want to distinguish are different. The quantity of fault location information provided by a fault location signature is called the diagnostic resolution of the signature, and the fault location information provided by the signature may be recorded in the form of a fault dictionary.

A minimal fault location signature with any specified degree of diagnostic resolution may be found by setting up a difference table[26,40] and solving the resulting minimization problem. Such a table would require  $2^n$  rows and a column for each equivalent fault class, plus a column for each pair of fault classes that need to be distinguished. Obviously, the minimization of such a large table is computationally impractical even for small size networks.

It appears that the secondary coefficients  $r_{12}$  to  $r_{1..n}$  are less capable of distinguishing faults than  $r_0$  and the primary coefficients  $r_1$  to  $r_n$ . For example, any input stuck-at fault will force  $r_{1..n}$  to zero. The difference table may be reduced to manageable size if we consider only  $r_0$  to  $r_n$ , plus other appropriate coefficients should  $r_0$  to  $r_n$  fail to distinguish all considered fault pairs. The number of rows in the table are greatly reduced, at the cost of obtaining only a near-minimal location signature.

Another near-minimal signature may be derived by modifying Chang's method of distinguishability criteria[7,8]. His method is equivalent to implicitly setting up a difference table, and counting the number of 1's in each row. This sum, or weight, is the number of fault pairs that the corresponding coefficient can differentiate, and reflects its ability to distinguish faults. At each iteration, the coefficient with the largest weight is selected (the selection is arbitrary in case of a tie). The weights of the other coefficients are then re-computed by subtracting from them the contribution of fault pairs that are already distinguished by the selected coefficient. The step repeats itself until all the remaining weights are zero.

Let  $r_\alpha^i$  denotes the faulty value of  $r_\alpha$  in the presence of fault  $i$ . Let  $r_\alpha^0$  denotes the fault free value. Let

$$(i \neq j)_\alpha = \begin{cases} 0 & \text{if } r_\alpha^i = r_\alpha^j \\ 1 & \text{if } r_\alpha^i \neq r_\alpha^j \end{cases} \dots \dots \dots (5.9)$$

then the initial weight of  $r_\alpha$  is

$$W_\alpha^0 = \sum_{(i,j)^0} (i \neq j)_\alpha \dots \dots \dots (5.10)$$

where  $(i,j)^0$  is the set of all pairs of faults that need to be distinguished, including the fault detection cases  $(0,j)$ . Suppose  $W_{\beta_1}^0$  is the largest among all weights so that  $r_{\beta_1}$  is selected, and  $(i,j)^1$  is the set of yet undistinguished faults, then the new weight of  $r_\alpha$  is

$$\begin{aligned} W_\alpha^1 &= \sum_{(i,j)^1} (i \neq j)_\alpha \\ &= W_\alpha^0 - \sum_{(i,j)^0} (i \neq j)_\alpha (i \neq j)_{\beta_1} \dots \dots \dots (5.11) \end{aligned}$$

and in general, after  $r_{\beta_k}$  is selected,

$$\begin{aligned} W_\alpha^k &= \sum_{(i,j)^k} (i \neq j)_\alpha \\ &= W_\alpha^{k-1} - \sum_{(i,j)^{k-1}} (i \neq j)_\alpha (i \neq j)_{\beta_k} \dots (5.12) \end{aligned}$$

Clearly, this method can easily be programmed into a computer and runs very efficiently.

Instead of further formalizing our algorithm, we shall illustrate it on a hypothesized network in the next example.

Example 5.3 A network has 3 inputs, 4 possible faults and two replaceable modules. Faults  $f_1$  and  $f_2$  are in module 1, while faults  $f_3$  and  $f_4$  belong to module 2. We are to derive a fault location signature which can detect any fault and

locate its presence down to the specific module. We first set up a table with each row corresponds to a coefficient  $r_\alpha$  and each column a fault  $f_i$ . Each entry in the table is  $r_\alpha^i$ . Suppose we choose to use only  $r_0$  and the primary coefficients  $r_1$ ,  $r_2$ , and  $r_3$  :

	$f_0$	module 1		module 2		$W_\alpha^0$	$W_\alpha^1$	$W_\alpha^2$
		$f_1$	$f_2$	$f_3$	$f_4$			
$r_0$	4	2	4	4	4	3	1	0
$r_1$	0	0	0	2	-2	6	-	-
$r_2$	0	-2	-2	-2	-2	4	2	-
$r_3$	-2	0	-2	0	0	3	1	0

The column corresponding to  $f_0$  represents the fault free case. The 8 pairs that need to be distinguished are :

$$(f_0, f_1) , (f_0, f_2) , (f_0, f_3) , (f_0, f_4) \\ (f_1, f_3) , (f_1, f_4) , (f_2, f_3) , (f_2, f_4)$$

Using (5.10) or otherwise, the initial weights  $W_\alpha^0$  are computed. Since  $r_1$  can distinguish most pairs, it is picked. Now the pairs that are not yet distinguished are :

$$(f_0, f_1) , (f_0, f_2)$$

from which we compute the column for  $W_\alpha^1$ .  $r_2$  is the next obvious choice and the iteration terminates since no pair of faults remains undistinguished. Thus a fault location signature is  $[r_1, r_2]$ , and we construct the following fault dictionary :

$r_1$	$r_2$	comment
0	0	fault free
0	-2	fault in module 1
2	0	fault in module 2
-2	0	fault in module 2

which may be referred to during testing.

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

Spectral coefficients are better tools for fault location than the traditional test vectors because they can assume more than just the two values of 0 and 1. Consequently, the spectral approach to fault location requires fewer reference values than its traditional counterpart.

## 5.6 CONCLUSION

In this chapter, we have described the spectral signature testing method which uses one or more spectral coefficients to test for single stuck-at faults in an irredundant network. Algorithms have been described to obtain minimal or near-minimal fault detection and location signatures.

Spectral signature testing of single faults is not as attractive as syndrome testing, because syndrome testing requires less sophisticated testing hardware[38] and syndrome-testability is easier to evaluate. However, it is a good

alternative for testing syndrome-untestable networks, especially for those that are abundant with non-unate lines. Furthermore, it provides a better fault location capability than using the syndrome alone. As we shall see in the next chapter, spectral testing is also suitable for testing multiple faults, and the syndrome is a nasty tool for multiple fault analysis as compared to higher order spectral coefficients.



## Chapter VI

### SPECTRAL TESTING OF MULTIPLE STUCK-AT FAULTS

#### 6.1 INTRODUCTION

A multiple stuck-at fault is a solid fault in which one or more lines of the network under test(NUT) are simultaneously stuck. The total number of stuck lines is called the multiplicity of the fault. Multiple fault analysis is important for two main reasons. Practically speaking, because of the increasing density of logic in VLSI chips as a result of technological advance, the assumption that only single faults can occur is no longer sufficient in practice. Indeed, many simple physical defects such as cracks and power failures manifest themselves as multiple faults. Some statistical studies by Goldstein[19] have also confirmed that in an LSI chip, multiple faults of multiplicity at least up to six must be tested to ensure satisfactory reliability. Theoretically speaking, a single fault is just a special multiple fault (multiplicity = 1), and multiple fault analysis completes some existing single fault concepts.

A major problem concerning multiple fault analysis is the large number of faults that can occur. In an irredundant network with  $p$  lines, there are  $2^p$  single faults and  $3^p - 1$

single and multiple faults possible. Thus it is not feasible to deal with all multiple faults individually, even for small size networks. Under the traditional test vector approach, multiple fault detection has been studied by a number of researchers [1,4,17,27,29,36,40,49]. Numerous methods such as functional equivalence[36], fault collapsing[49] and minimal checkpoints[4] have been proposed to reduce the number of faults to be considered. However, it appears that no effective test generation algorithm has yet been found.

Little work has been done on spectral testing of multiple faults. Tzidon et.al.[52] have shown that simple tree networks without XOR/XNOR gates are syndrome-testable for all multiple faults. Susskind[50] proved that a non-zero r-all (highest order coefficient of the intended function) covers all input multiple faults. Recently, Carter[6] considered the multiple fault coverage of the parity test on the syndrome of a network realizing an odd function. Carter's results are similar to some of those to be presented in this chapter. Since the present results are more general, and were developed using a different approach, his results will not be detailed.

In this chapter, we shall discuss spectral signature testing of multiple stuck-at faults in irredundant networks. We first develop the spectral condition for an input multiple fault to be  $r_\alpha$ -testable. Next we derive a functional

signature(FMSPS) which covers all input multiple faults. We then show that tree networks are easily tested for all multiple faults. For internally fanout-free and general network classes, a FMSPS is shown to cover well over 90% of all possible multiple faults. Using the FMSPS as a basis, algorithms are developed to obtain a near-minimal signature for all multiple faults in a non-tree network.

The results outlined above are the generalizations of those by Tzidon et.al.[52], Susskind[50] and Muzio and Miller[38,39]. Most of the present results have not been published before.

## 6.2 TESTABILITY OF INPUT MULTIPLE FAULTS

The development of the  $r_\alpha$ -testability condition of an input multiple fault is similar to that for an input single fault as described in sections 4.2 and 5.2. From theorem 2.2, we have

$$[R_0 \ R_1 \ \dots \ R_\beta] = \left(\frac{1}{2}\right)^{n-m} [R^0 \ R^1 \ \dots \ R^\beta] T^{n-m} \dots \dots (6.1)$$

or conversely

$$[R^0 \ R^1 \ \dots \ R^\beta] = [R_0 \ R_1 \ \dots \ R_\beta] T^{n-m} \dots \dots (6.2)$$

where  $\beta = 2^{n-m} - 1$ .

Consider an input stuck-at fault with multiplicity  $k=n-m$ . Without loss of generality assume that  $x_n, \dots, x_{m+1}$  are stuck at  $u_k, \dots, u_1$  ( $u_i \in [0,1]$ ). Let  $u$  be the integer such that

$$u = \sum_{i=1}^{n-m} u_i 2^{i-1} \dots \dots \dots (6.3)$$

then  $0 \leq u \leq \beta$ , and for convenience we say  $x_n \dots x_{m+1}$  is stuck-at- $u$  ( $x_n \dots x_{m+1}$ -s-a- $u$ ). Let  $\overset{*}{R}$  be the faulty spectrum, clearly,

$$[\overset{*}{R}_0 \ \overset{*}{R}_1 \ \dots \ \overset{*}{R}_\beta] = [R_u \ R_u \ \dots \ R_u] \dots \dots \dots (6.4)$$

and from (6.2),

$$\begin{aligned} [\overset{*}{R}^0 \ \overset{*}{R}^1 \ \dots \ \overset{*}{R}^\beta] &= [R_u \ R_u \ \dots \ R_u] T^k \\ &= 2^k [R_u \ \underline{0} \ \dots \ \underline{0}] \dots \dots \dots (6.5) \end{aligned}$$

hence

$$\overset{*}{R}^0 = 2^k R_u = [R^0 \ R^1 \ \dots \ R^\beta] t_{.u}^k \dots \dots \dots (6.6)$$

$$\overset{*}{R}^i = \underline{0} \quad \text{for } i > 0 \dots \dots \dots (6.7)$$

where  $t_{.u}^k$  is the  $(u+1)$ -th column of  $T^k$

Define  $\lambda = \lambda_k \dots \lambda_1$  as the labels of the stuck at inputs

- $x_{\lambda_k} \dots x_{\lambda_1}$ ,
- $\lambda$ -s-a- $u$  as the stuck-at fault involving the inputs in  $\lambda$ , where  $x_{\lambda_i}$  is s-a- $u_i$ ,
- $\lambda'$  as a non-empty subset of  $\lambda$

$\alpha$  as a coefficient label such that  $\alpha$  and  $\lambda$  are disjoint

$(r_u)_\alpha$  as the  $r_\alpha$  of  $R_u$

then we can summarize our results by the following :

Theorem 6.1  $\lambda$ -s-a-u is

1)  $r_\alpha$ -testable if and only if

$$r_\alpha \neq r_\alpha^* = [r_\alpha \ r_{\lambda_1\alpha} \ \dots \ r_{\lambda_k\alpha}] t_{.u}^k = 2^k (r_u)_\alpha \quad .(6.8)$$

2)  $r_{\lambda'_\alpha}$ -testable if and only if

$$r_{\lambda'_\alpha} \neq r_{\lambda'_\alpha}^* = 0 \quad . . . . .(6.9)$$

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

Example 6.1 Consider the fault  $x_j x_i$ -s-a- $u_2 u_1$

s-a-pattern		testability condition			
$u_2$	$u_1$	$r_\alpha$	$r_{i\alpha}$	$r_{j\alpha}$	$r_{ij\alpha}$
0	0	$r_\alpha \neq r_\alpha + r_{i\alpha} + r_{j\alpha} + r_{ij\alpha}$	$r_{i\alpha} \neq 0$	$r_{j\alpha} \neq 0$	$r_{ij\alpha} \neq 0$
0	1	$r_\alpha \neq r_\alpha - r_{i\alpha} + r_{j\alpha} - r_{ij\alpha}$	"	"	"
1	0	$r_\alpha \neq r_\alpha + r_{i\alpha} - r_{j\alpha} - r_{ij\alpha}$	"	"	"
1	1	$r_\alpha \neq r_\alpha - r_{i\alpha} - r_{j\alpha} + r_{ij\alpha}$	"	"	"

\_\_\_\_\_  $\Omega$  \_\_\_\_\_

Corollary 6.1 A sufficient but not necessary condition for  $r_\alpha$  to cover an input fault with multiplicity  $k$  is that  $r_\alpha$  is not divisible by  $k$ .  $\Omega$

Corollary 6.2[50] If the highest order coefficient  $r_{1..n}$  is non-zero, then any input multiple fault is covered by  $r_{1..n}$ .

Corollary 6.3 If the spectrum is odd, then any input multiple fault is covered by any one coefficient. In fact, the parity of  $r_0$  (the syndrome) is a sufficient test for all input multiple faults.  $\Omega$

### 6.3 A SIGNATURE FOR INPUT MULTIPLE FAULTS

A Functional Minimal Spanning Signature (FMSPS) is a minimal set of non-zero coefficients such that every input is involved in at least one coefficient. The following corollary of theorem 6.1 is obvious :

Corollary 6.4 A FMSPS is a signature for the set of all input multiple faults.

Proof From (6.9), any input multiple fault will cause at least one of the coefficients to zero.  $\Omega$

Obtaining a (Functional) Minimal Spectral Signature (FMSS) for input multiple faults on a network realizing  $f(x_n, \dots, x_1)$  with  $r_{1..n} = 0$  is computationally impractical, because such a procedure involves a fault table of size  $2^n$  by  $3^n - 1$ . A FMSPS is clearly a near-minimal alternative. Its derivation is very simple :

Algorithm 6.1 Derivation of a FMSPS for an irredundant  $n$ -variable function

Step 1. If  $r_{1..n}$  is non-zero, pick  $r_{1..n}$  and exit.

Step 2. Order all the non-zero coefficients in descending order of the number of variables involved.

Remove any coefficient from the list if the involved variables are already spanned by a higher order coefficient in the list.

Step 3. Construct a table with  $q$  rows and  $n$  columns, where  $q$  is the number of coefficients left in the list. An entry  $t_{ij}$  is 1 if the coefficient corresponding to the  $i$ th row spans the input variable  $x_j$ .

Step 4. Solve the resulting table minimization problem from step 3.  $\Omega$

Before we leave this section, we digress to compare quantitatively in terms of reference values the traditional and spectral approaches to input multiple fault testing. Weiss[55] has shown that for a minimal complete test set for all input multiple faults of a network realizing an irredundant  $n$ -variable function,

- 1) the greatest lower bound on the test length is 3, for all  $n > 1$ ,
- 2) for  $n \leq 5$ , the least upper bound is  $n+1$ ,
- 3) for  $n > 5$ , an upper bound is  $2n-4$ .

In spectral testing, a FMSPS covers the same set of faults. Clearly,

- 1) the greatest lower bound on the length of the signature is 1,

- 2) an upper bound is  $n$ , when all secondary coefficients are zero.

Since the spectral coefficients in a spectrum are not independent, it is possible that for large  $n$ , a smaller upper bound exists. Furthermore, most  $n$ -variable irredundant functions have a FMSPS length of 1. The following combinatorial results support this claim :

Result 6.1 For  $n$ -variable Boolean functions,

- 1) the number of distinct functions is

$$N_n = 2^{2^n} \dots \dots \dots (6.10)$$

- 2) the number of odd functions is

$$O_n = 2^{2^n - 1} \dots \dots \dots (6.11)$$

- 3) the number of functions with  $r_{1..n} = 0$  is

$$Z_n = 2^m C_m, \text{ where } m = 2^{n-1} \dots \dots \dots (6.12)$$

- 4) the number of redundant functions is

$$R_n = \sum_{r=1}^n (-1)^{r-1} n C_r 2^{2^{n-r}} \dots \dots \dots (6.13)$$

Proof 1) and 2) are obvious.

3) Split the  $2^n$  input combinations of a  $n$ -variable function into two halves such that the XOR of the values in any input combination from the first half yields 0, while that from the second half yields 1. From (2.15),  $r_{1..n}$  is the difference of the numbers of 1 function outputs of these two sets of input combinations. Any odd function has  $r_{1..n} \neq 0$ .



Consider an even function with  $r_0 = 2r$ , then  $r_{1..n} = 0$  if and only if  $r$  1's are assigned to the  $m = 2^{n-1}$  outputs of either set. Thus the number of such functions is

$$(\binom{n}{r})^2 = \binom{n}{r} \binom{n}{m-r}$$

Summing this over  $r=0(1)m$  and noting that the sum is the coefficient of  $x^m$  in the expansion of

$$(1+x)^m(1+x)^m = (1+x)^{2m},$$

we have

$$Z_n = \sum_{r=0}^m \binom{n}{r} \binom{n}{m-r} = 2^m \binom{n}{m}$$

4)  $f(r)$ , the number of functions which is redundant in any given set of  $r$  variables, is  $2^{2^{n-r}}$ . This is because the  $2^r$  subfunctions formed by the fixing the  $r$  variables at the  $2^r$  possible combinations must be equal in order for a function to be redundant in all the  $r$  variables. Such subfunction has  $n-r$  variables and hence  $2^{2^{n-r}}$  possibilities, some of which are redundant themselves. By the principle of inclusion and exclusion[43],

$$\begin{aligned} R_n &= \sum_{r=1}^n (-1)^{r-1} \binom{n}{r} f(r) \\ &= \sum_{r=1}^n (-1)^{r-1} \binom{n}{r} 2^{2^{n-r}} \end{aligned}$$

Notice that  $R_n \leq Z_n$ , since any redundant function has  $r_{1..n} = 0$ . In fact,  $R_n \ll Z_n$  as  $n$  becomes large.

The following table tabulates some proportions of  $N_n$ ,  $Z_n$ , and  $R_n$  as functions of  $n$ . It is shown empirically in the last column that for  $n > 5$ , over 90% of the non-redundant functions have non-zero  $r_{1..n}$ , and therefore FMSPS's of length one.

$n$	$Z_n/N_n$	$R_n/N_n$	$(Z_n - R_n)/N_n$
2	0.375	0.375	0.0
3	0.273	0.148	0.125
5	0.140	$7.57 \times 10^{-5}$	0.140
6	0.0993	$1.39 \times 10^{-9}$	0.0993
7	0.0704	$3.79 \times 10^{-19}$	0.0704

#### 6.4 TREE NETWORKS

From Bossen and Hong[4], the minimal checkpoints of a tree network are its input lines. That is, any multiple stuck-at fault is equivalent to some multiple fault among the input lines only. Thus a FMSPS is a signature for all multiple faults. We shall show that any function which is realizable by a tree network composed of AND, OR, NAND, NOR, NOT, XOR, XNOR gates only has a FMSPS of length one.

Define the weight[52]( $r_0$ , syndrome)  $W(f)$  of a Boolean function  $f(x_n, \dots, x_1)$  as the number of minterms it realizes. Define its height( $r_{1..n}$ , r-all[50])  $H(f)$  as

$$\begin{aligned}
 H(f) &= (\text{number of 1 outputs when } x_n \oplus \dots \oplus x_1 = 0) \\
 &\quad - (\text{number of 1 outputs when } x_n \oplus \dots \oplus x_1 = 1) \\
 &\quad \dots \dots (6.14)
 \end{aligned}$$

Let  $f = f(x_n, \dots, x_1) = f(y_p, \dots, y_1, x_q, \dots, x_1)$

$g = g(y_p, \dots, y_1)$

$h = h(x_q, \dots, x_1)$

then  $g$  and  $h$  are two variable disjoint functions. From Tzidon et.al.[52], we have :

Lemma 6.1

$$1) W(\bar{g}) = 2^P - W(g) \dots \dots \dots (6.15)$$

$$2) W(g.h) = W(g)W(h) \dots \dots \dots (6.16)$$

$$3) W(g+h) = 2^q W(g) + 2^p W(h) - W(g)W(h) \dots \dots (6.17)$$

$$\begin{aligned}
 4) W(g \oplus h) &= (2^P - W(g))w(h) + (2^q - W(h))W(g) \\
 &= 2^q W(g) + 2^P W(h) - 2W(g)W(h) \dots \dots (6.18)
 \end{aligned}$$

Proof Omitted. 1)-3) are given in [52], 4) is a simple addition.  $\Omega$

Using a similar approach for heights, we have

Lemma 6.2

$$1) H(\bar{g}) = -H(g) \dots \dots \dots (6.19)$$

$$2) H(g.h) = H(g)H(h) \dots \dots \dots (6.20)$$

$$3) H(g+h) = -H(g)H(h) \dots \dots \dots (6.21)$$

$$4) H(g \oplus h) = -2H(g)H(h) \dots \dots \dots (6.22)$$

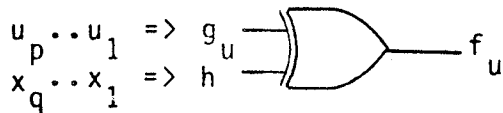
Proof We shall only prove 4), the other proofs are similar. As before,  $f=f(x_n, \dots, x_1)$ ,  $g=g(y_p, \dots, y_1)$ ,  $h=h(x_q, \dots, x_1)$ . By definition,

$$H(f) = (\text{number of 1 outputs when } x_n \oplus \dots \oplus x_1 = 0) \\ - (\text{number of 1 outputs when } x_n \oplus \dots \oplus x_1 = 1)$$

Now suppose  $y_p, \dots, y_1$  are fixed in some pattern

$$u = \sum_{i=1}^p u_i 2^{i-1}$$

and the resulting subfunction is  $f_u$  :



then considering the contribution to  $H(f)$  in such a situation, it is obvious that :

- 1) for each  $g_u = 0$  and  $u_p \oplus \dots \oplus u_1 = 0$ ,  $H(f)$  is increased by  $H(h)$ ,
- 2) for each  $g_u = 1$  and  $u_p \oplus \dots \oplus u_1 = 1$ ,  $H(f)$  is increased by  $H(h)$ ,
- 3) for each  $g_u = 0$  and  $u_p \oplus \dots \oplus u_1 = 1$ ,  $H(f)$  is decreased by  $H(h)$ ,

4) for each  $g_u = 1$  and  $u_p \oplus \dots \oplus u_1 = 0$ ,  $H(f)$  is decreased by  $H(h)$ .

Let the numbers of  $u$ 's satisfying 1)-4) be  $a, b, c$  and  $d$  respectively, then

$$a = 2^{n-1} - c \quad ; \quad b = 2^{n-1} - d$$

and

$$\begin{aligned} H(f) &= (a + d - b - c) H(h) \\ &= (2^{n-1} - c + d - 2^{n-1} + d - c) H(h) \\ &= -2 (c - d) H(h) \\ &= -2 H(g) H(h) \end{aligned}$$

as in (6.22).  $\Omega$

From the first lemma, we state theorems 6.2 and 6.3 :

Theorem 6.2[52] Let  $f(x_n, \dots, x_1)$  be realizable by a simple tree network without XOR/XNOR gates, then  $W(f)$  is odd.

Proof By induction on  $n$ . This is true for all non-trivial functions of one variable. Now we can represent any simple-tree realizable function  $f(x_n, \dots, x_1)$  as a sum-of-product or product-of-sum of two variable-disjoint simple-tree realizable functions  $g(y_p, \dots, y_1)$  and  $h(x_q, \dots, x_1)$ . Assume that any simple-tree realizable function of less than  $n$  variables has odd weight, so that  $W(g)$  and  $W(h)$  are both odd, then if

$$f = g+h$$

$$W(f) = 2^q W(g) + 2^p W(h) - W(g)W(h)$$

and  $W(f)$  is also odd. Similarly, if

$$f = g.h$$

then

$$W(f) = W(g)W(h)$$

and  $W(f)$  is again odd. Result follows by induction.  $\Omega$

Corollary 6.5[52] For any simple tree network, a parity test on the weight of the realized function is sufficient for detecting any multiple fault.  $\Omega$

Theorem 6.3 Let  $f(x_n, \dots, x_1)$  be realizable by a tree network composed of AND, OR, NAND, NOR, NOT and at least one XOR/XNOR gate, then  $W(f)$  is even.

Proof By induction on  $n$ . This is true for a tree network with 2 inputs and one XOR/XNOR gate. Now we can represent any tree-realizable function as a sum-of-product, product-of-sum, or XOR of two variable disjoint tree-realizable functions. From (6.16) and (6.17) of Lemma 6.1, it is obvious that  $W(g.h)$  and  $W(g+h)$  are even if either  $W(g)$  or  $W(h)$  is even. Therefore it is sufficient to show that the XOR of two variable disjoint functions results in a function with even parity, which will propagate from the XOR gate output to the network output. Result follows from (6.18), which shows that  $W(g \oplus h)$  is always even.  $\Omega$

From lemma 6.2, we can prove the next theorem :

Theorem 6.4 Let  $f(x_n, \dots, x_1)$  be any tree-realizable function, then  $H(f)$  is non-zero.

Proof By induction on  $n$ . This is true for all non-trivial functions of one variable. Now  $f$  can be expressed as the sum-of-product, product-of-sum, or XOR of two variable-disjoint tree-realizable functions  $g$  and  $h$ . Assume that any tree-realizable function of less than  $n$  variables has non-zero height, then  $H(g)$  and  $H(h)$  are both non-zero. From lemma 6.2, it follows that  $H(f)$  is also non-zero.  $\Omega$

Corollary 6.6 Any tree network is  $r_{1..n}$ -testable for all multiple faults.

Proof From theorem 6.4,  $r_{1..n} = H(f)$  is non-zero and is a FMSPS for all multiple faults.  $\Omega$

Example 6.2 The 4-input tree network given in example 4.3 is  $r_{1234}$ -testable (for all multiple faults). Furthermore, it is  $r_0$ -testable because

- 1)  $r_0 = 6$  is not divisible by 4. From corollary 6.1, all input faults of multiplicity 2 or more are covered,
- 2)  $r_1 - r_4$  are non-zero. Thus all input single faults are covered.

In fact, any multiple fault will result in the faulty  $r_0$  divisible by 4, and the parity of  $r_0/2$  is a sufficient test. Similarly, the network is  $r_\alpha$ -testable for any other  $r_\alpha$ .  $\Omega$

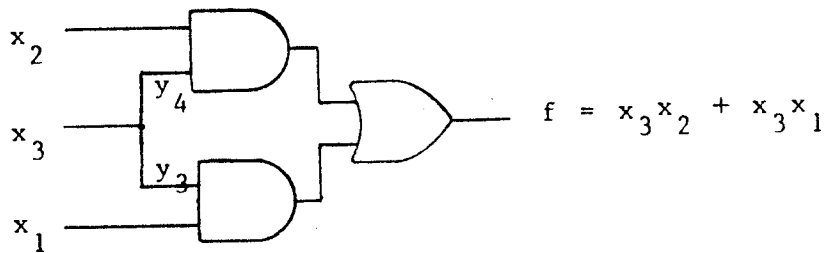
Thus tree networks can easily be tested for multiple faults. An implication of the preceding analysis is its possible application to the design of easily testable fanout-free networks composed of not only the seven basic gates but also complex gates (eg.[11] figure 25). For example, if a complex gate, like AND, OR, XOR gates, realizes a function with non-zero height provided that the variable-disjoint input subfunctions to the gate both have non-zero heights, then constructing a fanout-free network using this and the basic gates will result in a tree network with non-zero height, and any multiple fault will force the height to zero. Further research is required to determine the practicability of such a design.

Without complex gates, not all functions can be realized by tree networks, and we must turn our attention to more complex network classes.

#### 6.5 INTERNALLY FANOUT-FREE NETWORKS

An internally fanout-free network is one where only input lines can fanout. Any function can be realized by an internally fanout-free network (eg. PLA realization). By Bossen and Hong[4], the checkpoints of such a network are the fanout-free inputs and the branches of fanout inputs. For example, in the following network :





any multiple fault is equivalent to a fault involving  $x_1$ ,  $x_2$ ,  $y_3$ ,  $y_4$  only. Thus a signature for all the multiple faults on these lines is also one for the entire network.

In section 6.2, we established the testability conditions for input multiple faults. It is possible to extend the analysis to a multiple fault involving internal lines such as  $y_3$  and  $y_4$  in the previous example network. Such an extension may be based on a functional model similar to the one for internal single faults (figure 4.1). The internal lines involved in the fault would be treated as independent inputs, and a new, larger spectrum derived for the function realized by this hypothesized network. The testability conditions would then be expressed in terms of the intended and the new spectra. This approach has serious drawbacks :

- 1) There are too many multiple faults to be considered individually.
- 2) The size of the new spectrum increases exponentially with the number of internal lines involved.

- 3) The testability conditions resulted will likely be very complex.
- 4) The testability conditions are dependent on the layout of the network.

Therefore it is impractical to adopt the above approach in order to arrive at a signature for all multiple faults in a network. The approach that we shall consider is to begin with a FMSPS and add to it more coefficients until the signature covers all multiple faults. This approach is feasible because a FMSPS covers more than just input multiple faults. The next theorem generalizes the second part of theorem 6.1 :

Theorem 6.5 Any multiple fault involving  $k$  of the inputs and possibly some other input/internal lines will force all coefficients involving the inputs to zero.

Proof The faulty function realized will be independent of the  $k$  inputs, that is, it is redundant in  $k$  of its input variables. From theorem 2.3, all coefficients involving these inputs are zero.  $\Omega$

Corollary 6.7 A sufficient but not necessary condition for  $r_\alpha$  to cover a multiple fault involving any  $k$  inputs and possibly some other lines is that  $r_\alpha$  is not divisible by  $2^k$ .

Proof From theorem 2.4, a function that is redundant in  $k$  of its variables has spectral coefficients that are divisible by  $2^k$ .  $\Omega$

Carter[6] stated that any multiple fault involving one or more input lines of a network realizing an odd function will change the parity to even. The above theorem and corollary are clearly more general. In fact, they can be further generalized :

Theorem 6.6 In a general network, any solid fault which causes the network output to be vacuous in any  $k$  of its input lines will result in a faulty spectrum that has zero values for all coefficients involving the  $k$  inputs, and all coefficients in the faulty spectrum are divisible by  $2^k$ .  $\Omega$

Corollary 6.8 In a general network realizing an irredundant function, a FMSPS will at least cover

- 1) any multiple fault involving any of the inputs,
- 2) any multiple fault involving every of the fanout branches of any fanout input line,
- 3) any other multiple fault that causes the faulty network to become vacuous in any one of its inputs.  $\Omega$

As mentioned earlier, the checkpoints for an internally fanout-free network are the fanout-free inputs and the input fanout branches. Thus 1) and 2) above implies that most multiple faults for such a network are covered by a FMSPS. We shall develop some combinatorial results to illustrate this statement quantitatively. We begin by proving a result that has been quoted several times before :

Result 6.2 The number of possible multiple faults in a network with  $p$  lines is  $3^p - 1$ .

Proof Denote the fault state of the network by a  $p$ -tuple  $(\alpha_1 \dots \alpha_n)$  such that

$$\alpha_i = \begin{cases} 0 & \text{if line } i \text{ is s-a-0} \\ 1 & \text{if line } i \text{ is s-a-1} \\ x & \text{if line } i \text{ is normal} \end{cases}$$

then there are 3 possible assignments to each  $\alpha_i$  and hence  $3^p$  possible assignments to this  $p$ -tuple, representing all possible multiple fault states. Since the state  $(x \dots x)$  represents the fault free state, we have  $3^p - 1$  possible multiple faults.  $\Omega$

Next we count the number of checkpoint multiple faults in an internally fanout-free network which are candidate FMSPS-untestable (i.e. faults that may not be covered by an FMSPS). For simplicity, consider first an internally fanout-free network with  $n$  inputs, each of which has a fanout of 2. Then there are  $2n$  checkpoints along the fanout branches and hence  $3^{2n} - 1$  checkpoints that must be considered. We have the following result :

Result 6.3 An upper bound on the number of checkpoint multiple faults not covered by a FMSPS is  $5^n - 1$ .

Proof Denote any checkpoint fault by an n-tuple  $(\alpha_1 \dots \alpha_n)$  such that

$$\alpha_i = \begin{cases} xx & \text{if both fanout branches of} \\ & \text{of input } i \text{ are normal} \\ x0, 0x, x1, 1x & \text{if one of the two branches} \\ & \text{is stuck} \\ 00, 01, 10, 11 & \text{if both are stuck} \end{cases}$$

Now a FMSPS will cover any fault involving both fanout branches of input  $i$ , thus only the first five of the above nine assignments to  $\alpha_i$  need to be considered. Since the fault free state is  $(xx, \dots, xx)$ , the number of candidate FMSPS-untestable checkpoint faults is  $5^n - 1$ .  $\Omega$

The total number of possible multiple faults in the preceding internally fanout-free network is  $3^{3n+q} - 1$ , where  $q$  is the number of (internal) lines excluding inputs and input fanouts. A conservative estimate of  $q = n$  give rise to the following comparison :

$n$	$5^n - 1$	$5^n / 3^{4n}$
2	24	$3.66 \times 10^{-3}$
5	3124	$9.0 \times 10^{-7}$
7	78124	$3.1 \times 10^{-9}$
10	9765624	$8.0 \times 10^{-13}$

By including an FMSPS in a signature for multiple faults, more than 99% of all the possible faults are exempted from further consideration.

We can now state the combinatorial result for a general internally fanout-free network :

Result 6.4 An upper bound on the number of checkpoint multiple faults not covered by a FMSPS in a n-input internally fanout-free network, with  $x_i$  having  $k_i$  fanout branches is

$$\prod_{i=1}^n (3^{k_i} - 2^{k_i}) - 1 \quad . . . . . (6.23)$$

Proof We again use an n-tuple to represent all the checkpoint faults. Each  $\alpha_i$  has  $k_i$  symbols, and thus  $3^{k_i}$  combinations. If  $\alpha_i$  contains only 0's and 1's, the resulting n-tuple represents a FMSPS-testable fault. It follows that there are only  $3^{k_i} - 2^{k_i}$  permissible assignments to  $\alpha_i$ , and result follows.  $\Omega$

One easily verifies the above result for a tree network, which has no candidate FMSPS-untestable faults.

Although we have taken care of the bulk of all multiple faults by picking a FMSPS, the analysis of the rest is still a nasty problem. This is because, as (6.23) suggests, the number of candidate FMSPS-untestable checkpoint faults increases exponentially with the number of fanout inputs, and it may be computationally impractical to analyze them all. However, the analysis can still be feasible for very large networks with small number of fanout lines, in which case

fault collapsing is again a handy tool for further reducing the number of faults to be considered. In the following, we shall present two algorithms for deriving a complete multiple fault detection signature which contains a FMSPS. The algorithms are in fact applicable to not only internally fanout-free networks but also general irredundant networks.

Algorithm 6.2 Deriving a near-minimal signature for all multiple faults in an irredundant network -- alternative I

- Step 1. Start with an arbitrary FMSPS and note the faults it must cover.
- Step 2. Use first and third stage fault collapsing to divide the candidate FMSPS-untestable checkpoint faults into equivalent classes. Any fault that is equivalent to one of the covered faults can be discarded. Set up a fault table with a row for each coefficient not in the FMSPS.
- Step 3. For a representative member of a fault class, decide whether it is covered by the FMSPS.
- Step 4. If not, determine which other coefficients cover it, and enter the testability information into a column of the fault table.
- Step 5. Repeat steps 3-4 for every fault class.
- Step 6. Solve the resulting fault table minimization problem. A signature for all faults is the FMSPS plus the extra coefficients selected.  $\Omega$

Algorithm 6.3 Alternative II.

- Step 1. Note the fault that must be covered by an arbitrary FMSPS.
- Step 2. Use fault collapsing to divide the candidate FMSPS-untestable faults into equivalent classes.
- Step 3. For a representative member of each class, decide its testability with respect to each of the  $2^n$  coefficients, and add a new column to the fault table which has a row for each coefficient.
- Step 4. Add  $n$  columns to the fault table, one for each input. An entry in the new columns is 1 if the corresponding coefficient spans the corresponding input, otherwise it is 0.
- Step 5. Solve the resulting minimization problem to obtain a signature for all multiple faults.  $\Omega$

Algorithm 6.2 (alternative I) is more efficient since it requires a fault table with fewer columns, while algorithm 6.3 (alternative II) may yield a shorter signature.

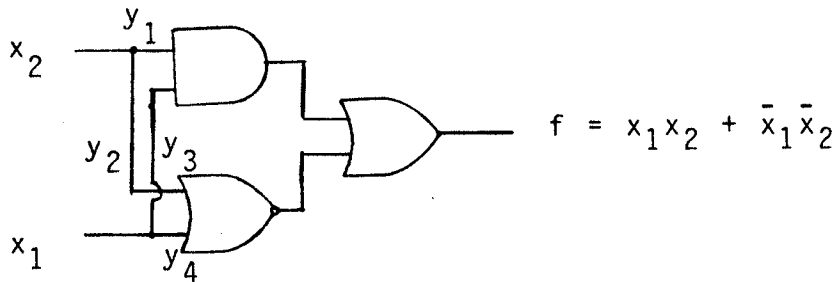
The two algorithms are incomplete without a discussion of a method to derive the  $r_\alpha$ -testability of a multiple fault involving internal lines. It appears that fault simulation is the most suitable method. The fault is injected to a working model of the network and the faulty network outputs



observed. This may be done by modelling the network with a computer program.

We shall illustrate the algorithms with a simple example.

Example 6.3 The following internally fanout-free network



has spectrum

$r_0$	$r_1$	$r_2$	$r_{12}$
2	0	0	2

The only FMSPS is  $[r_{12}]$ . From result 6.3, only  $5^2 - 1 = 24$  (as oppose to  $3^9 - 1 = 19682$ ) checkpoint multiple faults are candidate  $r_{12}$ -untestable. There are four checkpoints ( $y_1$ - $y_4$ ), and any checkpoint multiple fault of multiplicity  $> 2$  causes the network to become vacuous in one of  $x_1$  or  $x_2$  and is covered by  $r_{12}$ . Now we use fault collapsing to further reduce the 24 faults into 8 equivalent classes :

class	faults
1	$y_4/0$
2	$y_4/1, y_2/1, y_4y_2/11, y_4y_2/10, y_4y_2/01$
3	$y_3/0, y_1/0, y_3y_1/00, y_3y_1/01, y_3y_1/10$
4	$y_3/1$
5	$y_2/0$
6	$y_1/1$
7	$y_4y_1/01$
8	$y_3y_2/10$
-	$y_3y_1/11, y_4y_2/00 = y_4y_3y_2y_1/1100 \Rightarrow$ covered
-	$y_4y_1/00 = y_4y_3/00 \Rightarrow$ covered
-	$y_4y_1/10 = y_4y_3/10 \Rightarrow$ covered
-	$y_4y_1/11 = y_2y_1/11 \Rightarrow$ covered
-	$y_3y_2/00, 01, 11 \Rightarrow$ covered

By fault simulation, the following truth table may be obtained for the faulty functions corresponding to the 8 fault classes :

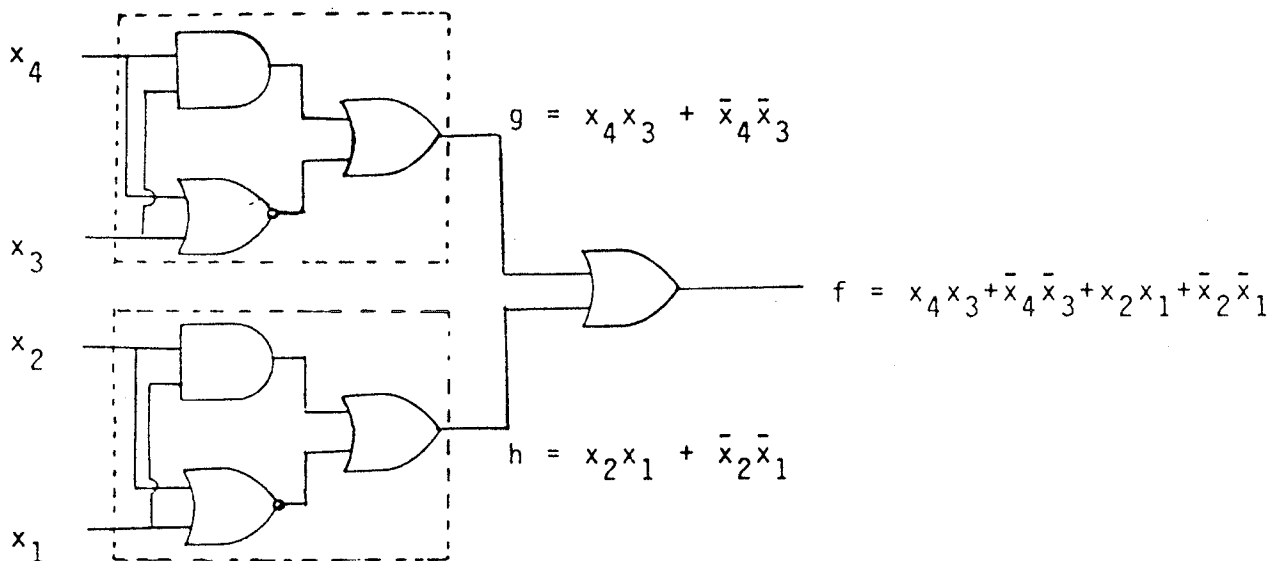
input		fault class							
$x_2$	$x_1$	1	2	3	4	5	6	7	8
0	0	1	0	1	1	1	1	1	1
0	1	0	0	0	1	1	0	0	1
1	0	1	0	0	0	0	1	1	0
1	1	1	1	0	1	1	1	1	1

From the table above, any fault from the 8 fault classes will cause a change in the parity of the spectrum

(even  $\rightarrow$  odd), thus the network is  $r_{12}$ -testable for all multiple faults.  $\Omega$

Recall from section 6.4 that  $r_{12}$  above is also called the height of the realized function. Using the results established for tree networks, it is easy to find a multiple fault signature for the internally fanout-free network in the next example :

Example 6.4 Consider the following network



If we apply algorithm 6.2, we have to investigate  $5^4 - 1 = 624$  multiple faults. By noting that the boxed blocks are identical to the network from the previous example, the problem can be greatly simplified. Since the network output  $f$  can be

represented as the OR of two variable-disjoint functions  $g$  and  $h$ , where

$$H(g) = H(h) = 2 \neq 0$$

it follows from (6.21) that

$$H(f) = H(g+h) = -H(g)H(h) = -4 \neq 0$$

Now a multiple fault confined to the block  $g$  (or block  $h$ ) only is covered by  $H(f)$ , because it changes  $H(g)$  (or  $H(h)$ ) alone and hence changes  $H(f)$ . Furthermore, any multiple fault that has two component faults, one from each block, is also covered because

- 1) Any component fault that causes  $H(g)$  or  $H(h)$  to become zero will, by (6.21), also force  $H(f)$  to zero.
- 2) Otherwise, both faulty  $H(g)$  and  $H(h)$  are odd, and again by (6.21),  $H(f)$  is forced from even to odd.

Therefore the network is  $r_{1234}$ -testable for all multiple faults. Similarly, if the OR gate at the network output is replaced by a NOR, AND, NAND, XOR or XNOR gates, the resulting network is  $r_{1234}$ -testable. Similar analysis may also be applied to more complex networks containing the block from example 6.2.  $\Omega$

The above represents a new concept of multiple fault analysis by partitioning the network under test into blocks such that if these blocks were to be considered as complex

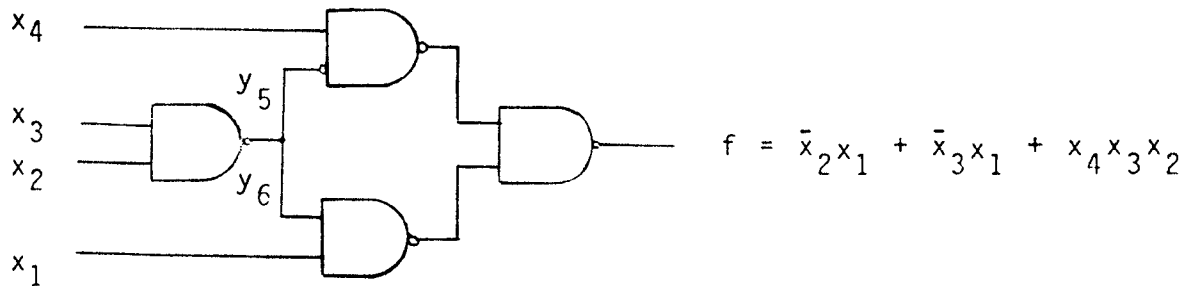
gates, the network becomes fanout-free. We shall call such a partition a block fanout-free partition of the NUT. Multiple fault analysis is first applied to the individual blocks and then combined for a multiple fault involving more than one block. This new concept is in contrast with the one proposed by Schertz and Mertz[49], who partitioned the network into blocks that are themselves fanout-free. Our concept is still under development, and further research is required before it can be applied to the analysis of multiple faults in general irredundant networks.

## 6.6 GENERAL NETWORKS

Algorithms 6.2 and 6.3 are not limited to internally fanout-free networks. For networks with internal fanout, the internal fanout branches are also checkpoints. Thus the candidate FMSPS-untestable checkpoint faults are :

- 1) faults involving the internal fanout branches only,
- 2) faults involving input fanout branches only, such that not all the fanout branches of any input are involved,
- 3) faults which are composed of two components, one from each of 1) and 2).

If a stuck-at fault on an internal stem will cause the network to become vacuous in one of its inputs, faults involving all its branches are also FMSPS-testable. We shall conclude this section with a simple example network that has internal fanout :

Example 6.5 For the network

the checkpoints are  $x_1-x_4$  and  $y_5, y_6$ . Now any fault involving the inputs are covered by an FMSPS. Furthermore, if both  $y_5$  and  $y_6$  are stuck, then the network becomes vacuous in  $x_3$  and  $x_2$ . Thus it is sufficient to consider only 4 faults --  $y_5/0$ ,  $y_5/1$ ,  $y_6/0$ , and  $y_6/1$ . But  $y_5/0$  is equivalent to  $x_1/0$  and  $y_6/1$  is equivalent to  $x_4/0$ . Thus only two faults are potentially FMSPS-untestable. By injecting the faults into the network, we arrive at the following truth table :

$x_4$	input			normal	network output	
	$x_3$	$x_2$	$x_1$		$y_5/1$	$y_6/0$
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	1	1	1
1	0	1	0	0	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1
				$r_0 = \frac{---}{8}$	$\frac{---}{9}$	$\frac{---}{11}$

Since each of the faults will change the output spectrum from even to odd, any coefficient will suffice. The problem of obtaining a multiple fault signature is reduced to that of obtaining a FMSPS of the intended function with spectrum :

$r_0$	$r_1$	$r_2$	$r_{12}$	$r_3$	$r_{13}$	$r_{23}$	$r_{123}$
8	-6	0	-2	0	-2	0	2

$r_4$	$r_{14}$	$r_{24}$	$r_{124}$	$r_{34}$	$r_{134}$	$r_{234}$	$r_{1234}$
-2	0	2	0	2	0	-2	0

FMSPS's such as  $[r_{123}, r_4]$  or  $[r_{12}, r_{34}]$  can easily be identified without setting up a fault table. Since no single coefficient covers all input single stuck-at faults, these signatures are of absolute minimal length.

### 6.7 CONCLUSION

We have presented some practical results on multiple fault detection. We have derived the FMSPS for all input multiple faults and shown that a FMSPS for any tree-realizable function is  $[r_{1..n}]$ . We have also shown that the majority of multiple faults in any irredundant network result in faulty functions that are redundant in one or more of the input variables, and are therefore covered by a FMSPS. Using a FMSPS as a basis, algorithms have been described to obtain a near minimal signature for all multiple faults in any network.

Most functions have a FMSPS length of one. For a nasty function having a long FMSPS, one may implement instead a superfunction of it that has non-zero  $r_{1..n}$ . The selection of such a superfunction is similar to that for the syndrome-testable OIL network design discussed in section 4.5. For a NUT realizing an odd function, a parity check on the syndrome ( $r_0$ ) is sufficient to cover any multiple fault of the types described in corollary 6.8.



In a given network, the probabilities of occurrences of its multiple faults are not the same. Single faults seem to occur most frequently. Thus for a network with abundant fan-out lines, verifying a FMSPS plus any additional coefficients necessary to ensure that all single faults are covered may be sufficient for achieving a practically satisfactory reliability.

## Chapter VII

### SUMMARY AND FUTURE RESEARCH TOPICS

In this chapter, we shall summarize the presented results and outline some future research topics brought forth by the present work.

We have reviewed the traditional approach to stuck-at fault testing of irredundant combinational networks and shown how some of its concepts such as fault table and fault equivalence can be carried over to spectral testing. We have also examined the principal published work on spectral testing, formalizing concepts and theorems wherever the formalization serves to clarify our discussion. Some improvements have been made to the published results and several new results have been established :

- 1) The OIL network class has been expanded to include networks that have input level XOR/XNOR gates (subsection 4.4.3).
- 2) A method has been proposed for implementing any irredundant function by an OIL network that is syndrome-testable for all single stuck-at faults (section 4.5).

- 3) A simple method has been described for testing syndrome-untestable networks (section 4.6).
- 4) The notion of a functional signature has been introduced (subsection 3.4.2). Two functional signatures (FMSSZ and FMCSZ) for all single faults in any OIL realization of the function in question have been described (subsection 5.4).
- 5) Chang's method of distinguishability criteria has been modified to obtain a fault location signature with any specified degree of diagnostic resolution.
- 6) The analysis of single faults has been extended to input multiple faults and a functional signature (FMSPS) has been developed to cover all input multiple faults. For  $n > 5$ , over 90% of  $n$ -variable functions have a non-zero  $r_{1..n}$  and hence a FMSPS length of one (sections 6.2 and 6.3).
- 7)  $r_{1..n}$  has been redefined as the height of the given function, and it has been proven that any function realizable by a tree network composed of AND, OR, NAND, NOR, NOT, XOR and XNOR gates only has a non-zero height and hence a FMSPS of length one. This one-coefficient signature will cover all multiple faults in the tree network (section 6.4).

- 8) The effect of a multiple fault on the network under test has been expressed in terms of the redundancy of the realized function (section 6.5), and the most important and practical result established by the present work is summarized by theorem 6.6 as :

Any solid fault which causes the network output to become vacuous in any  $k$  of its input lines will result in a faulty spectrum that has zero values for all coefficients involving the  $k$  inputs, and all coefficients in the faulty spectrum are divisible by  $2^k$ .

- 9) As a corollary to the above theorem, a FMSPS has been shown to cover the majority of multiple faults in the network under test, and it becomes possible to derive a complete multiple fault detection signature for non-tree networks with a small number of fanout lines (sections 6.5 and 6.6).

As a refinement to this work, the algorithms described for the derivation of signatures should be further optimized to achieve maximum computational efficiency before they are applied to practical networks.

Several combinatorial problems that are of particular interest remain unsolved by our present work :

- 1) The number of  $n$ -variable functions that have at least one zero-valued primary coefficient.
- 2) The number of  $n$ -variable functions that are realizable by a tree network composed of AND, OR, NAND, NOR, NOT, XOR and XNOR gates only.
- 3) The existence of an upper bound that is smaller than  $n$  for the length of a FMSSZ, FMCSZ or FMSPS of any  $n$ -variable function.

Futher research should be initiated covering the following aspects :

1) Spectral testing of bridging faults

Some results have already been developed for bridging faults involving two input lines. It is easy to extend them to cover the bridging of  $k$  input lines ( $k > 2$ ). The analysis of a bridging fault involving internal lines is by no means trivial.

2) Constrained testing

Constrained syndrome testing of single stuck-at faults has already been investigated[46,39]. It is straight-forward to generalize these results to spectral coefficients other than  $r_0$ , and to the testing of input multiple and input bridg-

ing faults. Constrained testing may solve the long test time problem associated with the spectral testing of large networks.

### 3) Syndrome-testable design

The syndrome-testable design discussed in section 4.5 is restricted to OIL network implementations only. Savir's method[45] is applicable to general networks but is known to be ineffective. Efforts should be made to further improve his method.

### 4) Testing tree networks using coefficients other than weights and heights

Analysis similar to that given in section 6.4 should be carried out for coefficients other than  $r_0$  and  $r_{1..n}$ . It will help to gain more insight into the properties of tree-realizable functions and the feasibility of tree implementations using complex gates.

### 5) Multiple fault analysis by block fanout-free partition

The fault analysis approach illustrated by example 6.4 should be developed. At present, it appears that the applicability of this approach is limited to networks with a simple configuration. Thus the possibility of a easily-testable design should also be investigated.

Spectral testing is the most promising approach to testing logic networks. Compared to the traditional approach, spectral testing greatly simplifies the derivation and application of the test procedure. In particular, the number of reference values that have to be handled during test time is significantly reduced. It is hoped that the present work will encourage further research in fault diagnosis using spectral techniques, and new results will be developed to produce more effective procedures for ensuring the correct operation of a logic network.

## BIBLIOGRAPHY

- [1] V.K.Agarwal and A.S.F.Fung, "multiple fault testing of largr circuits by single fault test sets", IEEE Trans. Comput., Vol.C-30, pp.855-865, Nov 1981.
- [2] D.B.Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets", IEEE Trans. Elec. Comput., Vol.EC-15, pp.66-73, Feb 1966.
- [3] R.G.Bennetts, "Introduction to digital board testing", Crane Russak, N.Y., 1982.
- [4] D.C.Bossen and S.J.Hong, "Cause-effect analysis for multiple fault detection in combinational networks", IEEE Trans. Comput., Vol.C-20, pp.1252-1257, Nov 1971.
- [5] M.A.Breuer and A.D.Friedman, "Diagnosis and reliable design of digital systems", Computer Science Press, 1976.
- [6] W.C.Carter, "The ubiquitous parity bit", Proc. FTCS-12, pp.289-296, Jun 1982.
- [7] H.Y.Chang, "An algorithm for selecting an optimal set of diagnostic tests", IEEE trans. Elec. Comput., Vol.EC-14, pp.706-711, May 1965.
- [8] H.Y.Chang, E.Manning and G.Metze, "Fault diagnosis in digital systems", Wiley Interscience, 1970.
- [9] C.K.Chow, "On the characterization of threshold functions", in Switching Theory and Logic Design, pp.34-38, IEEE special pub. no.S-134, Sep 1961.
- [10] C.R.Edwards, "The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis", IEEE Trans. Comput., Vol.C-24, pp.48-62, Jan 1975.
- [11] C.R.Edwards, "The design of easily tested circuits using mapping and spectral techniques", The Radio and Electronic Engineer, Vol.47, pp.321-342, Jul 1977.
- [12] A.D.Friedman, "Fault detection in redundant circuits", IEEE Trans. Elec. Comput., Vol.EC-16, pp.99-100, Feb 1967.



- [13] A.D.Friedman and P.R.Menon, "Fault detection in digital circuits", prentice Hall, N.J., 1971.
- [14] R.A.Frohwerk, "Signature analysis : a new digital field service method", Hewlett-Packard J., pp.2-8, May 1977.
- [15] H.Fujiwara and K.Kinoshita, "Testing logic circuits with compressed data", Proc. FTCS-8, pp.108-113, Jun 1978.
- [16] J.M.Galey, R.E.Norby and J.P.Roth, "Techniques for the diagnosis of switching circuit failures", IEEE Trans. Comm. Elec., Vol.33, pp.509-514, Sep 1964.
- [17] J.W.Gault, J.P.Robinson and S.M.Reddy, "Multiple fault detection in combinational networks", IEEE Trans. Comput., Vol.C-21, pp.31-36, Jan 1972.
- [18] J.F.Gimpel, "A reduction technique for prime implicant tables", IEEE Trans. Comput., EC-14, pp.535-541, 1965.
- [19] L.H.Goldstein, "A probablistic analysis of multiple faults in LSI circuits", IEEE Comp. Soc. Resp., R-77-304, Long Beach, CA.
- [20] J.P.Hayes, "Transition count testing of combinational logic circuits", IEEE Trans. Comput., Vol.C-25, pp.613-620, 1976.
- [21] D.A.Huffman, "The design and use of hazard-free switching networks", J. ACM., Vol.4, Jan 1957.
- [22] S.L.Hurst, "An introduction to orthogonal transforms and Rademacher-Walsh spectra", Notes on Proc. Conf. Recent Developments in Digital Logic Design, School of Elec. Eng., Univ. Bath, Sep 1977.
- [23] S.L.Hurst, "The use of spectral coefficients for logic design", Notes on Proc. Conf. Recent Developments in Digital Logic Design, School of Elec. Eng., Univ. Bath, Sep 1977.
- [24] S.L.Hurst, "The logical processing of digital signals", Crane Russak, N.Y., 1978.
- [25] M.G.Karpovsky, "Finite orthogonal series in the design of digital devices", Wiley, N.Y., 1976.
- [26] W.H.Kautz, "Fault testing and diagnosis in combination-al digital circuits", IEEE Trans. Comput., EC-17, pp.352-366, Apr 1968.

- [27] I.Kohavi and Z.Kohavi, "Detection of multiple faults in combinational logic network", IEEE Trans. Comput., Vol.C-21, pp.556-568, Jun 1972.
- [28] Z.Kohavi, "Switching and finite automata theory", 2nd Ed., McGraw-Hill, 1978.
- [29] C.T.Ku and G.M.Masson, "The Boolean difference and multiple fault analysis", IEEE Trans. Comput., Vol.C-24, pp.64-71, Jan 1975.
- [30] J.G.Kuhl and S.M.Reddy, "On the detection of terminal stuck faults", IEEE Trans. Comput., Vol.C-27, pp.467-469, May 1978.
- [31] S.C.Lee, "Digital circuits and logic design", Prentice Hall, N.J., 1976.
- [32] A.M.Lloyd, "A consideration of orthogonal matrices other than the Rademacher-Walsh type for the synthesis of digital networks", Int. J. on Electronics, Vol.47, pp.205-212, 1979.
- [33] G.Markowsky, "Syndrome testability can be achieved by circuit modification", IEEE Trans. Comput., Vol.C-30, pp.604-606, Aug 1981.
- [34] E.J.McCluskey, "Minimization of Boolean functions", Bell System Tech. J., Vol.35, pp.1417-1444, Nov 1956
- [35] E.J.McCluskey, "Transients in combinational logic circuits", in Redundancy Techniques for Computing Systems", Spartan Books, Washington D.C., 1962.
- [36] E.J.McCluskey and F.W. Clegg, "Fault equivalence in combinational logic networks", IEEE Trans. Comput., Vol.C-20, pp.1286-1293, Nov 1971.
- [37] K.C.Y.Mei, "Bridging and stuck-at faults", IEEE Trans. Comput., Vol.C-23, PP.720-726, Jul 1974.
- [38] J.C.Muzio and D.M.Miller, "Spectral signature for fault testing in combinational circuits", A report submitted to Tektronix Inc., Apr 1981.
- [39] J.C.Muzio and D.M.Miller, "Spectral method for fault detection in combinational circuits", A report submitted to Tektronix Inc., Sep 1981.
- [40] J.F.Poage, "Derivation of optimal tests to detect faults in combinational circuits", in Mathematical Theory of Automata, Polytechnic Press, Brooklyn, N.Y., pp.483-528, 1963.

- [41] I.B.Pyne and E.J.McCluskey, "The reduction of redundancy in solving prime implicant tables", IRE Trans. Elec. Comput., Vol.EC-11, pp.473-482, 1962.
- [42] H.Rademacher, "Einige satze uber reihen von allemeinen orthogonal funktionen", Math. Ann., Vol.87, pp.112-138, 1922.
- [43] J.Riordan, "An introduction to combinatorial analysis", Wiley, N.Y., 1958.
- [44] J.P.Roth, "Diagnosis of automata failure : a calculus and a method", IBM J. Research Development, Vol.10, pp.278-291, Jul 1966.
- [45] J.Savir, "Syndrome-testable design of combinational circuits", IEEE Trans. Comput., Vol.C-29, pp.442-451, June 1980.
- [46] J.Savir, "Syndrome testing of syndrome-untestable combinational circuits", IEEE Trans. Comput., Vol.C-30, pp.606-608, Aug 1981.
- [47] F.F.Sellers, M.Y.Hsiao and L.W.Bearnson, "Analyzing error with the Boolean difference", IEEE Trans. Comput., Vol.C-17, pp.676-683, Jul 1968.
- [48] J.Shanks, "Computation of the fast Walsh-Fourier transform", IEEE Trans. Comput., Vol.EC-18, pp.457-459, May 1969.
- [49] D.R.Shertz and G.Metze, "A new representation for faults in combinational digital circuits", IEEE Trans. Comput., Vol.C-21, pp.858-866, Aug 1972.
- [50] A.Susskind, "Testing by verifying Walsh coefficients", Proc. FTCS-11, pp.206-208, Jun 1981.
- [51] V.H.Tokmen, "Disjoint decomposability of multi-valued functions by spectral means", Proc. 10th International Sym. on Multi-valued Logic, N.W.Univ., Evansion, IL., 1980.
- [52] A.Tzidon, I.Berger and M.Yoeli, "A practical approach to fault detetction in combinational networks", IEEE Trans. Comput., Vol.C-27, pp.968-971, Oct 1978.
- [53] L.J.Ulman, "Computation of the Hadamard transform and the R transform in ordered form", IEEE Trans. Comput., Vol.EC-19, pp.359-360, Apr 1970.
- [54] J.L.Walsh, "A closed set of orthogonal functions", Amer. J. Math., Vol.45, pp.5-24, 1923.

- [55] C.D.Weiss, "Bounds on the length of terminal stuck-fault tests", IEEE Trans. Comput., Vol.C-21, pp.305-309, Mar 1972.
- [56] Texas Instruments Incorporated, "The TTL data book for design engineers", 2nd Ed., 1981.