CLASSIFICATION AND DIGITAL SYNTHESIS USING SPECTRAL TECHNIQUES

by

BALASUBRAMANIAN H. IYER
B.Tech., Indian Institute of Technology, 1979

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the Department
of
Computer Science

We accept this thesis as conforming
to the required standard

[redacted]
Dr. J. Muzio

[redacted]
Dr. G. Shoja

[redacted]
Dr. L. Robertson

[redacted]
Dr. G. Beer

ABSTRACT

This thesis is concerned with the effectiveness of digital circuit design and fault detection, and concentrates on the investigation of several techniques for the classification and synthesis of Boolean functions. A more compact representation of individual Boolean functions based on spectral classification using only $O(n^2)$ coefficients is derived.

Digital design based on spectral classification has the benefit that all functions can be realized using a small set of prototype functions. The resulting modular design is based on the implementation of the prototype function together with the appropriate transformations required to map the given function onto the prototype function. A new design philosophy utilizing symmetry techniques to implement the prototype function supplemented by appropriate spectral translations is developed. Its limitations for both synthesis and fault detection are investigated.

[redacted]
Jon C. Muzio, Supervisor
Department of Computer Science

[redacted]
G.C. Shoja
Department of Computer Science

[redacted]
L.P. Robertson
Department of Physics

[redacted]
G.A. Beer, External Examiner

[redacted]
A. McAuley
Chairman, Examining Committee

[redacted]
J.C. Muzio, Chairman
Department of Computer Science

CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jon Muzio, for his support and guidance. I am also grateful to the members of my committee, Dr. Ali Shoja and Dr. Lyle Robertson. I would like to thank The University Of Victoria for providing me with the facilities to continue my research.

Chapter I

INTRODUCTION

This work has been motivated by a desire to improve the effectiveness of digital circuit design and fault detection, and has concentrated on the investigation of several techniques for the classification and synthesis of Boolean functions. Special emphasis has been given to spectral representations of functions, which, in contrast to more traditional Boolean representations, allow information regarding certain global properties of a function to be compressed into single integer values at the outset of an analysis, thus avoiding the use of numerous (untransformed) function values in later calculations.

More specifically, for an n-variable function, the multiplication of a 2^n X 2^n -element orthogonal transform matrix by a 2^n -element function specification vector results in a column vector containing 2^n 'spectral coefficients', a process requiring only $n2^n$ additions and subtractions (using fast transform principles). As long as the transform matrix is invertible, no information is lost in moving to the spectral domain, and a spectrum may be transformed back into a function specification.

Historically, the transform matrices used in spectral techniques were pioneered by Rademacher and Walsh[1,2]. The spectral domain was first used in signal processing and data transmission problems, then possibilities were seen for its application in digital logic design. The work of Dertouzos and Lechner[3,4] led to the use of spectral techniques for function classification, and their use in synthesis algorithms and realization techniques.

Initial efforts in the present work were directed towards classification of functions, extending the work of others in this field by the use of spectral techniques. A brief survey of the different classification methods is presented in chapter 2. Some functions have networks which are identical except for minor differences, such as the permutation or the negation of inputs. The classification of switching functions into equivalence classes by equivalence under simple operations such as permutation or negation of input variables was first attempted[5-9] in order to better understand the functions that lead to essentially identical networks. These algebraic classification methods have not proved practically useful, because of the rapid increase in the number of classes for numbers of input variables greater than 4. One particular type of function, namely threshold functions (linearly-separable), has been widely researched. At first this may seem strange, since threshold-logic gates are not currently in common use. However, the classification of

linearly separable functions[3,10-16] and the listing of the resultant minimum integer threshold-logic solutions has given considerable insight into the difficulties to be encountered in any classification scheme. The subsequent evolution of the Hadamard transform and the resulting spectral coefficients, applicable to all functions, has led to a more compact classification approach (spectral classification).

The pioneering work of Chow[16] in identifying functions from threshold classes using a reduced set of n+1 coefficients from the the full set of 2^n coefficients motivated our present work, leading to the interesting results presented in chapter 3. A brief summary of Chow's work and the particular n+1 coefficients used for identification can be found in chapter 2. In view of the fact that n+1 coefficients are insufficient for the identification of individual functions from non-threshold classes, we have developed:

1. an empirical relation between these n+1 coefficients and the relative magnitude of the number of functions possessing the same n+1 coefficients;
2. an algorithm to identify the different functions that possess the same n+1 coefficients.

This led us to the discovery that the number of coefficients needed to classify all the 2^{2^n} boolean functions encountered is $O(n^2)$, rather than $O(2^n)$. Furthermore, we have shown that functions from a particular type of class, namely unate classes can be identified using only 2n + 1 coefficients. A

more detailed discussion of these results is presented in chapter 3.

In the field of circuit design, the classification of combinatorial logic functions has the practical objective of allowing us to specify a small set of **standard** or **prototype functions, from which all other functions can be** realized by appropriate interconnections, together with inverter and/or XOR gates in some cases. These prototype functions, one for each equivalence class, can be realized either by using the conventional approach of prime implicant factor extraction or by using certain other kinds of symmetries (e.g. two variable symmetries) exhibited by functions.

A review is given of the types[24] and detection[25] of symmetries in Boolean functions and their use in digital synthesis. A function which exhibits some symmetry is said to be decomposable, and symmetry synthesis is limited to these decomposable functions. Therefore, spectral classification methods and related operations that group sets of functions into equivalence classes were studied in order to supplement symmetry synthesis in cases where no symmetry exists. A new design philosophy utilizing symmetry techniques supplemented by spectral techniques can lead to a modular design. The resulting realization is briefly discussed in chapter 4, and involves supplementing the module implementing the prototype function with appropriate filter modules.

Some limitations to such a synthesis technique were discovered which have important implications for its practical utility. Spectral classification techniques tend to group functions devoid of symmetries together into certain equivalence classes, a tendency which is undesirable for symmetry synthesis methods since it means that classes of predominantly non-decomposable functions are formed. In order for our approach to synthesis to be feasible, every equivalence class must contain at least one function with some symmetry. Every 4 variable spectral class possesses such a function and Miller[28] hypothesized that this is true for higher values of n. An investigation pertaining to the feasibility of such a synthesis approach is described in chapter 5.

We developed two algorithms for testing the non-decomposability of spectral classes iand they are presented in chapter 5. One generates all the functions in an equivalence class and tests for their decomposability, while the other generates all the decomposable functions that might belong to the class under investigation and checks for their membership in that class. The former algorithm is efficient for testing the decomposability of a class while the latter is efficient for checking the non-decomposability of a class. We have investigated the feasibility of our new design philosophy by testing exhaustively every 5 variable spectral class for non-decomposability, using the latter algorithm.

Chapter II

CLASSIFICATION OF FUNCTIONS

The number of possible different functions of 'n' independent variables is 2^{2^n}, including degenerate functions[1]. Clearly this becomes a very large number for large 'n', and certainly too large to be individually listed in any algebraic or similar format for any value of 'n' greater than 3. Even worse, of course, is the situation for higher-valued functions, R > 2, where the number of functions now becomes R^{R^n}.

Some means of classification of functions into a set of equivalence classes, however, is useful from several aspects. Firstly there is the increased understanding of functions which have some common property, which may lead to some insight into the realization of all the functions in the same equivalence class. This may then have practical implications in that testing and fault diagnosis procedures may be standardized for a particular class. Secondly, there is the possibility of establishing a small set of 'standard functions' or 'prototype functions', one from each class.

---

[1] Degenerate functions of 'n' input variables are functions which do not depend upon all 'n' inputs to determine the function output, that is, one or more of the input variables are redundant.

To facilitate our discussion we define an **equivalence relation** i.e., R is an equivalence relation on a set S if it satisfies the following laws:

1. Reflexive law: f_i R f_i for all $f_i \in$ S.
2. Symmetric law: If f_i R f_j, then f_j R f_i holds for all f_i, $f_j \in$ S.
3. Transitive law: If f_i R f_j and f_j R f_k, then f_i R f_k holds for all f_i, f_j, $f_k \in$ S.

An arbitrary function f_i and all other functions f_j, f_k,..., such that f_i R f_j, f_i R f_k,..., are said to constitute an **equivalence class.** Hence, any function 'f' that belongs to a particular equivalence class 'C' may be realized by the implementation of some standard or prototype function for that class, together with the appropriate operations that map function 'f' onto the standard or prototype function. These mapping operations correspond to the method of classification adopted. In the subsequent sections, we present a review of the different classification methods and their corresponding operations.

2.1 ALGEBRAIC CLASSIFICATION METHODS

The three main algebraic classifications which have been developed for binary functions are the PN classification, the NPN classification and the SD classification. Each is progressively more compact in classifying the 2^{2^n} possible binary functions.

The principal algebraic classification method involves the consideration of the following operations, taken individually or collectively :

1. Negation (N) of one or more of the input variables of the function.
2. Permutation (P) of two or more of the input variables of the function.
3. Negation (N) of the output of the function.

2.1.1 The PN Classification

The earliest published method for the classification of binary functions was the method used in producing the Harvard switching function tables[5,6]. This method has subsequently been termed the PN classification where functions are equivalent under operations (1) and (2) mentioned above.

For illustration, the full PN classification of functions for n <= 2 contains 6 classes[10] and is as follows:

$f(x) = 0$	(one function in the class)
$f(x) = 1$	(one function in the class)
$f(x) = x_1$	(four functions in the class)
$f(x) = x_1x_2$	(four functions in the class)
$f(x) = x_1 + x_2$	(four functions in the class)
$f(x) = x_1 \oplus x_2$	(two functions in the class)

Total : 16 functions, six classes, for n <= 2.

2.1.2 The NPN Classification

In the NPN classification[7-9], functions are equivalent under the operations (1), (2) and (3) discussed above. This extra operation allows functions which are related by the application of De-Morgan's theorem to be grouped in the same equivalence class, for example the PN entries of x_1x_2 and $x_1 + x_2$ are both in the same NPN class.

The six PN classes for n <= 2, detailed in the previous section, can be compacted into four NPN classes[10] as follows:

$f(x) = 1$	(two functions in the class)
$f(x) = x_1$	(four functions in the class)
$f(x) = x_1 + x_2$	(eight functions in the class)
$f(x) = x_1 \oplus x_2$	(two functions in the class)

Total : 16 functions, four classes, for n <= 2.

2.1.3 The SD Classification

The third and most compact of the algebraic classification techniques is the SD (self-dual) classification of Goto and Takahas[3,8,10]. This classification was particularly relevant for the restricted class of linearly-separable (threshold) functions, but is applicable to all functions whether linearly-separable or not.

The definition of the SD classification first requires the introduction of dual and self-dual functions. The dual of any function f(x) is given by

$$f^d(x) \triangleq \overline{f(\bar{x})}$$

where the bar within the parenthesis indicates that all the x_i variables of the function are individually complemented, whilst the bar over the whole function indicates the usual output function negation. If now $f^d(x) = f(x)$, then $f(x)$ is said to be self-dual. However, if a self-dual function $f^{sd}(X)$ of n+1 variables $x_1,\ldots,x_n,x_{n+1}$ is considered, it may be decomposed using Shannon's decomposition, into two sub-functions $f_0(x)$ and $f_1(x)$ of n variables as follows:

$$f^{sd}(X) = \bar{x}_{n+1}[\ f_0(x_1,\ldots,x_n)\ ] + x_{n+1}[\ f_1(x_1,\ldots,x_n)\ ]$$

The property of a self-dual function implies that these two decomposition functions $f_0(x)$ and $f_1(x)$ are duals of each other, that is

$$f^{sd}(X) \triangleq x_{n+1}f(x) + \bar{x}_{n+1}f^d(x)$$

where $f(x) = f(x_1,\ldots,x_n)$. If we decompose $f^{sd}(x)$ about any x_i, the two resulting subfunctions will be duals of each other.

Now a function $f(x)$ and its dual $f^d(x)$ lie within the same NPN equivalence classification, being related by input and output negation operations, but the self-dual function $f^{sd}(X)$, sometimes termed the '**hyperfunction**' $f^h(X)$ of $f(x)$, lies above this NPN classification, unless $f(x)$ happens to be a self-dual function, in which special case $f^h(x) = f(x)$.

In the SD classification, both $f^{sd}(X)$ of n+1 variables and all possible Shannon decompositions into functions of n variables are classed in the same SD classification entry. Thus $f^{sd}(X)$ becomes a more compact algebraic classification than the preceding NPN-equivalence classification.

The full SD classification[2] of functions for n <= 3 contains 7 classes[10] and these classes are given below:

$f(x) = x_1$ (4 fns. in the class)

$f(x) = x_1x_2 + x_2x_3 + x_1x_3$ (32 fns. in the class)

$f(x) = x_1 \oplus x_2 \oplus x_3$ (8 fns. in the class)

$f(x) = (x_1 + x_2 + x_3)x_4 + x_1x_2x_3$ (128 fns. in the class)

$f(x) = (x_1x_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3)x_4$
$+ (x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)\bar{x}_4$ (64 fns. in the class)

$f(x) = x_1(x_2x_3 + \bar{x}_2\bar{x}_3)x_4$
$+ (x_1 + x_2\bar{x}_3 + \bar{x}_2x_3)\bar{x}_4$ (96 fns. in the class)

$f(x) = (\bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3)x_4$
$+ (x_1x_2 + x_2x_3 + x_1x_3 + \bar{x}_1\bar{x}_2\bar{x}_3)\bar{x}_4$ (128 fns. in the class)

Total : 256 functions, 7 entries, for n <= 3.

For comparison purposes the PN, NPN and SD classification statistics[11] are shown in Table 2.1. In algebraic classification procedure, the number of equivalence classes

---

[2] The second and the third class consists of self-dual functions of exactly three variables and non-self-dual functions of exactly two variables. The fourth and the succeeding classes consist of self-dual functions of exactly four variables and non-self-dual functions of exactly three variables.

quickly becomes large and inconvenient to use. So alternative means of classification such as will be considered in the following sections are particularly advantageous.

Number of input variables (n)	1	2	3	4	5	6
Total number of functions of upto n variables	4	16	256	65536	$\leq 4.3X10^9$	$\leq 1.8X10^{19}$
PN classification	3	6	22	402	1228158	$\leq 4X10^{14}$
NPN classification	2	4	14	222	616126	$\leq 2X10^{14}$
SD classification	1	3	7	83	109958	*
Total number of functions of exactly n variables	2	10	218	64594	$\leq 4.3X10^9$	$\leq 1.8X10^{19}$
PN classification	1	3	16	380	1227756	$\leq 4X10^{14}$
NPN classification	1	2	10	208	615904	$\leq 2X10^{14}$
SD classification	0	2	4	76	109875	*

Note: (1) The SD classification includes the hyperfunctions of n+1 variables.

(2) * represents entries not computed.

Table 2.1. The PN, NPN, and SD classification statistics for binary functions of upto 6 variables.

2.2 LINEARLY-SEPARABLE (THRESHOLD) FUNCTIONS

Linearly-separable, or threshold, functions form a particular sub-set of all functions, for which a numerical as distinct from an algebraic classification procedure is particularly straightforward. However, threshold functions of n variables form a diminishingly small proportion of all functions of n variables as n increases[11], (see Table 2.2).

n	3	4	5	6	7
All functions	256	65536	$\simeq 4.3X10^{9}$	$\simeq 1.8X10^{19}$	$\simeq 3.4X10^{38}$
Threshold fns.	104	1882	94572	15028134	8378070864

Table 2.2. The total number of binary functions and the total number of threshold functions for n = 3 through n = 7.

Linearly-separable functions are functions which, when their minterms are considered as 2^n equi-spaced nodes in n-dimensional space, possess a plane surface which unambiguously divides all true (f(x) = 1) nodes from all false (f(x) = 0) nodes. This is illustrated in Fig. 2.1(a).

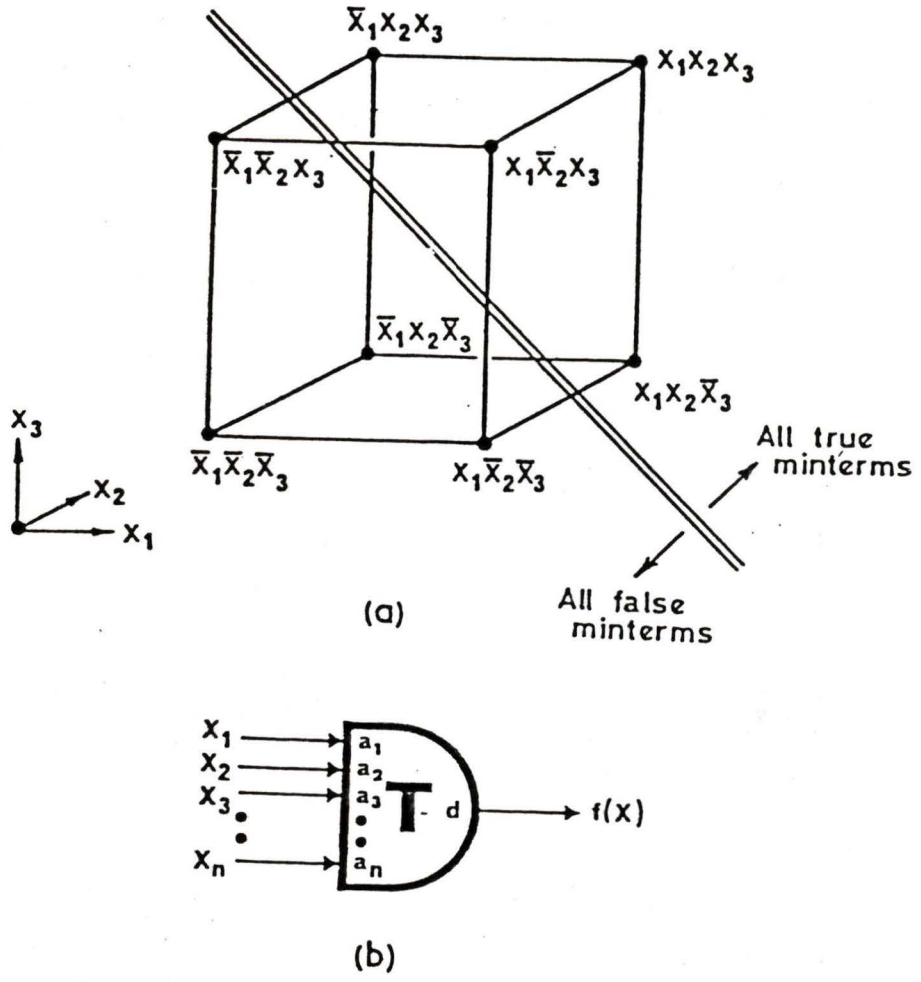


Fig. 2.1: Concepts of linear-separability and threshold functions.

a) the hypercube construction with a separating plane; function $f(x) = x_1x_2 + x_2x_3 + x_1x_3$

$= \langle 1.x_1 + 1.x_2 + 1.x_3 \rangle_2$

is illustrated.

b) the general symbol for a threshold-logic gate.

The equation of this plane in n-dimensional Euclidian space with axes $x_1, x_2, \ldots, x_n$ is

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = d \qquad \ldots (2.1)$$

where $a_1, a_2, \ldots, a_n$ and d are constants. Hence for any point lying on the plane or between the plane and the origin we have

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n <= d \qquad \ldots (2.2)$$

whilst for any point on the plane or on the side remote from the origin we have

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n >= d \qquad \ldots (2.3)$$

From this basic consideration it readily follows that the mathematical equation for a linearly-separable function may be written

$$f(x) = 0 \text{ if } a_1x_1 + a_2x_2 + \ldots + a_nx_n < d, \qquad \ldots (2.4)$$

and

$$f(x) = 1 \text{ if } a_1x_1 + a_2x_2 + \ldots + a_nx_n >= d. \qquad \ldots (2.5)$$

assuming the separating plane just passes through the lowest summation true minterm(s). This functional relationship may be paraphrased by the notation:

$$f(x) = < a_1x_1 + a_2x_2 + \ldots + a_nx_n >_d \qquad \ldots (2.6)$$

Note that normal arithmetic relationships hold in these considerations, and not modulo 2.

The threshold logic gate is illustrated in Fig. 2.1(b). The parameters $a_1, a_2, \ldots, a_n$ are termed the weights of the respective x_i input variables, giving their relative importance in determining the gate output state, whilst d is termed the gate threshold discrimination, or merely threshold. The value $a_i x_i$ at each minterm is termed the gate input summation, which must equal or exceed d to realize f(x) = 1. Further details of threshold logic theory, including tolerance considerations of the input weights and gate threshold discrimination are given in any standard reference (e.g. Muroga[10]).

2.2.1 Classification of Linearly-separable functions

We shall now briefly review the classification of threshold functions using Chow parameters based on the original work of C.K. Chow[16]. The basic Chow parameters of a function f(x) are defined as

$$CH(x) \triangleq Ch(x_1), Ch(x_2), \ldots, Ch(x_n); Ch(x_0) \quad \ldots (2.7)$$

where

$Ch(x_i)$ = occurrence of x_i taken over all true minterms; i = 1 to n,

and $Ch(x_0)$ = total number of true minterms.

For example, given $f(x) = x_1x_2 + x_1x_3$, we first expand the function into its three true minterms $x_1x_2\bar{x}_3 + x_1x_2x_3 + x_1\bar{x}_2x_3$, whence:

	$ch(x_1)$	$ch(x_2)$	$ch(x_3)$;	$ch(x_0)$
$x_1x_2\bar{x}_3$:	1	1	-	
$x_1x_2x_3$:	1	1	1	
$x_1\bar{x}_2x_3$:	1	-	1	
$CH(x) =$	3 ,	2 ,	2 ;	3

It will be apparent that each $Ch(x_i)$ value corresponds to the number of times each literal x_i occurs in the minterm expansion, ignoring the presence of $\bar{x}_i$.

However, the original Chow parameters can be modified to derive another integer-value vector which is more relevant for threshold classification purposes[3,10-15]. The modified-Chow parameters are defined as

$$B(x) \triangleq b_1, b_2, \ldots, b_n; b_0 \qquad \ldots (2.8)$$

where

$$b_i = 2Ch(x_i) - Ch(x_0); \; i = 1 \text{ to } n,$$

$$\text{and } b_0 = Ch(x_0) - 2^{n-1}.$$

The original Chow parameters are always positive integer values; the modified Chow parameters are integers too, but not necessarily all positive. Taking our previous example

with the original Chow parameters CH(x) = 3, 2, 2; 3, the modified-Chow parameters become B(x) = 3, 1, 1; -1. A useful property of these modified Chow parameters is that if a given function f(x) is independent of a particular variable x_i, then the associated b_i parameter is always zero-valued. However, the converse is not true, as may be shown by determining the Chow parameters for, say $f(x) = x_1 \oplus x_2$.

It may be simply shown that each linearly-separable function has one unique non-zero set of Chow parameters, and that no other function possesses this same set of values[10]. In order to guarantee uniqueness, both magnitude and sign are necessary in the modified set. Hence the modified Chow parameters are particularly appropriate and convenient for cataloguing threshold functions.

The Chow parameter classification table for all linearly-separable functions of upto 4 variables[11] is given in Table 2.3. Note that (conventionally) the b_i parameters are listed in descending order of magnitude, with no fixed correspondence to $b_1,\ldots,b_n$; b_0. The principal reason for such tabulations, however, is not merely as a classification table, but much more as a means to

1. determine whether any given function is linearly-separable or not.
2. if linearly-separable, then to give the minimum integer threshold realization for the function.

n	b_i	a_i
n <= 3		
1)	-4 0 0 0	1 0 0 0
2)	-3 1 1 1	2 1 1 1
3)	-2 2 2 0	1 1 1 0
n <= 4		
1)	-8 0 0 0 0	1 0 0 0 0
2)	-7 1 1 1 1	3 1 1 1 1
3)	-6 2 2 2 0	2 1 1 1 0
4)	-5 3 3 1 1	3 2 2 1 1
5)	-4 4 4 0 0	1 1 1 0 0
6)	-4 4 2 2 2	2 2 1 1 1
7)	-3 3 3 3 3	1 1 1 1 1

Table 2.3. The Chow parameter classification for all linearly-separable functions of n <= 4.

Note:

i) for the 21 n <= 5 entries, see Hurst[11].

ii) for the 135 n <= 6 entries, see Hurst[11].

iii) for the 2470 n <= 7 entries, see Winder[17].

Details of the full use of these classification tables may be found in[3,11,14], and a more detailed discussion of the theoretical development of the Chow parameter classification, and the relationship between the n+1 numerical parameters which embrace more than one NPN classification group

and SD algebraic classifications which likewise each embrace more than one NPN group, may be found in [3,8,10,12,15,16].

When we review the several possible classes of boolean functions, we find that the linearly separable class constitutes the most restrictive class. We may readily demonstrate that the n+1 Chow parameters are inadequate to define unambiguously functions of an unrestricted class - as an example all the Chow parameters of any Exclusive-OR or Exclusive-NOR function will be found to be zero, and hence do not uniquely identify the function. Thus the use of this particular set of only n+1 numerical parameters for function classification purposes has obvious limitations.

2.3 SPECTRAL CLASSIFICATION

Any given combinatorial function f(x) may be explicitly defined by its truthtable, which lists all 2^n input combinations, giving the function output value f(x) for each input combination. Conventionally, all x_i input variables and the output value of f(x) are expressed in {0,1} notation. With the order of tabulation of the input variables standardized, then the 2^n local output values of f(x) may be treated as a column vector Z defining f(x). Should we recode the binary logic values from {0,1} to {+1,-1}, respectively, then an alternative column vector Y defining f(x) will be obtained. Both the {0,1} and {1,-1} coding schemes for the function, $f(x) = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1x_2x_3$, are illustrated Table 2.4. The reason for such recoding can be found elsewhere[11].

Truthtable			Output vector	
x_3	x_2	x_1	Z	Y
0	0	0	0	1
0	0	1	1	-1
0	1	0	1	-1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	-1

Table 2.4: Truthtable for example 3-variable function $f(x) = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1x_2x_3$, and function vector Z and Y in {0,1} and {1,-1} coding respectively.

As an alternative to these 2^n entries we may also specify 2^n coefficients which define f(x), these coefficients being the 'spectral coefficients' of f(x); the coefficients may also be listed in a defined order to give a column vector representation, which we will subsequently identify by R or S according as the original functional specification is in {0,1} or {1,-1} coding respectively. As will be seen, the entries in R and S will no longer be binary values as in the local value truth table vectors Z and Y. We shall first consider the Hadamard transform $[T^n]$ which will directly generate S from Y and R from Z, following which we shall consider the use of these spectral coefficients for classification purpose.

2.3.1 The Hadamard orthogonal transform matrix

The Hadamard transform is a complete orthogonal square matrix, with row and column entries $\epsilon\{+1,-1\}$, and with a recursive structure as follows:

$$T^0 = [+1]$$

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

Note that the dimensions of the transform are $2^n \times 2^n$ for any n. For increasing n the transforms have the following structure:

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

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

etc.

For alternative definitions of Hadamard transforms and other representations of the Rademacher-Walsh transform, the reader should consult Hurst[11]. The recursive definition of the Hadamard transform is the most relevant for our purposes.

2.3.2 The Hadamard transformation of binary data

The Hadamard transform matrix may be used to transform any 2^n column vector of numbers into an alternative data column vector. No information is lost in this transformation, the original data being fully recoverable from the resulting vector by application of the inverse transform T^n. The spectrum R or S can be obtained from Z or Y using the following:

$$[T^n][Z] = R \quad \text{or} \quad [T^n][Y] = S \qquad \ldots (2.9)$$

Consider the 3-variable example function given in Table 2.4. Using the {0,1} coding scheme, we may transform Z with an n = 3 Hadamard transform as shown in Fig. 2.2. For clarity all entries in this transformation are identified. The tabulation order of the spectral coefficients and the notation used to represent these coefficients are also shown.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -3 \end{bmatrix} \begin{matrix} r_0 \\ r_1 \\ r_2 \\ r_{12} \\ r_3 \\ r_{13} \\ r_{23} \\ r_{123} \end{matrix}$$

$$[T^n] \qquad\qquad [Z] = [R]$$

Fig. 2.2: Labelling scheme in {0,1} coding.

If the output vector f(x) is recoded from {0,1} to {+1,-1}, then we have the alternative transformation shown in Fig. 2.3.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ -2 \\ -2 \\ -2 \\ 6 \end{bmatrix} \begin{matrix} s_0 \\ s_1 \\ s_2 \\ s_{12} \\ s_3 \\ s_{13} \\ s_{23} \\ s_{123} \end{matrix}$$

$$[T^n] \qquad\qquad [Y] = [S]$$

Fig. 2.3: Labelling scheme in {+1,-1} coding.

The spectra R and S each uniquely represent the given function f(x), since we can easily recover the original Z and Y using:

$$Z = 1/2^n(T^nR) \quad \text{and} \quad Y = 1/2^n(T^nS) \qquad \ldots (2.10)$$

The individual coefficient values in R and S are related as follows:

$$r_0 = 1/2 \ (\ 2^n - s_0 \ ),$$

$$r_i = - \ 1/2 \ (\ s_i \ ), \ i = 0. \qquad \ldots (2.11)$$

In subsequent discussions, we shall refer to individual spectral coefficients by r for {0,1} coding and by s for {1,-1} coding. Moreover, we re-group the coefficients and label them as follows:

s_0 : the zero-order coefficient

s_i, i = 1 to n : the primary or first order coefficients

s_{ij}, ij = 12, 13, ... : second-order coefficients

⋮

Collectively, all second and higher order coefficients will be termed the 'secondary' coefficients.

2.3.3 Invariant Spectral operations

It has been shown that there are in total five invariance operations which may be applied to the full set of 2^n spectral coefficients. These operations are merely stated below; formal proofs may be found in Edwards[18] and elsewhere[3,11,19].

Let $\propto$ be some substring, possibly empty, from {1,...,n}. Each invariant spectral operation transforms f --> f^*. A definition of f^* with the relevant spectral coefficient interchanges for each operation is listed below, for some function $f(x_1, .., x_i, .., x_j, .., x_n)$:

1. Permutation of any input variables x_i and x_j, $i \neq j$, $i,j \in \{1,2,..,n\}$:

   $$f^*(x) = f(x_1,..,x_j,..,x_i,..,x_n).$$

   This requires the interchange of 2^{n-2} pairs of coefficient values

   $$s_{i\propto} \quad <---> \quad s_{j\propto}, \qquad i,j \notin \propto .$$

Note, coefficients s_0, s_k, s_{ij},... remain unchanged.

2. Negation of any input variable x_i, $i \in \{1,2,..,n\}$:

$$f^*(x) = f(x_1,..,x_i,..,x_n).$$

This requires the negation of 2^{n-1} spectral coefficient values

$$s_{i\alpha} \quad \text{---->} \quad -s_{i\alpha}, \qquad i \notin \alpha .$$

Note, coefficients s_0, s_j, s_{jk},... remain unchanged.

3. Negation of a network output:

$$f^*(x) = \bar{f}(x).$$

This requires the negation of all 2^n spectral coefficients

$$s_0 \quad \text{---->} \quad -s_0$$

$$s_\alpha \quad \text{---->} \quad -s_\alpha$$

4. Replacement of any variable x_i by $x_i \oplus x_j$, $i = j$, $i,j \in \{1,2,..,n\}$:

$$f^*(x) = f(x_1,..,x_i \oplus x_j,..,x_j,..,x_n).$$

This requires the interchange of 2^{n-2} pairs of coefficient values

$$s_{i\alpha} \quad \text{<---->} \quad s_{ij\alpha}, \qquad i,j \notin \alpha .$$

Note, coefficients s_0, s_j, s_{jk},... remain unchanged.

5. Finally, replacement of the network output f(x) by $f(x) \oplus x_i$, $i \in \{1,2,..,n\}$:

$$f^*(x) = f(x) \oplus x_i.$$

This results in the interchange of 2^{n-1} pairs of spectral coefficients

$$s_{i\alpha} \quad <---> \quad s_\alpha, \qquad i \notin \alpha .$$

Note, all 2^n coefficients are involved in this operation.

The first three are just the NPN operations that we have seen before and they do not alter the order of any coefficient value; the fourth operation interchanges certain coefficient values from all orders except the zero order coefficient s_0, and involves 2^{n-1} coefficient moves; whilst the final operation brings the value of s_0 into the interchange, and moves all 2^n coefficients. Hence, operations (4) and (5), repeated as necessary, allow any spectral coefficient value to be moved to any other order, including the zero-order position s_0. The magnitudes of the complete set of coefficients, however, remain invariant under any of these operations.

2.3.4 Spectral Classification

The algebraic NPN classification method uses the first three of the five invariant spectral operations. Should the last two spectral operations be added to the NPN classification scheme, a more compact classification method will result. The addition of operation (4) results in a PN^2T classification, and the addition of operations (4) and (5) results in the PN^2TD classification. Two functions are in the same PN^2T or PN^2TD class, if the relevant spectral operations can transform one function to the other. A detailed discussion of such transformations can be found in Hurst[11]. The set of spectral coefficients for all functions in the same class remains the same, their order, sign and positions changing for the different functions.

Each class contains a canonic function, which is the function in the class for which the low order coefficients have maximum values. The canonic classification tables for functions of upto n variables is a listing of the coefficient values, in the normal identification order, which is s_0; $s_1,\ldots,s_n$; $s_{12},\ldots$; $s_{12\ldots n}$; with the zero and first -order coefficients s_0; $s_1,\ldots,s_n$, arranged in descending magnitude order. Thus s_0 is conventionally the highest magnitude coefficient in these canonic classification tables, followed by $s_1,\ldots,s_n$. Given a function f(x), the procedure of finding its canonic function $f_c(x)$ is called linearization and is described below in 2.3.4.1.

2.3.4.1 The linearization procedure

The linearization procedure is based on that of[19], to which the reader is referred for proof of the procedure. Let A and B be matrices of dimension n x n. We shall initially describe the procedure to find $f_C(x)$ for a PN^2T class, and later extend the procedure for a PN^2TD class.

1. Initialize B to be empty.
2. Select a spectral coefficient other than r_0 as follows:
   a) the coefficient(s) of largest magnitude,
   b) if more than one coefficient satisfies (a), then select the one(s) of lowest order,
   c) if more than one coefficient remains, then select the one with the largest decimal subscript.
3. Add the binary representation of the decimal subscript of the selected coefficient as a new column of B with the bit corresponding to x_i in the ith row. Now, delete the selected coefficient from the list.
4. Delete all spectral coefficients whose decimal subscripts have binary representation which are bit by bit mod 2 sums of some subset of the existing columns of B.
5. Repeat steps 2 to 4, (n - 1) times ignoring coefficients that have been deleted. Clearly the resulting B is non-singular. All summations are performed mod 2.

6. A is set to be the mod 2 inverse of B. This can be deduced using simple Gaussian elimination.

At the conclusion of this procedure, the columns of B define the spectral operations to be performed on f(x) to get $f_c(x)$. In particular, column i defines the XOR function that replaces x_i. The input variables to this XOR are those x_j for which the jth entry in the ith column of B is 1. The matrix A is used to determine the spectrum of the canonic function $f_c(x)$ corresponding to f(x).

Should the canonic function $f_c(x)$ for a PN^2TD class be required, then the above procedure must be extended to accommodate type (5) spectral operation. First, the largest magnitude coefficient over the coefficients excluding r_0 is determined. If this magnitude is less than or equal to $r_0 - 2^{n-1}$, no type (5) translation is required and the linearization is applied directly.

If the largest magnitude coefficient exceeds $r_0 - 2^{n-1}$, proceed as follows. If a first order coefficient has largest magnitude, a type (5) translation to exchange it with r_0 can be applied directly. If this is not the case, an initial type (4) translation that brings the largest magnitude coefficient to the first order is applied, followed by the type (5) translation. In either case, the linearization procedure is applied to the resulting spectrum.

The linearization procedure to find $f_c(x)$ for a PN^2T class is illustrated by applying it to the function f(x),

whose spectrum is given below in the normal identification ordering mentioned earlier:

R = 8; 0, 0, -2, 0; 0, -2, -2, 0, 0, 2; -2, 0, 2, 2; -6.

step 1 : B = 0

step 2 : $r_{1234} = -6$

step 3 :
$$B = \begin{matrix} 1 \\ 1 \\ 1 \\ 1 \end{matrix}$$

step 4 : ---

step 2 : i) $r_3 = r_{13} = r_{23} = r_{34} = r_{123} = r_{134} = r_{234} = 2$

ii) r_3 is chosen

step 3 :
$$B = \begin{matrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{matrix}$$

step 4 : delete r_{124}

step 2 : i) $r_{13} = r_{23} = r_{34} = r_{123} = r_{134} = r_{234} = 2$

ii) $r_{13} = r_{23} = r_{34} = 2$

iii) r_{34} is chosen

step 3 :
$$B = \begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}$$

step 4 : delete r_4, r_{12}, r_{123}

step 2 : i) $r_{13} = r_{23} = r_{134} = r_{234} = 2$

ii) $r_{13} = r_{23} = 2$

iii) r_{23} is chosen

step 3 :

$$B = \begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{matrix}$$

This is the final B and its columns define the spectral operations, namely,

x_1 <--- $x_1 \oplus x_2 \oplus x_3 \oplus x_4$,

x_2 <--- x_3,

x_3 <--- $x_3 \oplus x_4$,

x_4 <--- $x_2 \oplus x_3$.

step 6 : The mod 2 inverse of B is

$$A = \begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{matrix}$$

A allows the calculation of the spectral coefficients of the canonic function $f_c(x)$ using

$$A\gamma = \beta$$

where matrix multiplication is mod 2, γ is the binary representation of a coefficient subscript of f(x) and β is the binary representation of the position that this coefficient will occupy in $f_c(x)$.

For example,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad = \quad \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

indicating that r_{134} in f(x) becomes r_{124} in $f_c(x)$. The resulting spectrum for the example function is given below in normal identification ordering:

R_C = 8; -6, -2, 2, -2; 0, 0, 0, 0, 0, 0; -2, 2, -2, 2; 0.

2.3.5 Continued spectral classification

The zero and first order coefficients for the canonic PN^2TD classification entries for upto 4 and 5 variables[11] are shown in Appendices A and B respectively in {1,-1} coding. Several points concerning this classification will be highlighted.

1. With all the zero and first-order spectral coefficients made positive, it does not follow that all the higher order coefficients are likewise positive. All valid spectra have the property that the sum of the spectral coefficients must always total $\pm 2^n$, and this must be true for the canonic functions listed in the classification tables. Clearly, the full set of 2^n coefficients always contains redundancy, in that at least one coefficient could be removed.

2. The compactness of this classification procedure is very striking; all 65,536 functions of upto four variables are contained in the eight classes (Appendix A). For the 4.3×10^9 functions of upto five variables, the number of classes increases to 48 (Appendix B). The relative size estimates of the different 4 variable PN^2T classes[20] and 5 variable PN^2TD classes[4] are shown in Appendix C.

Comparing this complete classification for all functions with the linearly-separable classification of Table 2.3, it will be noticed that seven of the eight entries for n <= 4 in Appendix A are compatible to the entries in the Chow classification Table 2.3. For n <= 5, 21 of the 48 entries (Appendix B) appear in the Chow list[11]. Comparing the compatible entries of Table 2.3 and Appendix A we find:

1. the modified-Chow parameters b_i, i = 1 to n, introduced in the previous section are given by half the resultant spectral coefficients s_i, i = 1 to n; and
2. the modified-Chow parameter b_0 is given by the negation of half the spectral coefficient s_0.

Theoretically we may multiply the n modified-Chow parameters $b_1,\ldots,b_n$ by any constant without altering their information content, and similarly we may multiply the b_0 parameter by the same or any other constant. Thus there are an infinite range of Chow parameter definitions possible; the primary spectral coefficients $s_1,\ldots,s_n$, and the zero-order spectral

coefficient s_0 constitute one such definition. In the subsequent discussions, we will use this definition based on the spectrum, for the modified-Chow parameters. In addition we will drop the qualification 'modified' from here on, and refer to them as merely Chow parameters.

2.4 SUMMARY

In this chapter we have considered the classification of combinatorial logic functions, both in the function domain using an algebraic approach, and also in the spectral domain using operations on the spectral coefficients. All the relevant procedures of algebraic methods are encapsulated by available operations in the spectral domain, which in general prove to be more convenient and compact than the function domain procedures.

The Chow-parameter classification method for threshold functions, has direct ties to the subsequent evolvement of spectral coefficients, and the very interesting possibilities of using a reduced set of spectral coefficients for classification of all functions will be discussed in the next chapter.

Chapter III

SPECTRAL CLASSIFICATION

3.1 INTRODUCTION

Spectral techniques for functional classification were discussed in chapter 2. It was shown that given any function f(x) and its spectrum, there are five invariance operations namely:

- negation of a variable (N);
- negation of the function (N);
- permutation of the variables (P);
- spectral translation (T);
- disjoint spectral translation (D);

These operations can be used to reorder and sign-change the spectral coefficients so that a positive canonic ordering results. These operations divide the set of boolean functions into equivalence classes[3] with each class being uniquely defined by its canonic function. For subsequent discussions, let D denote a PN^2TD class.

---

[3] In this chapter the classification referred to is the PN^2TD classification.

We have already seen in chapter 2, how the 2^n spectral coefficients for any given function f(x) may be used as a ready means of function classification. A set of coefficients used to define exactly one boolean function is called the signature of the function. We shall use the term 'basis set' to refer to the zero and first order coefficients of a function when it is put in canonic form.

The Chow parameter classification for threshold classes has been shown to be a special case of spectral classification, the particular properties of functions belonging to these classes being that only n+1 of the full set of 2^n coefficients are necessary for classification purposes. The coefficients are precisely those that constitute the basis set which in this case is a signature. Thus, any function belonging to a threshold class is completely defined by the knowledge of the operations required to put it into canonic form and the n+1 coefficients defining the basis set of the canonic function.

Since the basis set containing n+1 coefficients is particularly compact and convenient for cataloguing functions from threshold classes, it was of interest to investigate whether the full set of 2^{2^n} functions could be classified using a signature containing less than 2^n coefficients for a particular n. The basis set does not constitute a signature for functions from non-threshold classes[11]. In other words, for non-threshold classes, there is more than one

function defined by the basis set and we shall refer to those functions with identical basis sets as solutions covered by the basis set. In order to extend the basis set to a signature, we need to augment the basis set with more coefficients. For threshold classes, there is only one solution covered by the basis set and so the basis set constitutes a signature. For non-threshold classes, the number of coefficients that need to be added to the basis set in order to derive a signature clearly depends on the number of solutions covered by the basis set.

In order to gain some insight into the number of coefficients that need to be added to the basis set to derive a signature, we researched for a possible correlation between the number of solutions covered by the basis set and a particular metric σ defined on the basis set. In order to develop this correlation, a search procedure was developed for extracting the solutions covered by the basis set, and a detailed discussion of this procedure is given in section 3.2. The definition of the metric σ and the underlying properties which are measured by this particular metric are also discussed in section 3.2.

A unate function is one in which no input variable x_i appears in both complemented and uncomplemented form (x_i and $\bar{x}_i$) in some minimized sum-of-products expression for f(x). For example the function $x_1\bar{x}_2 + \bar{x}_2x_3$ is a unate function, but $x_1\bar{x}_2 + x_2x_3$ is a non-unate function. Care must be taken

to ensure that the expression for f(x) is appropriately minimized; for example $f(x) = x_1 + \bar{x}_1\bar{x}_2$ is unate, because there exists the minimized form $f(x) = x_1 + \bar{x}_2$.

It may readily be shown that all linearly-separable functions must be unate [10,11]. The converse, however, does not hold, as may be illustrated by the simple unate but non-linearly-separable function $f(x) = x_1x_2 + x_3x_4$.

An equivalence class is termed unate if the canonic function $f_c(x)$ defining that class is unate in all its input variables. The non-threshold classes could be sub-divided into two categories based on this property, and termed unate classes and non-unate classes. Similarly to threshold classification, we derived a signature for functions using the solutions generated by the search procedure discussed in section 3.2. This numerical classification procedure uses a reduced set of coefficients, whose number varies from n+1 (minimum) to $(n^2 + 3n - 2)/2$ (maximum), the actual number necessary increasing as one moves from the most restrictive class of functions (linearly-separable) to the unrestricted class of all 2^{2^n} functions. A detailed discussion of this procedure for up to 5 variable functions can be found in section 3.3.

The computational complexity of the procedures adopted becomes unreasonable for n larger than 5. Consequently, in this chapter all conclusions reached are for $n < 6$.

3.2 SIGNIFICANCE OF THE METRIC

In this section we investigate the underlying properties which are measured by a particular metric σ defined on the basis set. We shall use Q to denote the basis set. A definition of the metric[21] is given below:

$$\sigma = \sum_{S_i \in Q} S_i^2 \qquad \ldots (3.1)$$

In order to investigate the correlation between the metric σ and the number of solutions covered by the basis set, we developed a procedure to extract all those solutions. A detailed discussion of this search procedure is given in the following subsection.

3.2.1 Search procedure for extracting solutions

The problem to be addressed is, given a basis set containing n+1 coefficients, to generate all the solutions covered by that basis set. The search is therefore to find all the F_i which satisfy the following system of equations.

$$T^* F_i = R^* \qquad \ldots (3.2)$$

where

T^* is a $(n+1) \times 2^n$ matrix consisting of a subset of the rows of T (the Hadamard transform),

R^* is a column vector containing the basis set and

F_i are the solutions covered by the basis set.

Instead of directly solving the system of equations (3.2), we have translated the problem into a straightforward sum of integers problem. This is done by translating the partial transform T^* and the basis set R^* to {0,1} coding, giving

$$T^+F_i = R^+ \qquad \ldots (3.3)$$

where

$$t^+_{ij} = 1/2 \; (\; t^*_{0j} + t^*_{ij} \;) \quad 1 <= i <= n+1, \; 0 <= j <= 2^n-1$$

$$t^+_{0j} = t^*_{0j} \qquad 0 <= j <= 2^n-1$$

$$r^+_i = 1/2 \; (\; r^*_0 + r^*_i \;) \qquad 1 <= i <= n+1$$

$$r^+_0 = r^*_0$$

For a 4 variable function, the partial transform T^+ is given below:

$$T^+ = \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \ldots (3.4)$$

Let {B} be the set of integers {0, ..., 2^n - 1}. If we ignore the first row of the partial transform T^+ shown above, we find that the column vectors of the matrix represent the binary encoding of the elements of {B}, decreasing in magnitude from left to right. For a 5 variable function, the set {B} contains the integers {0, ..., 31}. Considering the

first row of the partial transform T^+, the system of equations defined by 3.3 requires a set of r^+_0 columns to be selected from T^+. This selection is analogous to selecting r^+_0 values from set {B}. Let {C} be the set of all possible selections of r^+_0 elements of set {B}, and let each element of {C} be thought of as a r^+_0-tuple.

Considering rows 2 through n+1 in the partial transform T^+, we could define d as follows:

$$d = r^+_i 2^{i-1} \qquad \ldots (3.5)$$

The sum of all the projections of an element of {C} should equal d to conform to the system of equations 3.3. Each such conforming C_i from the set {C} could be used to evaluate a solution F_i covered by the basis set.

This search procedure is illustrated by the following example.

Example :

Consider the basis set { 7, 7, 3, 3, 3, 3 }, for the canonic function of 5 variable class 23. Equation 3.4 shows the partial transform T^+ for a 4 variable function. After transforming R^* to {0,1} coding, we have:

$$R^{+} = \begin{matrix} 7 \\ 7 \\ 5 \\ 5 \\ 5 \\ 5 \end{matrix}$$

Using equation 3.5 we have:

$$d = 157; \quad (7 + 10 + 20 + 40 + 80)$$

The equivalence class in the example has 3 solutions covered by the basis set and the set {C} corresponding to these solutions is given below:

$$\{C\} = \begin{matrix} (& 7 & 15 & 23 & 25 & 27 & 29 & 31), \\ (& 11 & 15 & 21 & 23 & 27 & 29 & 31), \\ (& 13 & 15 & 19 & 23 & 27 & 29 & 31). \end{matrix}$$

3.2.2 Properties measured by the metric

Using the search procedure discussed in the previous subsection, the solutions covered by the basis set were enumerated for all the 4 and 5 variable equivalence classes. The number of solutions for each of the 4 and 5 variable equivalence classes are given in Appendix A and B respectively.

Hypothesis 3.2.2.1

For n <= 5, given two classes D_i and D_j with corresponding σ_i, σ_j and p_i, p_j the number of solutions covered by the basis set, then $p_i >= p_j$ if and only if $\sigma_i <= \sigma_j$.

Discussion:

All the equivalence classes for up to 3 variables are linearly-separable and hence have only one solution. Using equation 3.1, the values of σ for each of the 4 and 5 variable equivalence classes were computed and tabulated (Appendix A & B). For this definition of σ , Hypothesis 3.2.2.1 is easily disproved using the following counter example from Appendix B.

$D_i = 22$	$D_j = 41$
$\sigma_i = 608$	$\sigma_j = 600$
$P_i = 28$	$P_j = 12$

Five variable class 41 is unate in all its input variables whereas class 22 is non-unate in 4 of the 5 input variables. This led us to believe that the property of unateness has a role to play and hence should be incorporated in the definition of σ in order to consider hypothesis 3.2.2.1. As a result σ was modified as follows:

$$\sigma = \left(\sum_{S_i \in Q} S_i^2 \right) - (n-u)n \qquad \ldots (3.6)$$

where

U = number of unate input variables in the function.

n = total number of input variables.

For n = 4 and n = 5, the σ_m values for all the equivalence classes were computed and are shown in Appendix A and B

respectively. The correlation between the metric σ_m and the solutions covered by the basis set defined in Hypothesis 3.2.2.1 was then verified. A counter example to this hypothesis was not found using the modified definition of σ, and as a result this correlation could be used to predict the relative magnitude of the number of solutions covered by the basis set for an equivalence class.

3.3 SIGNATURES

As an initial attempt, a signature for all functions encountered can be constructed using 2^{n-1} coefficients. This elementary bound follows immediately by considering the n variable function as 2^{n-3} distinct 3 variable functions. Since we know that any 3 variable function has a signature using four coefficients[4], the result follows.

However, it was of interest whether we could do better than 2^{n-1} coefficients for the functions encountered. How much better depends entirely on how many times it is necessary to partition a function to get sub-functions in the threshold classes. The partitioning procedure used is Shannon's decomposition and it is defined below:

---

[4] Every 3 variable class is a threshold class; the basis set constitutes a signature for threshold classes.

Definition:

A function f(x) of n variables, can be partitioned using Shannon's decomposition about some input variable x_i, $1 <= i <= n$, into two sub-functions $f_0(x)$ and $f_1(x)$ of n-1 variables as follows:

$$f(x) = \bar{x}_i f_0 (x_1..,x_{i-1},x_{i+1},..x_n) + x_i f_1(x_1..,x_{i-1},x_{i+1},..x_n)$$

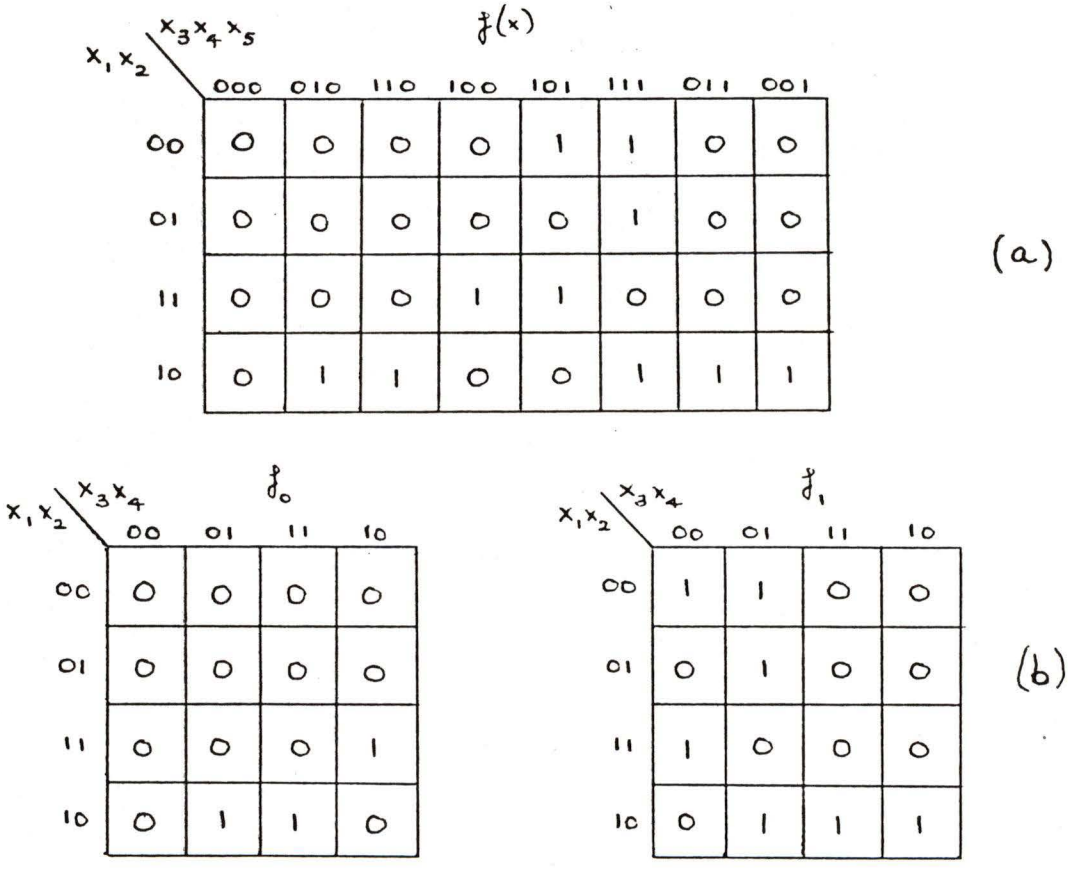


Fig. 3.1: (a) Map of the canonic function of class 40.
(b) Four variable partitions f_0 and f_1 of the canonic function of class 40.

We shall illustrate the partitioning procedure for one of the 5 variable classes which has a large number of possible solutions for the basis set -- class 40, with 457 solutions. The canonic function of class 40 is illustrated in the map (Fig. 3.1a). Partitioning about input variable x_5 gives the two 4 variable functions f_0 and f_1 illustrated in Fig. 3.1b.

Upon inspection it becomes apparent that both f_0 and f_1 are in 4 variable threshold classes and hence both f_0 and f_1 could be classified using their respective basis sets. As a consequence equivalence class 40 could be classified using only 10 coefficients -- the basis sets for the respective sub-functions. However, this depends entirely on whether there exists a partitioning scheme such that one of the resulting sub-functions at each partitioning stage belongs to a threshold class.

Hypothesis 3.3.1

If a canonic function $f_C(x)$ of n variables is partitioned using Shannon's decomposition about some input variable x_i, $1 <= i <= n$, into two sub-functions $f_0(x)$ and $f_1(x)$ of n-1 variables, then either f_0 or f_1 belongs to a threshold class.

Discussion :

The canonic functions of all the 5 variable non-threshold classes were partitioned using Shannon's decomposition and the resulting sub-functions were tested for their membership

in threshold classes. At least one of the resulting sub-functions belonged to a threshold class.

Conclusion :

A canonic function $f_c(x)$ of n input variables can always be partitioned into two sub-functions of n-1 variables, such that at least one of the resulting sub-functions belong to a threshold class. A counter example to this hypothesis has not been found.

Hypothesis 3.3.2

A signature for all functions encountered could be constructed using only ((n)(n+1)/2) - 2 coefficients derived from the spectra of sub-functions resulting from a Shannon's decomposition.

Discussion :

This result depends on hypothesis 3.3.1. Since it is possible to partition a canonic function into two sub-functions such that at least one of the sub-functions is in a threshold class, the upper bound on the number of coefficients required is drastically reduced from 2^{n-1} to ((n)(n+1)/2) - 2, a much more reasonable number. This result can be proved recursively using 3 variable classes as base cases.

Since every 3 variable class is a threshold class, they have a signature of 4 coefficients. A 4 variable function could be partitioned into two 3 variable sub-functions and

as a result the 4 variable function would have a signature of 4+4 coefficients. Similarly, a 5 variable function would lead to two 4 variable sub-functions where the threshold sub-function is classified using 5 coefficients and the other 4 variable sub-function is classified using at most 4+4 coefficients. If we now consider an n variable function, we need to perform Shannon's decomposition (n-3) times; the number of coefficients in the signature is given by:

$$\left(\sum_{i=4}^{n} i\right) + 4$$

$$= \left(\sum_{i=1}^{n} i\right) - \left(\sum_{i=1}^{3} i\right) + 4$$

$$= \left(\frac{n(n+1)}{2}\right) - 2$$

Conclusion :

The upper bound on the number of coefficients needed to construct a signature for all functions is clearly ((n)(n+1)/2) - 2.

By Hypothesis 3.3.2, the signature for a function may be derived from the spectra of sub-functions resulting from a Shannon's decomposition. This classification scheme is cumbersome because the coefficients that constitute such a signature do not correspond to the coefficients in the spectrum of the function under investigation. The transformation from a signature containing sub-function coefficients to one containing coefficients from the spectrum of the function under investigation is computationally complex[22]. Hence

an alternative approach by which we could form a signature by directly selecting a set of coefficients from the spectrum of the function under investigation was developed. A discussion of this approach follows.

From Appendices A & B, we find that only threshold classes have a unique solution. All other non-threshold classes have more than one solution and hence the basis set needs to be augmented with more coefficients in order to derive a signature for non-threshold classes. Our subsequent discussions on signatures are based on the results compiled using 5 variable PN^2TD classes and these results were verified for 4 variable classes.

Our goal is to add coefficients to the basis set in order to uniquely identify the solutions covered by the basis set. As a worst case, let us suppose that the coefficients in the complete spectrum all have the same magnitude (except for r_0); they would have different signs in order to maintain the sum of all the coefficients at $\pm 2^n$. If this were true, then all we would have is the sign information to distinguish the different solutions covered by the basis set. As a result we would need k coefficients to uniquely identify 2^k solutions. On the other hand, if the coefficients in the spectrum have different magnitudes, then we could use less than k coefficients to uniquely identify 2^k solutions.

3.3.1 Unate classes

Equivalence class 41 is unate and has the maximum number of 12 solutions among unate classes (Appendix B). In order to uniquely classify these 12 solutions for class 41, we need to augment the basis set with (at most) four more coefficients to derive a signature. This forms the upper bound on the number of coefficients required to construct a signature for 5 variable unate classes. In other words, there exists a set of 10 coefficients in the spectrum for 5 variable unate classes that would constitute a signature.

However the 10 coefficients may be in different positions for the various unate classes. In order to form a particular set of coefficients that could be used for the identification of all the unate classes, it may be necessary to add more coefficients to the signature.

Hypothesis 3.3.1.1.

1. For unate classes, 2n + 1 coefficients form a signature.
2. The coefficients that constitute the signature for unate classes can be selected in one of the two ways mentioned below:
   a) Basis set augmented by any n second order coefficients from the top half of the spectrum.
   b) Basis set augmented by any n third order coefficients from the bottom half of the spectrum.

The solutions covered by the basis set were generated using the search procedure discussed in section 3.3.1 for four and five variable unate equivalence classes. This set of solutions was subsequently used to test Hypothesis 3.3.1.1. and the hypothesis was verified.

3.3.2 Non-unate classes

The threshold and unate classes are equivalence classes restricted by particular properties, so these restricted classes could be viewed as special cases of the unrestricted class of functions. Hence, a signature for the unrestricted class of functions should constitute a signature for both the threshold and unate classes, because they are special cases.

For unate classes, the basis set augmented by n coefficients constituted a signature. From Appendix B, we see that the number of solutions covered by the basis set is greater than 2^n for certain non-unate classes. Hence as a worst case, for non-unate classes with all coefficients except for r_0 equal in magnitude, 2n + 1 coefficients clearly do not constitute a signature. The information pertaining to the number of solutions covered by the basis set for the different classes (From Appendix B) allows us to conclude that at most 13 coefficients would be required to uniquely identify these solutions for this category of functions.

But in order to define some particular set of coefficients that could be used as a signature, we might need to include more than 13 coefficients in addition to the basis set. The actual number of coefficients that need to be added to the basis set depends on the information content of the coefficients selected to form a signature. It has been shown by Miller et al[23] that information is not evenly distributed in the spectrum; the higher order end of the spectrum contains more information than the lower order end. As a consequence, in order to derive a minimal signature, we might have to select coefficients from the higher order end of the spectrum.

For a 5 variable function, we have:

1. one zero and one fifth order coefficient (r_0, r_{12345}),
2. n first and fourth order coefficients,
3. 2n second and third order coefficients.

The basis set contains the zero and first order coefficients. So, we are left with the other orders 2, 3, 4 and 5 for the selection of 13 coefficients. It is clear from the number of coefficients in each of these orders that no single order is sufficient. We need to select a mixture of coefficients from these four orders, preferably from the higher order end of the spectrum.

Hypothesis 3.3.2.1

1. For non-unate equivalence classes, (n-1) + (n * (n+1))/2 coefficients constitute a signature. In other words, we need (n * (n+1))/2 - 2 additional coefficients to uniquely identify the solutions covered by the basis set.
2. The basis set augmented by the last (n * (n+1))/2 - 2 coefficients constitutes a signature for non-unate equivalence classes (where the coefficients are in binary order -- see chapter 2).

The solutions covered by the basis set were generated using the search procedure discussed in section 3.2.1 for four and five variable non-unate equivalence classes. This set of solutions was subsequently used to test hypothesis 3.3.2.1. and a counter example to that hypothesis was not found.

3.4 SUMMARY

In this chapter we have derived a numerical classification procedure to identify individual functions without ambiguity, using a reduced set of coefficients. We have already seen that linearly-separable classes have a sufficient signature containing n+1 coefficients, where the coefficients are precisely those that become the zero and first order coefficients when the function is put in canonic form.

We have shown that functions in unate classes can be identified by inspecting the basis set augmented by either 'n' second order coefficients selected from the top half of the spectrum, or by 'n' third order coefficients selected from the bottom half of the spectrum when the function is put in canonic form. In all, 2n + 1 coefficients are sufficient to form a signature.

The non-unate equivalence classes could be classified using the basis set augmented by the last ((n * (n+1))/2 - 2 coefficients (where the coefficients are in binary order -- see chapter 2).

These results are of particular importance because they could lead to standardized testing and fault-diagnosis procedures. A new design philosophy utilizing this spectral classification technique is discussed in subsequent chapters.

Chapter IV

DIGITAL SYNTHESIS USING SYMMETRIES

4.1 INTRODUCTION

The philosophy of digital synthesis using symmetry techniques supplemented by spectral methods is briefly reviewed in this chapter. It must be noted that the extraction of conventional prime implicant factors for digital synthesis, is not given priority. In viewing the symmetric properties of a function, freedom to consider groups of minterms of possibly mixed values and separated by Hamming distances of greater than one is present, as will be apparent in the subsequent sections.

We shall limit our discussion to two-variable symmetries. The two variable symmetry types can be classified into two groups, the first group associated with Hamming distances of two and the second group with Hamming distances of one. A Hamming distance of 1 refers to any two minterms which differ in the value of one variable only, for example $x_1x_2x_3x_4$ and $x_1x_2x_3\bar{x}_4$. Similarly, a Hamming distance of 2 relates to two minterms which differ in the value of two variables only, for example $x_1x_2x_3x_4$ and $x_1x_2\bar{x}_3\bar{x}_4$, and so on. It must be noted that Hamming distances of greater than 2 are not relevant for symmetries in two variables. A brief

description of the different symmetries[24] is given in Appendix F.

The detection of the symmetry conditions in any given function f(x) clearly is a prerequisite to any design method based upon their presence. A function possessing a symmetry is said to be decomposable. A method of detection of symmetries in individual functions[25] is briefly presented in Appendix G and a possible digital synthesis scheme using this property is presented in the section 4.2.

The proportion of non-decomposable functions increases with increasing n (number of variables). All 3-variable functions are decomposable, but this is not true for n greater than 3. As a consequence, digital synthesis using symmetries alone is not feasible. For non-decomposable functions, symmetry techniques can be supplemented by spectral translation methods and a brief discussion of this approach to digital synthesis is presented in section 4.3. For this to give a design method for all functions, it would be necessary for every spectral class to contain a function with a symmetry.

4.2 SYNTHESIS USING SYMMETRIES

The simple example shown in Fig. 4.1(a) is used to illustrate the synthesis method, and to indicate the basis of comprehensive design algorithms. For a detailed discussion, the reader should refer to Edwards et al[24]. The figures

used for illustration in this section are from Edwards et al[24].

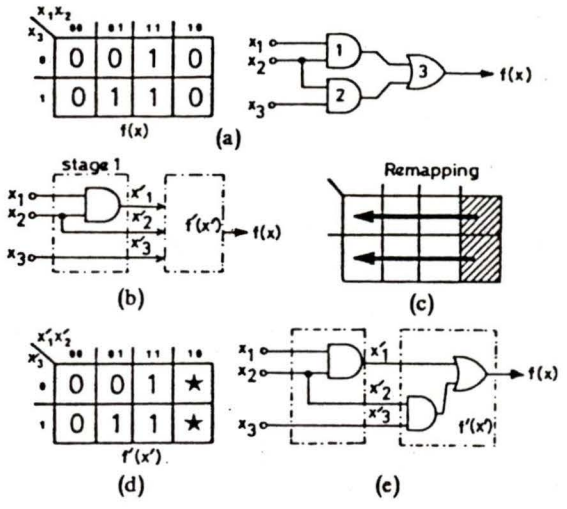


Fig. 4.1 : A simple example of remapping:

(a) Given function f(x) and conventional realization.

(b) First stage synthesis.

(c) Remapping produced by first-stage synthesis.

(d) Resultant second-stage function $f'(x')$, with don't care area $x'_1x'_2$.

(e) Realization for f(x) based upon (d).

Suppose the circuit shown in Fig. 4.1(a) is redrawn as in Fig. 4.1(b), where the AND gate 1 of Fig. 4.1(a) has now been treated as a synthesis stage in a nondisjoint decomposition. The general module for this stage, shown dotted, has inputs x_1, x_2, x_3 and outputs x'_1, x'_2, x'_3. Function $f'(x'_1, x'_2, x'_3)$ represents the remainder function to be synthesized. The relationships between x_1, x_2, x_3 and x'_1, x'_2, x'_3 are given in Table 4.1.

x_1	x_2	x_3	x'_1	x'_2	x'_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	1	1

Table 4.1: Relationship between x_1, x_2, x_3 and x'_1, x'_2, x'_3.

Clearly $x_1\bar{x}_2\bar{x}_3$ has mapped to $\bar{x}'_1\bar{x}'_2\bar{x}'_3$, and $x_1\ x_2\ x_3$ has mapped to $\bar{x}'_1\bar{x}'_2x'_3$, with the remaining input minterms mapping unchanged. This is shown in Fig. 4.1(c). Notice that this remapping is consistent with our given function, as $f(1, 0, 0) = f(0, 0, 0)$ and $f(1, 0, 1) = f(0, 0, 1)$.

For the second stage remainder function $f'(x'_1, x'_2, x'_3)$ the input vector $x'_1\bar{x}'_2$ can never occur, and hence these minterms may be allocated don't-care values as shown in

Fig. 4.1(d). If we allocate 1 to these don't-cares, $f'(x') = x'_1 + x'_2x'_3$, giving the realization shown in Fig. 4.1(e), which of course is the same as the original realization of Fig. 4.1(a).

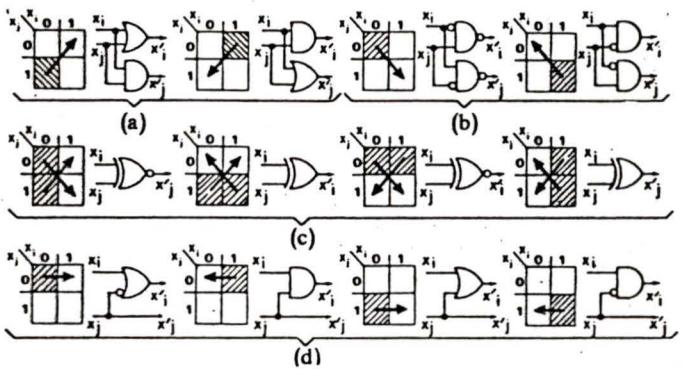


Fig. 4.2 : Remapping functions for different classes of symmetry along with an example realization.

(a) NES $\{x_i, x_j\}$.

(b) ES $\{x_i, x_j\}$.

(c) MS $\{x_i, x_j\}$.

(d) $\{SVS\ x_i\}\bar{x}_j$ and $\{SVS\ x_i\}x_j$.

However, what is illustrated in this example is the difference in design philosophy between a conventional design method and the new method. In Fig. 4.1(a) the AND gate 1 is conventionally associated with the prime implicant factor x_1x_2; under the new method, however, this AND gate is associated with the symmetry which exists in the given function f(x) between the minterm areas $\bar{x}_1\bar{x}_2$ and $x_1\bar{x}_2$, that is in n-space $\bar{x}_2$, and is being used as a remapping function to capitalize upon this symmetry.

Each of the types of symmetry detailed in section 4.2 is associated with appropriate remapping functions, which serve to pass to the next level of realization a reduced set of minterm conditions, thus allowing don't-cares to be incorporated at the next level of realization. The remapping functions for all of these symmetries along with an example of their realization are shown in Fig. 4.2.

4.3 SYNTHESIS USING SPECTRAL METHODS

We have already seen in chapter 2, how the spectral coefficients for any n-variable function f(x), may be rearranged under the five invariance operations to provide a canonic classification for any binary function. We shall now consider briefly, the significance of the five invariance operations for possible network synthesis.

The permutation of two input variables x_i and x_j has no useful significance for synthesis purposes, because it

involves only a relabelling of the input variables. Similarly, network synthesis using either (or both) of the next two invariance operations on the spectral coefficients, that is negation of any one or more input variables and negation of the whole function, are clearly not useful in a synthesis method[18]. However, let us now consider the last two of the five possible invariance operations. These will be seen to be potentially useful.

First consider the replacement of any input variable x_i by the exclusive-OR signal $[x_i \oplus x_j]$, $i <> j <> 0$, as illustrated in Fig. 4.3(a). This is the type (4) spectral operation discussed in chapter 2, and it results in a function $f'(x)$, whose spectral coefficients are given by the interchange of the coefficient values of $f(x)$ as follows:

$$r_{i\alpha} \text{ <---> } r_{ij\alpha} \qquad [\text{ Chapter 2 }]$$

that is in all coefficients which contain i delete j if it also appears and append it if it does not appear, leaving unchanged all coefficients not containing i in their subscripts.

For the type (5) spectral operation discussed in chapter 2, if a feed-forward path as shown in Fig. 4.3(b) is made, then the spectral coefficients for $f''(x)$ are given by the interchange of the coefficient values of $f(x)$ as follows:

$$r_{i\alpha} \text{ <---> } r_{\alpha} \qquad [\text{ Chapter 2 }]$$

that is in all 2^n coefficients append i if it does not appear and delete i if it does appear. The functions $f'(x)$ and $f''(x)$ that result by applying the last two invariant spectral translations respectively are called residual or core functions. Formal proof of these final two invariance operations may be found in Edwards[18].

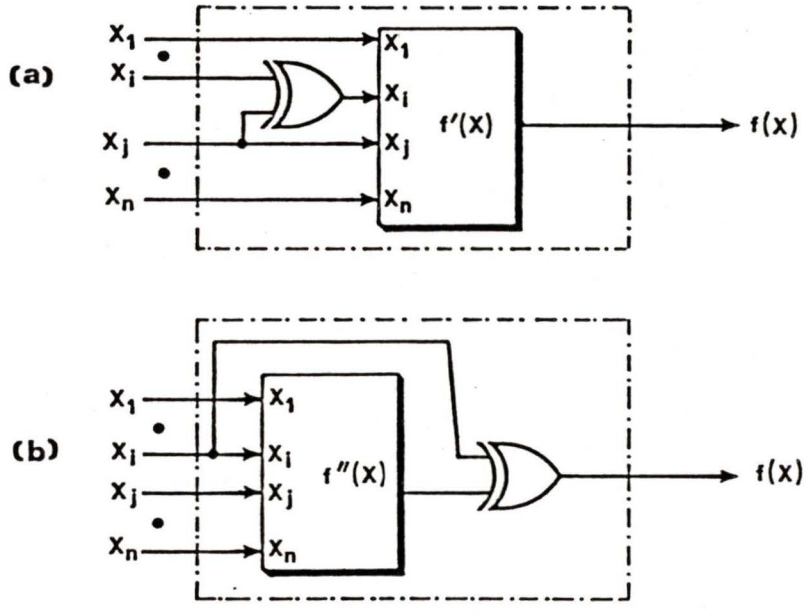


Fig. 4.3 : Digital synthesis using spectral operations.
(a) Given function f(x), with input XOR.
(b) Given function f(x), with output XOR.

4.4 A NEW DESIGN PHILOSOPHY

In chapter 2 we were principally concerned with the **classification** of functions, whereby many dissimilar functions could be transformed into a standard canonic function using spectral translations. Our interest is now in deriving a core function, using either/both of the two spectral translations. If this core function is a simple one in comparison with the original function f(x), then we have a possibly useful means of synthesizing f(x). This results in a modular design for the realization of function f(x). If type (4) operations are used then we call it a prefilter module and if type (5) operations are used then it is called a postfilter module. These modules implementing the spectral translations consist of XOR gates. Thus any function f(x) can be realized by supplementing the module used to implement the core function $f'(x)$ by appropriate filter modules.

In the case of digital synthesis using symmetry techniques supplemented by spectral methods, our criterion is to find a core function $f'(x)$ that is decomposable, so that we can take advantage of its symmetries in a realization. The last two invariance operations could be used to yield a core function which possesses two variable symmetries, whereas the original function f(x) had no symmetries. For cases where the given function f(x), that needs to be synthesised, is non-decomposable, we could adopt a strategy where the first stage synthesis uses spectral techniques to yield a

decomposable core function, and subsequent stages use synthesis based on symmetries. This new design philosophy is illustrated using the following example.

Example :

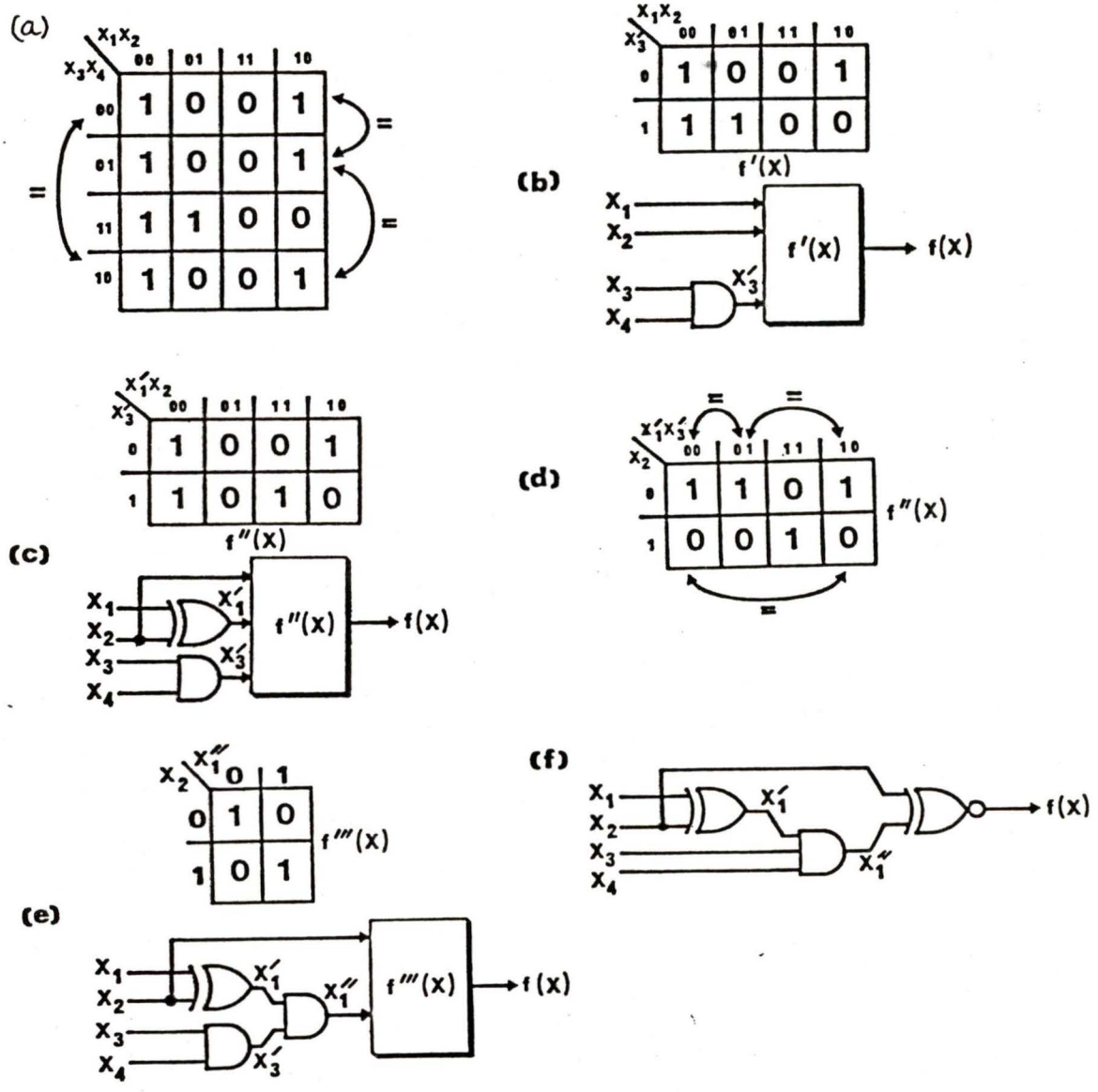
The Karnaugh map for the example function to be synthesized is shown in Fig. 4.4(a). Using the algorithm described in Appendix F the following symmetries were identified in the example function:

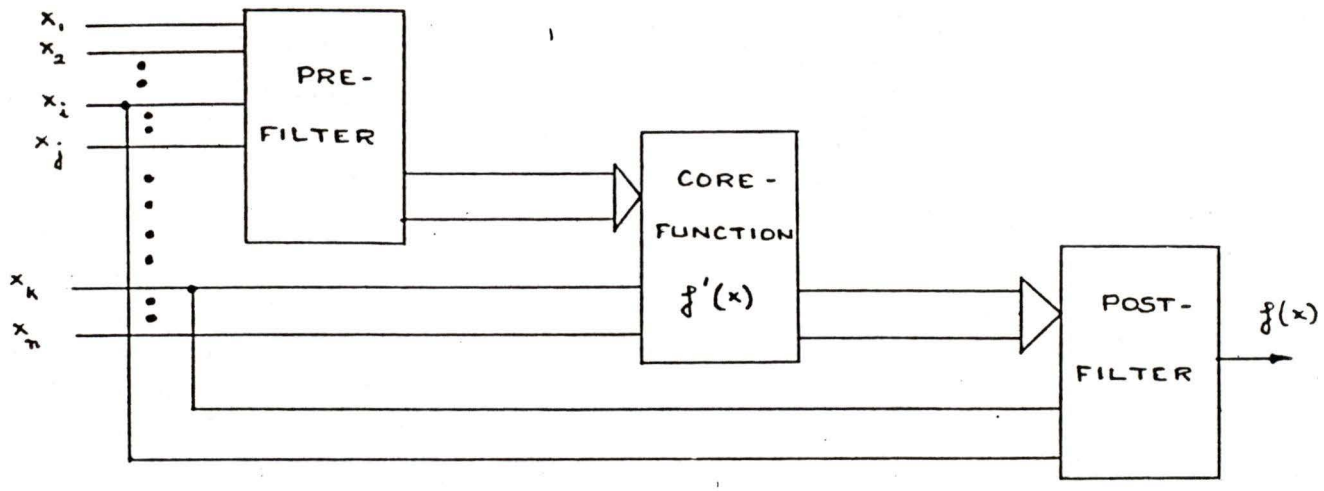
1. Non-equivalence symmetry NES $\{x_3, x_4\}$,
2. Multiple single variable symmetries $\{\text{SVS } x_3\}\bar{x}_4$ and $\{\text{SVS } x_4\}\bar{x}_3$ and
3. Single variable symmetries $\{\text{SVS } x_1\}\bar{x}_3$ and $\{\text{SVS } x_1\}\bar{x}_4$.

Using the collective symmetries (a) and (b), we may reduce the size of the remaining function to be synthesized by combining these three identical rows into one row by a first level AND remapping function. In effect we are making two adjacent minterm rows of the map into don't cares, and then discarding them entirely. This gives us the part realization shown in Fig. 4.4(b).

Now if we apply a spectral translation operation to replace input x_1 by $[x_1 \oplus x_2]$, this will interchange the $\bar{x}_1 x_2$ and $x_1 x_2$ minterms as shown in Fig. 4.4(c). The reason for this operation is that we now have in the residual function of Fig. 4.4(c) multiple symmetries in x'_1 and x'_3. We may more clearly see this if we replot Fig. 4.4(c) with alternate axes, as shown in Fig. 4.4(d).

A remapping function to utilize these symmetries and reduce still further the size of the residual function is another AND remapping function as shown in Fig. 4.4(e). The final residual function is now merely the exclusive-NOR of these two inputs, which if the two two- input AND gates are combined into one three input AND gate gives us the final realization shown in Fig. 4.4(f). In fig. 4.4(g) we have given a general strategy for this new design approach.





(g)

Fig. 4.4 : A simple example of our new design philosophy:

a) the function = $\bar{x}_1\bar{x}_2 + \bar{x}_2(\bar{x}_3 + \bar{x}_4) + \bar{x}_1x_3x_4$

b) first symmetry reduction

c) spectral translation between x_1 and $x_1 \oplus x_2$

d) remapping of (c) to show x_1', x_3' symmetries

e) second symmetry reduction

f) final realization

g) general strategy for the new design approach

4.5 SUMMARY

In this chapter we have reviewed a number of possible techniques for the synthesis of combinatorial logic networks, principally using conventional vertex and XOR gates.

Digital synthesis using symmetry techniques supplemented by spectral operations allows us the possibility of establishing a small set of 'standard core functions', one from each equivalence class that is decomposable. Any function f(x) that belongs to an equivalence class may be synthesised by the implementation of the appropriate linear pre-filter and/or linear post-filter in addition to the standard core function $f'(x)$ for that class.

The feasibility of such a digital synthesis entirely depends on the presence of a decomposable core function in every equivalence class. The feasibility of such a synthesis method as well as its relevance to standardized fault detection procedures are discussed in the next chapter.

Chapter V

FEASIBILITY OF DIGITAL SYNTHESIS USING SYMMETRIES

5.1 INTRODUCTION

This chapter discusses the feasibility of digital synthesis using symmetry techniques supplemented by spectral methods. In chapters 2 and 3, we developed a classification method based on spectral operations. The first four spectral operations lead to a PN^2T classification, and the addition of the last spectral operation results in a PN^2TD classification. In subsequent discussions, we shall use C to refer to a PN^2T class and D to refer to a PN^2TD class.

The feasibility of our new design philosophy is contingent on the existence of a decomposable core function $f'(x)$ in every spectral class. It was hypothesised by Miller[28] that every C type spectral class would possess a decomposable core function. For functions of up to four input variables, there are 18 C type classes and 8 D type classes and these are shown in Appendix D. It is easily shown that every such C type class possesses a core function $f'(x)$ that is decomposable.

The number of C and D type classes increases to 191 and 48 respectively for functions of up to to five variables and

these are shown in Appendix E. One or more C type classes constitute a D type class. If we were to allow post filters in our design then the feasibility of our new design approach depends on the existance of a decomposable core function in a D type class. The decomposability of 5-variable D type spectral classes and the feasibility of our new design philosophy is discussed in this chapter.

5.2 DECOMPOSABILITY OF SPECTRAL CLASSES

For digital synthesis using symmetry methods supplemented by spectral techniques to be possible for 5-variable functions, we need a decomposable core function $f'(x)$ from every 5-variable D type class. In order to test the non-decomposability of a D type class, we need to test all its constituent C type classes. The canonic functions of all of the 191 C type classes shown in Appendix E were tested for the property of decomposability, using the algorithm outlined in Appendix G. Of the 191 classes, the canonic functions of 160 were found to be decomposable. The remaining 31 C type classes whose canonic functions were non-decomposable are shown in Table 5.1.

In chapter 2 we discussed the linearization procedure to obtain $f_C(x)$, given $f'(x)$. Now, we need to generate all possible core functions $f'(x)$, given $f_C(x)$. The core functions thus generated can be checked for decomposability using the algorithm discussed in Appendix G. As an initial

PN^2TD Class	PN^2T Class
25	25A (96); 25B (97)
26	26A (101);
29	29A (113); 29B (114)
31	31A (125); 31B (126); 31C (127)
32	32A (131);
33	33E (139);
34	34A (141);
36	36A (148); 36B (149); 36C (150)
38	38A (155); 38B (156)
40	40A (164); 40B (165); 40C (166); 40D (167)
41	41A (168); 41B (169); 41C (170)
42	42E (175);
44	44A (179); 44C (181); 44D (182)
45	45A (183);
46	46C (187);
48	48A (190); 48B (191)

Table 5.1 : 5 variable classes whose canonic functions are non-decomposable.

strategy, it was decided to consider a comparatively small number of possible core functions from each of these 31 C type classes.

```
X1  <--- X1 xor X2
X1  <--- X1 xor X3
X1  <--- X1 xor X4
X1  <--- X1 xor X5
X1  <--- X1 xor X2 xor X3
X1  <--- X1 xor X2 xor X4
X1  <--- X1 xor X2 xor X5
X1  <--- X1 xor X3 xor X4
X1  <--- X1 xor X3 xor X5
X1  <--- X1 xor X4 xor X5
X1  <--- X1 xor X2 xor X3 xor X4
X1  <--- X1 xor X2 xor X3 xor X5
X1  <--- X1 xor X2 xor X4 xor X5
X1  <--- X1 xor X3 xor X4 xor X5
X1  <--- X1 xor X2 xor X3 xor X4 xor X5

Similarly, for X2, X3, X4, and X5 (75 translations).
```

Fig. 5.1 : Translations resulting from considering the the first column of matrix B defined in the linearization procedure in Chapter 2.

For an n-variable function, the dimension of the B matrix defined in the linearization procedure is n X n. Let us first consider all possible spectral translations that can

be applied to a canonic function $f_c(x)$, using a single column of B. There are (n * (n + 1))/2 such translations for a single column of B and these translations for the first column in B are shown in Fig. 5.1. If each column is treated independently, then we would have a total of (n * (n * (n+1)/2)) such translations and for a 5-variable function, there are 75 such translations.

The 75 translations discussed above were applied in an initial attempt to get a decomposable core function $f'(x)$ from each of the remaining 31 C type classes shown in Table 5.1. The 22 C type classes which revealed a decomposable core function $f'(x)$ as a result of applying the 75 translations were eliminated from Table 5.1 and the remaining 9 C type classes that had failed to reveal a decomposable core function are shown in Table 5.2. It is worth noting that these 9 C type classes completely constitute the 3 D type classes { 40, 41, 48 }.

In order to exhaustively test the non-decomposability property of the C type classes shown in Table 5.2, we need to consider all possible combinations of the columns of the B matrix. This would require a total of $(n * (n+1))^n$ different spectral translations that need to be applied to a canonic function $f_c(x)$, to yield core functions $f'(x)$. For a 5-variable class, there are 15^5 such translations and hence we have an upper bound on the number of core functions $f'(x)$ that need to be generated and checked for

PN^2TD Class	PN^2T Class
40	40A (164); 40B (165); 40C (166); 40D (167)
41	41A (168); 41B (169); 41C (170)
48	48A (190); 48B (191)

Table 5.2 : Classes from Table 5.1 that failed to reveal a decomposable function after applying the 75 translations shown in Fig. 5.1.

decomposition. But, the number of individual functions in each of the 5-variable C type classes is clearly less than the number of possible spectral translations that need to be applied, and this is evident from the size estimates[3] of the 5-variable D type classes shown in Appendix C. This implies that

1. we can arrive at certain core functions $f'(x)$, by applying at least two different sets of spectral translations to the canonic function $f_c(x)$.
2. since matrix B needs to be non-singular, it is possible that certain translations are not applicable for a particular $f_c(x)$.

Considering the amount of computation involved in this approach, a systematic procedure to generate all the core functions $f'(x)$, given $f_c(x)$, without having to perform all

the 15^5 different spectral translations, is necessary in order to verify the non-decomposability of a C type class. Even if this were possible, the amount of computation that would be necessary in order to declare a C type class as non-decomposable would be enormous because the algorithm involves the following steps:

1. generate a core function f(x) using the linearization procedure.
2. check whether f(x) is decomposable.
3. If f(x) is non-decomposable, then go to step 1 else stop.

Hence there was a need for an alternative algorithm.

An alternative approach with the potential to do considerably better in terms of the amount of computation involved is presented in the following section. The efficiency of this alternative algorithm depends on the spectral summary of the canonic function under investigation, but it does far better than the previous algorithm.

5.3 NON-DECOMPOSABILITY ALGORITHM FOR A CLASS

An important feature of the algorithm discussed in this section is the means by which the nonexistence of decomposable functions in a class is verified. Instead of attempting to generate every possible core function in the class, all the **decomposable** functions that might belong to the class are generated, and checked for membership in the class. A detailed discussion of this search procedure follows.

In our discussion of the search algorithm to find a 2-variable symmetry in a class, we shall limit ourselves to a PN^2T classification. As a consequence of this limitation, the number of true minterms (r_0) remains invariant for all the functions in an equivalence class.

We have already seen in the previous chapter the rationale for partitioning the given function f(x) into four quarters, where each quarter represents the minterm areas 00, 01, 10, and 11 respectively for some pair of variables x_i and x_j. Let us for convenience use the notation a, b, c and d to refer to the number of true minterms in the four quarters of the function, with 'a' representing the first quarter, 'b' representing the second quarter and so on. But, we are free to arbitrarily assign different minterm areas to the different quarters. In fact there are 2^4 ways of assigning the four different minterm areas 00, 01, 10, 11 over a pair of two variables x_i and x_j.

Since our primary goal is to test the decomposability of a class,

1. there is no need to distinguish between the different types of symmetries discussed in the previous chapter and
2. we only need to consider one possible pair of input variables x_i and x_j over which a symmetry is present.

Hence,

1. we can arbitrarily choose any one of the 2^4 assignment schemes mentioned above, since if any one of the 16 functions is in the equivalence class under investigation, so are the other 15. The only difference will be in the type of symmetry that is present or absent. In our discussion of the search algorithm, we shall assign minterm areas 00, 01, 10, and 11 to 'a', 'b', 'c' and 'd' respectively.
2. using the same rationale, we can further reduce our search space by applying the following restriction:

$$a \geq b \geq c \geq d \qquad \ldots (5.1)$$

3. we can arbitrarily choose the last two quarters 'c' and 'd' to exhibit a 2-variable symmetry. So the minterm areas corresponding to c and d are identical and hence we have

$$c = d \qquad \ldots (5.2)$$

Equation 5.2 allows us to replace 'd' by 'c'.

The number of true minterms in the given function f(x), (r_0 value), is now distributed among the four quarters 'a', 'b', 'c' and 'd' in all possible ways, and the entries that do not conform to the equations 5.1 and 5.2 are eliminated. The different stages of the algorithm are illustrated using the C type class 40D as an example. Let us assume that x_1

is the most significant variable and x_n the least significant variable. The r_0 value for class 40D is 10 and hence we need to allocate the 10 true minterms to the four quarters of the function 'a', 'b', 'c' and 'd' in all possible ways using equations 5.1 and 5.2 as guidelines. The conforming entries for class 40D are given in Table 5.3.

a	b	c	d
8	2	0	0
7	3	0	0
6	4	0	0
5	5	0	0
8	0	1	1
7	1	1	1
6	2	1	1
5	3	1	1
4	4	1	1
6	0	2	2
5	1	2	2
4	2	2	2
3	3	2	2
4	0	3	3
3	1	3	3
2	2	3	3
2	0	4	4
1	1	4	4
0	0	5	5

Table 5.3: Possible ways of distributing the 10 true minterms to the 4 quarters 'a', 'b', 'c' and 'd' where columns 'c' and 'd' are identical.

We have already seen in chapter 2 that the set of spectral coefficients for all functions in the same class

remains the same, only their order, sign, and positions changing for the different functions. As a result, we can evaluate the frequency of the set of coefficients that appear in the spectrum of the canonic function $f_c(x)$ of a class and refer to it as the spectral summary. The spectral summary need not necessarily be unique for the different classes. For example 5-variable classes 41C, 42B and 43B all have the same spectral summary of (1 x 11, 5 x 5, 10 x 3, 16 x 1), but have unique spectral signatures. Since the value of r_0 remains invariant in a PN^2T classification, we can eliminate r_0 from the spectral summary and refer to the resulting spectral summary by {K}. The class chosen as an example for illustration has a unique spectral summary and after deletion of r_0, class 40D has a spectral summary {K} = {10 x 4, 15 x 2, 6 x 0}. In subsequent discussions, we shall refer to {K} as the spectral summary.

Since we know the number of true minterms in the four quarters of $f_c(x)$, we can evaluate the spectral coefficients r_1, r_2 and r_{12} based on that information. Using equation (5.2) we have:

$$r_0 = a + b + 2c \quad \ldots (5.3)$$

$$r_1 = a + b - 2c \quad \ldots (5.4)$$

$$r_2 = a - b \quad \ldots (5.5)$$

$$r_{12} = - r_2 \quad \ldots (5.6)$$

Equations 5.3 + 5.4 yield:

$$2 (a + b) = r_0 + r_1 \text{ where } r_1 \in \{K\} \quad \ldots (5.7)$$

Since 2c is always even, $r_0 + r_1$ should have the same parity as r_0 (Equation 5.4). These equations allow us to eliminate non-conforming entries from Table 5.3, and the resulting entries are shown in Table 5.4.

a	b	c	d
5	1	2	2
4	2	2	2
3	3	2	2
4	0	3	3
3	1	3	3
2	2	3	3

Table 5.4: Possible combinations resulting from filling four columns with ten true minterms, where columns 'c' and 'd' are identical and the values of the spectral coefficients r_1, r_2 and r_{12} belong to {10, 4, 2}.

With just the values of a, b, c and d available, these are the only coefficients that could be evaluated. At this stage we cannot evaluate all the 2^n spectral coefficients because the rows corresponding to those coefficients in the Hadamard transform do not have identical entries in the partitions considered. Subsequent partitioning of these four columns 'a', 'b', 'c' and 'd' would enable us to evaluate all the 2^n spectral coefficients and these coefficients could then be checked against the spectral summary of the

equivalence class under investigation. For an n-variable equivalence class, we need to perform n-1 steps in order to evaluate the complete spectrum of the function. We start the search with column 'a' containing a quarter of function f(x). In other words, column 'a' contains 2^{n-2} minterms at the beginning of the search and performing (n-1) steps allows us to partition column 'a' at each successive step, eventually resulting in 2^{n-2} partitions where each partition contains a single minterm.

<table>
<tr><td colspan="8">a</td></tr>
<tr><td colspan="4">a_0</td><td colspan="4">a_1</td></tr>
<tr><td colspan="2">a_{00}</td><td colspan="2">a_{01}</td><td colspan="2">a_{10}</td><td colspan="2">a_{11}</td></tr>
<tr><td>a_{000}</td><td>a_{001}</td><td>a_{010}</td><td>a_{011}</td><td>a_{100}</td><td>a_{101}</td><td>a_{110}</td><td>a_{111}</td></tr>
</table>

Fig. 5.2 : Labelling scheme for column 'a' at each step.

For a 5-variable function, we need to perform 4 partitioning steps, and the labelling scheme for partitioning column 'a' is illustrated in Fig. 5.2 at each partioning step. The 8 minterms that constitute column 'a' at the start of the search procedure result in 8 sub-partitions after all the partitioning steps and each of these 8 sub-partitions contain a single minterm at the end of the search procedure.

We can now perform the remaining partitioning steps and the spectral coefficients that are defined by the partitions at each step can be evaluated and checked against the spectral summary of the equivalence class under investigation. The non-conforming entries are eliminated at each step. If after performing the necessary (n-1) partitioning steps we do not find a decomposable function, then we have an equivalence spectral class that is non-decomposable.

The second stage partitioning is now performed, and this allows us to evaluate the coefficients r_3, r_{13}, r_{23} and r_{123} because the respective rows in the Hadamard transform now have identical entries in each of these 8 partitions. The different possible ways of assigning the number of true minterms in 'a', 'b', 'c' and 'd' into its sub-partitions is now considered. From the information provided by the sub-partitions, the spectral coefficients r_3, r_{13}, r_{23} and r_{123} are evaluated using the equations given below.

$$r_3 = a_0 + b_0 + 2c_0 - (a_1 + b_1 + 2c_1). \quad \ldots (5.8)$$

$$r_{13} = a_1 + b_1 + 2c_0 - (a_0 + b_0 + 2c_1). \quad \ldots (5.9)$$

$$r_{23} = a_1 + b_0 - (a_0 + b_1). \quad \ldots (5.10)$$

$$r_{123} = -r_{23} \quad \ldots (5.11)$$

The coefficients thus evaluated should conform to the spectral summary, or else those entries will be eliminated from Table 5.4.

The third stage of the partitioning scheme is now attempted, and the equations that define the spectral coefficients r_4, r_{34}, r_{24}, r_{234}, r_{14}, r_{134}, r_{124}, and r_{1234} are given below.

$$r_4 = a_{00} - a_{01} + a_{10} - a_{11} + b_{00} - b_{01} + b_{10} - b_{11} + 2(c_{00} - c_{01} + c_{10} - c_{11}) \quad \ldots (5.12)$$

$$r_{34} = -a_{00} + a_{01} + a_{10} - a_{11} - b_{00} + b_{01} + b_{10} - b_{11} + 2(-c_{00} + c_{01} + c_{10} - c_{11}) \quad \ldots (5.13)$$

$$r_{24} = -a_{00} + a_{01} - a_{10} + a_{11} + b_{00} - b_{01} + b_{10} - b_{11} \quad \ldots (5.14)$$

$$r_{234} = a_{00} - a_{01} - a_{10} + a_{11} - b_{00} + b_{01} + b_{10} - b_{11} \quad \ldots (5.15)$$

$$r_{14} = -a_{00} + a_{01} - a_{10} + a_{11} - b_{00} + b_{01} - b_{10} + b_{11} + 2(c_{00} - c_{01} + c_{10} - c_{11}) \quad \ldots (5.16)$$

$$r_{134} = a_{00} - a_{01} - a_{10} + a_{11} + b_{00} - b_{01} - b_{10} + b_{11} + 2(-c_{00} + c_{01} + c_{10} - c_{11}) \quad \ldots (5.17)$$

$$r_{124} = a_{00} - a_{01} + a_{10} - a_{11} - b_{00} + b_{01} - b_{10} + b_{11} \quad \ldots (5.18)$$

$$r_{1234} = -a_{00} + a_{01} + a_{10} - a_{11} + b_{00} - b_{01} - b_{10} + b_{11} \quad \ldots (5.19)$$

The efficiency of the algorithm is enhanced further if the bit patterns in the respective rows of the Hadamard transform are considered. This procedure is illustrated using stage 3 of our algorithm for a 5-variable function. The Hadamard transform for a 3-variable function (Fig. 5.3) is

used here for illustration, since the transform for a 5-variable function is too large to be presented. Higher orders of the transform may be constructed using the recursive definition given in chapter 2.3.1. We shall rearrange the rows of the Hadamard transform shown in Fig. 5.3 to obtain the transform shown in Fig. 5.4, for reasons which will become clear later. Since $T^nZ = R$, we can partition the rows of the Hadamard transform at each stage of our algorithm to correspond to the partitions of the function 'a', 'b', 'c', and 'd'. Let us label the partitions in the transform as a^*, b^*, c^*, and d^* respectively.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad \begin{matrix} r_0 \\ r_3 \\ r_2 \\ r_{23} \\ r_1 \\ r_{13} \\ r_{12} \\ r_{123} \end{matrix}$$

Fig. 5.3: The Hadamard transform for a 3-variable function.

For a 5-variable function, at stage 3 of our algorithm, each partition of the function would contain two minterms and for a 3-variable function, we will have two minterms in each partition at the first stage of our algorithm (Fig. 5.4). The possible bit patterns in the partitions of the Hadamard transform corresponding to the partitions of the function are 11, $1\bar{1}$, $\bar{1}1$ and $\bar{1}\bar{1}$.

1	1	1	1	1	1	1	1	r_0
				Stage 1				
a^*		b^*		c^*		d^*		
1	1	1	1	-1	-1	-1	-1	r_1
1	1	-1	-1	1	1	-1	-1	r_2
1	1	-1	-1	-1	-1	1	1	r_{12}
				Stage 2				
a^*_0	a^*_1	b^*_0	b^*_1	c^*_0	c^*_1	d^*_0	d^*_1	
1	-1	1	-1	1	-1	1	-1	r_3
1	-1	-1	1	1	-1	-1	1	r_{23}
1	-1	1	-1	-1	1	-1	1	r_{13}
1	-1	-1	1	-1	1	1	-1	r_{123}

Fig. 5.4: The Hadamard transform with the rows rearranged for a 3-variable function.

At this stage, information pertaining to function specification is available in partitions of two minterms. Consequently, we need to select those rows in the transform that have identical entries in the partitions considered at this stage for evaluation of spectral coefficients. Referring back to Fig. 5.4, we see that the rows in the transform corresponding to the coefficients r_0, r_1, r_2, and r_{12} have identical entries in the partitions a^*, b^*, c^*, and d^*. The other

coefficients cannot be evaluated at this stage because the functional specification for each minterm is not available. This is transparent if we consider the bit patterns in the partitions for the remaining rows of the transform. The spectral coefficients corresponding to these rows will be evaluated in subsequent stages of our algorithm. Further inspection reveals that the first row contains all 1's and this row corresponds to the coefficient r_0. But, r_0 was given; so this coefficient need not be re-evaluated. In subsequent stages, there will be other such coefficients that have already been evaluated and we need to eliminate those rows in the transform. This can be done by backtracking one stage. If 11 and $\bar{1}\bar{1}$ are the possible bit patterns for the partitions in the transform at the present stage, then 1111, $11\bar{1}\bar{1}$, $\bar{1}\bar{1}11$, and $\bar{1}\bar{1}\bar{1}\bar{1}$ are the possible bit patterns for the previous stage. Identical entries in the partitions of the transform at this previous stage would have been used for coefficient evaluation in that stage itself and hence we could eliminate the rows with the four-bit patterns 1111 or $\bar{1}\bar{1}\bar{1}\bar{1}$. This eliminates the row corresponding to the coefficient r_0 from the transform and this leaves us with the task of evaluating r_1, r_2, and r_{12} at this stage.

If a similar procedure is adopted for a 5-variable function at stage 3, we would have accomplished two things:

1. selected the coefficients to be evaluated at this stage, and

2. the rows in the Hadamard transform corresponding to these coefficients would have four-bit patterns $1\bar{1}\bar{1}\bar{1}$ or $\bar{1}\bar{1}111$ in the corresponding partitions of the previous stage.

For example, if a_0 was 3 in stage 2 of our algorithm, then this would generate {(3,0), (2,1), (1,2) (0,3)} as possible pairs of values for a_{00} and a_{01} respectively at stage 3 of our algorithm for a 5-variable function. Since the four-bit patterns corresponding to the partitions a_0, a_1, b_0, b_1, c_0 and c_1 in the selected rows of the Hadamard transform for the coefficients under evaluation are either $1\bar{1}\bar{1}\bar{1}$ or $\bar{1}\bar{1}111$, we have the following:

For bit pattern $1\bar{1}\bar{1}\bar{1}$,

$$a_{00} - a_{01} = \{(3-0), (2-1), (1-2), \text{and } (0-3)\}$$
$$= \{(3), (1), (-1) \text{ and } (-3)\}$$

For bit pattern $\bar{1}\bar{1}111$,

$$a_{01} - a_{00} = \{(0-3), (1-2), (2-1) \text{ and } (3-0)\}$$
$$= \{(-3), (-1), (1) \text{ and } (3)\}$$
$$= a_{00} - a_{01}$$

Let us use the superscript notation

$$a^0 = a_{00} - a_{01}$$
$$a^1 = a_{10} - a_{11},$$

and let b^0 b^1, c^0 and c^1 be defined similarly. Using this superscript notation, the equations 5.12 to 5.19 are redefined below:

$$r_4 = a^0 + a^1 + b^0 + b^1 + 2\,(c^0 + c^1) \quad \ldots(5.20)$$

$$r_{34} = -a^0 + a^1 - b^0 + b^1 + 2\,(-c^0 + c^1) \quad \ldots(5.21)$$

$$r_{24} = -a^0 - a^1 + b^0 + b^1 \quad \ldots(5.22)$$

$$r_{234} = a^0 - a^1 - b^0 + b^1 \quad \ldots(5.23)$$

$$r_{14} = -a^0 - a^1 - b^0 - b^1 + 2\,(c^0 + c^1) \quad \ldots(5.24)$$

$$r_{134} = a^0 - a^1 + b^0 - b^1 + 2\,(-c^0 + c^1) \quad \ldots(5.25)$$

$$r_{124} = a^0 + a^1 - b^0 - b^1 \quad \ldots(5.26)$$

$$r_{1234} = -a^0 + a^1 + b^0 - b^1 \quad \ldots(5.27)$$

Let Q and P be matrices defined as follows:

$$Q = \begin{bmatrix} a^0 + a^1 + b^0 + b^1 \\ -a^0 + a^1 - b^0 + b^1 \\ -a^0 - a^1 + b^0 + b^1 \\ a^0 - a^1 - b^0 + b^1 \end{bmatrix}$$

$$P = \begin{bmatrix} 2 * (c^0 + c^1) \\ 2 * (-c^0 + c^1) \\ 0 \\ 0 \end{bmatrix}$$

The spectral coefficients defined by the equations 5.20 to 5.23 could be computed by adding the matrices Q and P. The spectral coefficients defined by the equations 5.24 to 5.27 could be computed by evaluating (-Q + P). Similar enhancements to the efficiency of the algorithm could be made at every stage of the partitioning scheme.

Further stages of partitioning are performed until the complete spectrum is evaluated and checked against the spectral summary of the class under investigation.

5.4 CONTINUED DECOMPOSABILITY OF SPECTRAL CLASSES

The algorithm discussed in the previous section can be used to test the decomposability of a C type class. In order to test the decomposability of a D type class, we need to apply the same algorithm to each constituent C type class.

Of the 5-variable D type classes that are yet to reveal a decomposable core function $f'(x)$ (Table 5.2), class 40 has a unique spectral summary. A unique spectral summary allows us to quickly identify functions from an equivalence class merely by looking at the frequency of occurrence of the different magnitude coefficients; their position and sign need not be considered. As a consequence class 40 was chosen for investigation and all its four constituent C type classes, 40A, 40B, 40C and 40D were tested for non-decomposability using the algorithm discussed in the previous section.

To generate each and every 5 variable core function f(x) and then to check for its decomposability would have taken roughly a day of CPU time on a VAX 11/780 system for a C type class (15^5 * 0.9 sec). The efficiency of the algorithm developed by us made possible the investigation of the 5-variable C type classes 40A, 40B, and 40C which took

roughly 100 minutes each while class 40D took about 50 minutes of CPU time on a VAX 11/780 system. The 5 variable PN^2TD class 40 was non-decomposable and as a result digital synthesis using symmetry techniques supplemented by spectral operations is not feasible for that class. Although 2 variable symmetries are the strongest form of symmetry, we could consider weaker (more than 2 variables) symmetries. These weaker symmetries and their detection have been studied[26]; however their further relevance to digital synthesis needs to be pursued.

5.5 FAULT DETECTION

Fault detection techniques based on data compression have been pursued by a number of authors in parallel with more conventional methods based on test sets. One such data compression method, called syndrome testing[29] (r_0 testing), checks for a difference between the number of true minterms in the faulty and fault-free functions. Syndrome testing of two-level realizations and PLA's has been investigated for single stuck-at-faults[30,31]. This testing approach has also been generalized to spectral coefficient testing[32,33], this being known as r_α testing. The spectral technique compares the value of r_α for the faulty and fault-free functions. It is a direct generalization of syndrome testing in that the syndrome is one particular coefficient of the whole set.

In our new design philosophy we would realize any function from a given equivalence class by supplementing the module implementing the decomposable core function with appropriate filter modules. Even if a class has a decomposable core function, the realization of that function is often not two-level and hence is not syndrome testable. Moreover, the testing of the filter modules is rather more complex (assuming that the filter modules can only be viewed from beyond the module implementing the core function). It can be hypothesised that the testing of these filter modules would involve the verification of m particular spectral coefficients where m is the number of XOR gates in the filter module.

5.6 SUMMARY

In this chapter we presented an alternative algorithm for testing the non-decomposability property of a spectral class, using fast transform principles to enhance efficiency. Instead of generating all the core functions that belong to an equivalence class by using the linearization procedure and then testing each core function for decomposability, we have enumerated all decomposable functions that could possibly belong to an equivalence class and checked for their presence in the class under consideration.

The 5-variable PN^2TD class 40 was found to be non-decomposable and consequently digital synthesis using symmetry

methods supplemented by spectral operations is not feasible for certain classes of functions. The only other PN^2TD classes which may be non-decomposable are classes 41 and 48, all the remainder having been shown to be decomposable.

Chapter VI

CONCLUSIONS

The spectral techniques introduced in chapter 2 represent possibly one of the more significant advances in switching theory and applications for many years, and could alter many of our present methods for logic design, fault diagnosis, and associated tasks. As was mentioned earlier, among the benefits to be realized from study of these techniques are an increased understanding of functions which have some common property, insight into the realization of all functions in the same equivalence class and standardization of testing and fault diagnosis procedures for a particular class.

A correlation between the number of solutions covered by the basis set and a particular metric σ defined on the basis set was established. An algorithm was developed to identify the different solutions covered by the basis set. Use of the correlation and the algorithm made possible the classification of all Boolean functions by using at most $O(n^2)$ spectral coefficients. As an initial attempt, functions from unate classes were classified using only $2n + 1$ coefficients. The coefficients that constitute a signature for these classes can be either the basis set augmented by any n second-order coefficients from the top half of the spectrum

or the basis set augmented by any n third-order coefficients from the bottom half of the spectrum.

Functions from threshold classes and unate classes form but a small proportion of the full set of 2^{2^n} functions, and hence a classification scheme using a reduced set of coefficients is desirable for functions from the remaining unrestricted (non-unate) classes. By considering functions of 5 variables, it was hypothesized that those functions can be classified using $(n^2 + 3n - 2)/2$ coefficients where the coefficients that constitute the signature are precisely the basis set augmented by the last $(n^2 + n - 4)/2$ coefficients in the spectrum (when the coefficients are arranged in binary order as shown in chapter 2).

The results discussed above are based on the spectral classification of functions of up to 5 variables; the unavailability of spectral classification tables and the amount of computation involved in their enumeration prevented us from pursuing verification of the results for higher values of n. A more complete theoretical description of all these results is desirable before attempting to extend them to higher values of n.

An efficient algorithm based on fast transform principles was developed to exhaustively test the non-decomposability property of spectral classes. Use of this algorithm made possible the feasibility study of our new design philosophy which requires every spectral class to possess a

decomposable function. The non-decomposability of every function in 5 variable PN^2TD class 40 shows that the usefulness of this philosophy is not universal and is limited to certain classes of functions. Even if a class has a decomposable core function, the realization of that function is often not two-level and hence is not syndrome testable. Moreover the testing of the filter modules is rather more complex (assuming that the filter modules can only be viewed from beyond the module implementing the core function). It can be hypothesised that the testing of these filter modules would involve the verification of m particular spectral coefficients where m is the number of XOR gates in the filter module.

Although 2 variable symmetries are the strongest form of symmetry, we could consider weaker (more than 2 variables) symmetries; however their further relevance to digital synthesis needs to be pursued.

BIBLIOGRAPHY

1. Walsh, J.L., **A closed set of orthogonal functions**, American J. Math., 45, 5-24, 1923.

2. Rademacher, H., **Einige Satze uber reihen von allgemeinen orthogonal funktionen**, Math. Annen, 87, 112-138, 1922.

3. Dertouzos, M.L., **Threshold Logic: A Synthesis Approach**, MIT Press, Cambridge, Mass., 1965.

4. Lechner, R.J., **Harmonic analysis of switching functions**, In recent developments in switching theory, Academic Press, New York, 1971.

5. Staff of Harvard Computational Laboratory, **Synthesis of Electronic Computing Systems**, Harvard University Press, Cambridge, Mass., 1957.

6. Slepian, D., **On the number of symmetry types of Boolean functions on n variables**, Can. J. Math. 5(2), 185-93, 1954.

7. Elspas, B., **Self-complementary symmetry types of Boolean functions**, Trans. IRE EC9 264-70, 1960.

8. Goto, E., and Takahasi, H., **Some theroms useful in threshold-logic for enumerating Boolean functions**, Proc. IFIP Congress, 747-51, Aug. 1962.

9. Harrison, M.A., **Combinatorial Problems in Boolean Algebras and Applications to the Theory of Switching**, Ph.D. Thesis, Electrical Engineering Department, University of Michigan, 1963.

10. Muroga, S., **Threshold Logic and its Applications**, John Wiley Interscience, New York, 1971.

11. Hurst, S.L., **The Logic Processing of Digital Signals**, Crane-Russak, New York, Edward Arnold, London, 1978.

12. Winder, R.O., **Fundamentals of Threshold Logic**, Air Force Cambridge Lab. Report No. 1, Contract AFCRC-68-0066, Jan. 1968.

13. Lewis, P.M., and Coates, C.L., **Threshold Logic**, John Wiley, New York, 1967.

14. Hurst, S.L., **Threshold Logic: An Engineering Survey**, Mills and Boon, London, 1971, translated Schwellwertlogik, UTB, Heidelberg, 1974.

15. Sheng, C.L., **Threshold Logic**, Academic Press, New York, 1969.

16. Chow, C.K., **'On the characterization of threshold functions'**, SCTLD, IEEE Special Publication No. S134, 34-38, Sep. 1961.

17. Winder, R.O., **'Threshold functions through n = 7'**, Scientific Report No. 7, Air Force Cambridge Research Lab., Contract AF19(604)-8423, 1969.

18. Edwards, C.R., **'The application of the Rademacher-Walsh transform to Boolean function classification and threshold-logic synthesis'**, Trans. IEEE, C.24, 48-62, Jan. 1975.

19. Karpovsky, M.G., **Finite Orthogonal series in the design of digital devices**, John Wiley, New York, 1976.

20. Edwards, C.R., **Private communications with Muzio, J.C.**

21. Muzio, J.C., Miller, D.M., **'Signatures, Threshold classes and Unate classes'**, Report to Tektronix Inc., 1981.

22. Muzio, J.C., **Composite spectra and the analysis of switching circuits**, Trans. on computers, IEEE, C-29, 750-753, 1980.

23. Miller, D.M., Muzio, J.C., **'Distribution of symmetry information in spectrum of Boolean function'** Electronic Letters, Vol. 15, No. 25, 816-817, Dec. 1979.

24. Edwards, C.R., Hurst, S.L., **'A digital synthesis procedure under function symmetries and mapping methods'**, Trans. on computers, IEEE, C-27, No. 11, 985-997, 1978.

25. Miller, D.M., Muzio, J.C., **'Detection of symmetries in totally specified or partially specified combinational functions'** Computers and Digital techniques, Vol. 2, No. 5, 203-209, 1979.

26. Muzio, J.C., Miller, D.M., Hurst, S.L., **'Multivariable symmetries and their detection'**, IEE Proceedings, Vol. 130, Pt. E, No. 5, 141-147, 1983.

27. Roth, J.P., **'Algebraic topological methods for the synthesis of switching systems I'**, Trans. Am. Math. Soc., 88, 301-326, 1959.

28. Miller, D.M., **Private communications with Muzio, J.C.**

29. Savir, J., **'Syndrome testable design of combinational circuits'**, Trans. IEEE, C 29, 442-451, 1980.

30. Eris, E., Miller, D.M., **'Syndrome testable internally unate combinational circuits'**, Elect. Lett. 19, 637-638, 1983.

31. Eris, E., Muzio, J.C., **'Syndrome and autocorrelation testable internally unate combinational circuits'**, Elect. Lett. (submitted).

32. Muzio, J.C., Miller, D.M., **'Spectral techniques for fault detection'**, Proc. 12 Int. Symp. on Fault-Tolerant Computing, 297-302, 1982.

33. Muzio, J.C., Miller, D.M., **'Spectral signatures for fault testing in combinational circuits'**, Report to Tektronix Inc., 1981.

Appendix A

4 Variable PN^2TD Classes Basis set coded in {1,-1} domain										
PN^2TD class	S_0	S_1	S_2	S_3	S_4	K	σ	σ_m	Number of solutions	Notes
1A	16	0	0	0	0	0	256	256	1	T,u
2A	14	2	2	2	2	0	212	212	1	T,u
3B	12	4	4	4	0	0	192	192	1	T,u
5B	8	8	8	0	0	0	192	192	1	T,u
4A	10	6	6	2	2	0	180	180	1	T,u
7A	6	6	6	6	6	0	180	180	1	T,u
6A	8	8	4	4	4	0	176	176	1	T,u
8A	4	4	4	4	4	4	80	64	28	

Note: T means threshold;
u means unate.

Appendix B

5 Variable PN^2TD Classes

Basis set coded in {1,-1} domain

PN^2TD class	S_0	S_1	S_2	S_3	S_4	S_5	K	σ	σ_m	Number of solutions	Notes
1	32	0	0	0	0	0	0	1024	1024	1	T,u
2	30	2	2	2	2	2	0	920	920	1	T,u
3	28	4	4	4	4	0	0	848	848	1	T,u
4	26	6	6	6	2	2	0	792	792	1	T,u
6	24	8	8	8	0	0	0	768	768	1	T,u
15	16	16	16	0	0	0	0	768	768	1	T,u
5	24	8	8	4	4	4	0	752	752	1	T,u
7	22	10	10	6	2	2	0	728	728	1	T,u
8	22	10	6	6	6	6	0	728	728	1	T,u
12	18	14	14	2	2	2	0	728	728	1	T,u
9	20	12	12	4	4	0	0	720	720	1	T,u
11	20	8	8	8	8	8	0	720	720	1	T,u
21	12	12	12	12	12	0	0	720	720	1	T,u
10	20	12	8	8	4	4	0	704	704	1	T,u
16	16	16	12	4	4	4	0	704	704	1	T,u
17	16	16	8	8	8	0	0	704	704	1	T,u
13	18	14	10	6	6	2	0	696	696	1	T,u
14	18	10	10	10	6	6	0	696	696	1	T,u
19	14	14	14	6	6	6	0	696	696	1	T,u
20	14	14	10	10	10	2	0	696	696	1	T,u
18	16	12	12	8	8	4	0	688	688	1	T,u
25	16	16	8	8	4	4	0	672	672	2	u
29	14	14	10	10	6	6	0	664	664	2	u
23	18	14	6	6	6	6	0	664	664	3	u
26	16	12	8	8	8	8	0	656	656	3	u
33	12	12	12	12	8	4	0	656	656	3	u
30	14	10	10	10	10	6	0	632	632	6	u
35	12	12	12	8	8	8	0	624	624	7(*1)	u
36	12	12	12	8	8	8	0	624	624	7(*1)	u
41	10	10	10	10	10	10	0	600	600	12	u

34	12	12	12	12	4	4	2	608	598	12	
22	20	12	4	4	4	4	4	608	588	28	
27	16	8	8	8	8	8	3	576	561	40	
31	14	10	10	10	6	6	3	568	553	42	
24	18	10	6	6	6	6	4	568	548	62	
38	12	12	8	8	8	8	3	544	529	72	
42	10	10	10	10	10	6	3	536	521	85	
28	16	8	8	8	8	4	4	528	508	133	
43	10	10	10	10	10	2	4	504	484	196	
32	14	10	10	6	6	6	4	504	484	208	
39	12	12	8	8	8	4	4	496	476	239	
37	12	12	12	4	4	4	5	480	455	364	
44	10	10	10	10	6	6	5	472	447	386	
40	12	8	8	8	8	8	5	464	439	457	
45	8	8	8	8	8	8	2	384	374	2210(*2)	
46	8	8	8	8	8	8	2	384	374	2210(*2)	
47	8	8	8	8	8	4	5	336	311	5305	
48	8	8	8	8	8	0	-	320	---	----	

Note: Basis set is identical for classes 35 and 36;
The 7 solutions (*1) are shared between these two classes. For classes 45 and 46, 2210 solutions (*2) are shared.

T means threshold;
u means unate.

Appendix C

Size information for 4 Variable PN^2T Classes	
PN^2T Class	Size of Class
1	2
2	30
3	32
4	480
5	1920
6	240
7	1680
8	1120
9	3360
10	13440
11	840
12	280
13	3360
14	13440
15	10080
16	5376
17	8960
18	896

Total number of functions = 65,536.

Size estimates for 5 Variable PN^2TD Classes

Sample size = 130,000 approximately.

PN^2TD Classes	Size	PN^2TD Classes	Size
1	R	21	0.02
2	R	22	0.05
3	R	23	0.4
4	0.01	24	0.66
5	0.05	25	0.45
6	<0.01	26	6.7
7	0.04	27	0.41
8	0.24	28	2.5
9	0.02	29	7.4
10	0.6	30	9.8
11	0.64	31,32	13.3
12	<0.01	33,35	2.7
13	0.57	34,37	0.77
14	3.3	36	10.0
15	R	38,39	15.7
16	0.07	40	3.9
17	0.03	41,42,43	10.5
18	4.9	44	2.5
19	0.5	45,46,48	0.28
20	0.62	47	0.32

Note: Size gives percentage of functions in the sample that belonged to a particular class.

R means rare.

Appendix D

4 variable PN^2T classes Function coded in {0,1} domain			
PN^2T Class	PN^2TD Class	r_0 r_1 r_2 r_3 r_4	Spectral Summary
1	1A	0 0 0 0 0	16 x 0
2	1B	8 8 0 0 0	2 x 8 14 x 0
3	2A	1 1 1 1 1	16 x 1
4	2B	7 7 1 1 1	2 x 7 14 x 1
5	3A	8 6 2 2 2	1 x 8 1 x 6 7 x 2 7 x 0
6	3B	2 2 2 2 0	8 x 2 8 x 0
7	3C	6 6 2 2 0	2 x 6 6 x 2 8 x 0
8	4A	3 3 3 1 1	4 x 3 12 x 1
9	4B	5 5 3 1 1	2 x 5 2 x 3 12 x 1
10	4C	7 5 3 3 1	1 x 7 1 x 5 3 x 3 11 x 1
11	5A	8 4 4 4 0	1 x 8 4 x 4 11 x 0
12	5B	4 4 4 0 0	4 x 4 12 x 0
13	6A	4 4 2 2 2	2 x 4 8 x 2 6 x 0
14	6B	6 4 4 2 2	1 x 6 2 x 4 7 x 2 6 x 0

15	6C	8 4 4 2 2	1 x 8 2 x 4 8 x 2 5 x 0
16	7A	5 3 3 3 3	1 x 5 5 x 3 10 x 1
17	7B	7 3 3 3 3	1 x 7 6 x 3 9 x 1
18	8A	6 2 2 2 2	1 x 6 15 x 2

Appendix E

5 variable PN^2T classes

Function coded in {0,1} domain

PN^2T Class	PN^2TD Class	r_0	r_1	r_2	r_3	r_4	r_5	Spectral Summary
1	1A	16	16	0	0	0	0	2 x 16 30 x 0
2	1B	0	0	0	0	0	0	32 x 0
3	2A	15	15	1	1	1	1	2 x 15 30 x 1
4	2B	1	1	1	1	1	1	32 x 1
5	3A	16	14	2	2	2	2	1 x 16 1 x 14 15 x 2 15 x 0
6	3B	14	14	2	2	2	0	2 x 14 14 x 2 16 x 0
7	3C	2	2	2	2	2	0	16 x 2 16 x 0
8	4A	15	13	3	3	3	1	1 x 15 1 x 13 7 x 3 23 x 1
9	4B	13	13	3	3	1	1	2 x 13 6 x 3 24 x 1
10	4C	3	3	3	3	1	1	8 x 3 24 x 1
11	5A	14	12	4	4	2	2	1 x 14 1 x 12 3 x 4 15 x 2 12 x 0
12	5B	16	12	4	4	2	2	1 x 16 1 x 12 3 x 4 16 x 2 11 x 0
13	5C	12	12	4	2	2	2	2 x 12 2 x 4 16 x 2 12 x 0
14	5D	4	4	4	2	2	2	4 x 4 16 x 2 12 x 0

15	6A	16	12	4	4	4	0	1 x 16	1 x 12	7 x 4	23 x 0					
16	6B	12	12	4	4	0	0	2 x 12	6 x 4	24 x 0						
17	6C	4	4	4	4	0	0	8 x 4	24 x 0							
18	7A	15	11	5	5	3	1	1 x 15	1 x 11	3 x 5	4 x 3	23 x 1				
19	7B	13	11	5	5	1	1	1 x 13	1 x 11	3 x 5	3 x 3	24 x 1				
20	7C	11	11	5	3	1	1	2 x 11	2 x 5	4 x 3	24 x 1					
21	7D	5	5	5	3	1	1	4 x 5	4 x 3	24 x 1						
22	8A	13	11	5	3	3	3	1 x 13	1 x 11	1 x 5	9 x 3	20 x 1				
23	8B	15	11	5	3	3	3	1 x 15	1 x 11	1 x 5	10 x 3	19 x 1				
24	8C	11	11	3	3	3	3	2 x 11	10 x 3	20 x 1						
25	8D	5	5	3	3	3	3	2 x 5	10 x 3	20 x 1						
26	9A	16	10	6	6	2	2	1 x 16	1 x 10	3 x 6	12 x 2	15 x 0				
27	9B	14	10	6	6	2	0	1 x 14	1 x 10	3 x 6	11 x 2	16 x 0				
28	9C	10	10	6	2	2	0	2 x 10	2 x 6	12 x 2	16 x 0					
29	9D	6	6	6	2	2	0	4 x 6	12 x 2	16 x 0						
30	10A	14	10	6	4	4	2	1 x 14	1 x 10	1 x 6	4 x 4	13 x 2	12 x 0			
31	10B	14	10	6	4	2	2	1 x 14	1 x 10	1 x 6	4 x 4	13 x 2	12 x 0			
32	10C	16	10	6	4	4	2	1 x 16	1 x 10	1 x 6	4 x 4	14 x 2	11 x 0			
33	10D	12	10	6	4	2	2	1 x 12	1 x 10	1 x 6	3 x 4	14 x 2	12 x 0			

34	10E	10	10	4	4	2	2	2	x	10	4	x	4	14	x	2	12	x	0
35	10F	6	6	4	4	2	2	2	x	6	4	x	4	14	x	2	12	x	0
36	11A	12	10	4	4	4	4	1	x	12	1	x	10	5	x	4	15	x	2
								10	x	0									
37	11B	14	10	4	4	4	4	1	x	14	1	x	10	6	x	4	14	x	2
								10	x	0									
38	11C	16	10	4	4	4	4	1	x	16	1	x	10	6	x	4	15	x	2
								9	x	0									
39	11D	6	4	4	4	4	4	1	x	6	6	x	4	15	x	2	10	x	0
40	12A	15	9	7	7	1	1	1	x	15	1	x	9	3	x	7	27	x	1
41	12B	9	9	7	1	1	1	2	x	9	2	x	7	28	x	1			
42	12C	7	7	7	1	1	1	4	x	7	28	x	1						
43	13A	15	9	7	5	3	3	1	x	15	1	x	9	1	x	7	2	x	5
								6	x	3	21	x	1						
44	13B	13	9	7	5	3	1	1	x	13	1	x	9	1	x	7	2	x	5
								5	x	3	22	x	1						
45	13C	15	9	7	5	3	1	1	x	15	1	x	9	1	x	7	2	x	5
								6	x	3	21	x	1						
46	13D	11	9	7	3	3	1	1	x	11	1	x	9	1	x	7	1	x	5
								6	x	3	22	x	1						
47	13E	9	9	5	3	3	1	2	x	9	2	x	5	6	x	3	22	x	1
48	13F	7	7	5	3	3	1	2	x	7	2	x	5	6	x	3	22	x	1
49	14A	13	9	5	5	5	3	1	x	13	1	x	9	3	x	5	8	x	3
								19	x	1									
50	14B	15	9	5	5	3	3	1	x	15	1	x	9	3	x	5	9	x	3
								18	x	1									
51	14C	15	9	5	5	5	3	1	x	15	1	x	9	3	x	5	9	x	3
								18	x	1									
52	14D	13	9	5	5	3	3	1	x	13	1	x	9	3	x	5	8	x	3
								19	x	1									
53	14E	11	9	5	5	3	3	1	x	11	1	x	9	2	x	5	9	x	3

								19	x	1									
54	14F	7	5	5	5	3	3	1	x	7	3	x	5	9	x	3	19	x	1
55	15A	16	8	8	8	0	0	1	x	16	4	x	8	27	x	0			
56	15B	8	8	8	0	0	0	4	x	8	28	x	0						
57	16A	16	8	8	6	2	2	1	x	16	2	x	8	2	x	6	14	x	2
								13	x	0									
58	16B	14	8	8	6	2	2	1	x	14	2	x	8	2	x	6	13	x	2
								14	x	0									
59	16C	10	8	8	2	2	2	1	x	10	2	x	8	1	x	6	14	x	2
								14	x	0									
60	16D	8	8	6	2	2	2	2	x	8	2	x	6	14	x	2	14	x	0
61	17A	16	8	8	4	4	4	1	x	16	2	x	8	8	x	4	21	x	0
62	17B	12	8	8	4	4	0	1	x	12	2	x	8	7	x	4	22	x	0
63	17C	16	8	8	4	4	0	1	x	16	2	x	8	8	x	4	21	x	0
64	17D	8	8	4	4	4	0	2	x	8	8	x	4	22	x	0			
65	18A	14	8	6	6	4	4	1	x	14	1	x	8	2	x	6	4	x	4
								13	x	2	11	x	0						
66	18B	12	8	6	6	4	2	1	x	12	1	x	8	2	x	6	3	x	4
								14	x	2	11	x	0						
67	18C	16	8	6	6	4	2	1	x	16	1	x	8	2	x	6	4	x	4
								14	x	2	10	x	0						
68	18D	12	8	6	4	4	2	1	x	12	1	x	8	2	x	6	3	x	4
								14	x	2	11	x	0						
69	18E	16	8	6	6	4	4	1	x	16	1	x	8	2	x	6	4	x	4
								14	x	2	10	x	0						
70	18F	14	8	6	6	4	2	1	x	14	1	x	8	2	x	6	4	x	4
								13	x	2	11	x	0						
71	18G	10	8	6	4	4	2	1	x	10	1	x	8	1	x	6	4	x	4
								14	x	2	11	x	0						
72	18H	8	6	6	4	4	2	1	x	8	2	x	6	4	x	4	14	x	2
								11	x	0									

73	19A	15	7	7	7	3	3	1	x	15	3	x	7	1	x	5	7	x	3
								20	x	1									
74	19B	13	7	7	7	3	3	1	x	13	3	x	7	1	x	5	6	x	3
								21	x	1									
75	19C	11	7	7	3	3	3	1	x	11	3	x	7	7	x	3	21	x	1
76	19D	9	7	7	3	3	3	1	x	9	2	x	7	1	x	5	7	x	3
								21	x	1									
77	20A	15	7	7	5	5	5	1	x	15	2	x	7	4	x	5	4	x	3
								21	x	1									
78	20B	11	7	7	5	5	1	1	x	11	2	x	7	3	x	5	4	x	3
								22	x	1									
79	20C	15	7	7	5	5	1	1	x	15	2	x	7	4	x	5	4	x	3
								21	x	1									
80	20D	13	7	7	5	5	1	1	x	13	2	x	7	4	x	5	3	x	3
								22	x	1									
81	20E	9	7	5	5	5	1	1	x	9	1	x	7	4	x	5	4	x	3
								22	x	1									
82	21A	16	6	6	6	6	6	1	x	16	6	x	6	10	x	2	15	x	0
83	21B	10	6	6	6	6	0	1	x	10	5	x	6	10	x	2	16	x	0
84	21C	14	6	6	6	6	0	1	x	14	6	x	6	9	x	2	16	x	0
85	22A	14	10	6	2	2	2	1	x	14	1	x	10	1	x	6	29	x	2
86	22B	10	10	2	2	2	2	2	x	10	30	x	2						
87	22C	6	6	2	2	2	2	2	x	6	30	x	2						
88	23A	15	9	7	3	3	3	1	x	15	1	x	9	1	x	7	12	x	3
								17	x	1									
89	23B	13	9	7	3	3	3	1	x	13	1	x	9	1	x	7	11	x	3
								18	x	1									
90	23C	9	9	3	3	3	3	2	x	9	12	x	3	18	x	1			
91	23D	7	7	3	3	3	3	2	x	7	12	x	3	18	x	1			

92	24A	13	9	5	3	3	3	1 x 13 15 x 1	1 x 9	1 x 5	14 x 3
93	24B	15	9	5	3	3	3	1 x 15 14 x 1	1 x 9	1 x 5	15 x 3
94	24C	11	9	3	3	3	3	1 x 11	1 x 9	15 x 3	15 x 1
95	24D	7	5	3	3	3	3	1 x 7	1 x 5	15 x 3	15 x 1
96	25A	16	8	8	4	4	2	1 x 16 9 x 0	2 x 8	4 x 4	16 x 2
97	25B	14	8	8	4	4	2	1 x 14 10 x 0	2 x 8	4 x 4	15 x 2
98	25C	16	8	8	4	2	2	1 x 16 9 x 0	2 x 8	4 x 4	16 x 2
99	25D	12	8	8	4	2	2	1 x 12 10 x 0	2 x 8	3 x 4	16 x 2
100	25E	8	8	4	4	2	2	2 x 8	4 x 4	16 x 2	10 x 0
101	26A	14	8	6	4	4	4	1 x 14 14 x 2	1 x 8 9 x 0	1 x 6	6 x 4
102	26B	10	8	4	4	4	4	1 x 10 9 x 0	1 x 8	6 x 4	15 x 2
103	26C	16	8	6	4	4	4	1 x 16 15 x 2	1 x 8 8 x 0	1 x 6	6 x 4
104	26D	12	8	6	4	4	4	1 x 12 15 x 2	1 x 8 9 x 0	1 x 6	5 x 4
105	26E	8	6	4	4	4	4	1 x 8 9 x 0	1 x 6	6 x 4	15 x 2
106	27A	12	8	4	4	4	4	1 x 12	1 x 8	11 x 4	19 x 0
107	27B	16	8	4	4	4	4	1 x 16	1 x 8	12 x 4	18 x 0
108	27C	8	4	4	4	4	4	1 x 8	12 x 4	19 x 0	
109	28A	14	8	4	4	4	4	1 x 14 7 x 0	1 x 8	8 x 4	15 x 2
110	28B	12	8	4	4	4	2	1 x 12	1 x 8	7 x 4	16 x 2

								7	x	0									
111	28C	16	8	4	4	4	2	1	x	16	1	x	8	8	x	4	16	x	2
								6	x	0									
112	28D	8	4	4	4	4	2	1	x	8	8	x	4	16	x	2	7	x	0
113	29A	15	7	7	5	5	3	1	x	15	2	x	7	2	x	5	10	x	3
								17	x	1									
114	29B	13	7	7	5	5	3	1	x	13	2	x	7	2	x	5	9	x	3
								18	x	1									
115	29C	15	7	7	5	3	3	1	x	15	2	x	7	2	x	5	10	x	3
								17	x	1									
116	29D	13	7	7	5	3	3	1	x	13	2	x	7	2	x	5	9	x	3
								18	x	1									
117	29E	9	7	5	5	3	3	1	x	9	1	x	7	2	x	5	10	x	3
								18	x	1									
118	29F	11	7	7	5	3	3	1	x	11	2	x	7	1	x	5	10	x	3
								18	x	1									
119	30A	15	7	5	5	5	5	1	x	15	1	x	7	5	x	5	7	x	3
								18	x	1									
120	30B	13	7	5	5	5	3	1	x	13	1	x	7	5	x	5	6	x	3
								19	x	1									
121	30C	15	7	5	5	5	3	1	x	15	1	x	7	5	x	5	7	x	3
								18	x	1									
122	30D	11	7	5	5	5	3	1	x	11	1	x	7	4	x	5	7	x	3
								19	x	1									
123	30E	13	7	5	5	5	5	1	x	13	1	x	7	5	x	5	6	x	3
								19	x	1									
124	30F	9	5	5	5	5	3	1	x	9	5	x	5	7	x	3	19	x	1
125	31A	13	7	5	5	5	3	1	x	13	1	x	7	3	x	5	12	x	3
								15	x	1									
126	31B	15	7	5	5	5	3	1	x	15	1	x	7	3	x	5	13	x	3
								14	x	1									
127	31C	13	7	5	5	3	3	1	x	13	1	x	7	3	x	5	12	x	3
								15	x	1									

128	31D	15	7	5	5	3	3	1 x 15 14 x 1	1 x 7	3 x 5	13 x 3	
129	31E	11	7	5	5	3	3	1 x 11 15 x 1	1 x 7	2 x 5	13 x 3	
130	31F	9	5	5	5	3	3	1 x 9	3 x 5	13 x 3	15 x 1	
131	32A	15	7	5	5	3	3	1 x 15 14 x 1	1 x 7	3 x 5	13 x 3	
132	32B	13	7	5	5	3	3	1 x 13 15 x 1	1 x 7	3 x 5	12 x 3	
133	32C	11	7	5	3	3	3	1 x 11 15 x 1	1 x 7	2 x 5	13 x 3	
134	32D	9	5	5	3	3	3	1 x 9	3 x 5	13 x 3	15 x 1	
135	33A	16	6	6	6	6	4	1 x 16 11 x 0	4 x 6	4 x 4	12 x 2	
136	33B	12	6	6	6	6	2	1 x 12 12 x 0	4 x 6	3 x 4	12 x 2	
137	33C	14	6	6	6	6	4	1 x 14 12 x 0	4 x 6	4 x 4	11 x 2	
138	33D	14	6	6	6	4	2	1 x 14 12 x 0	4 x 6	4 x 4	11 x 2	
139	33E	16	6	6	6	6	2	1 x 16 11 x 0	4 x 6	4 x 4	12 x 2	
140	33F	10	6	6	6	4	2	1 x 10 12 x 0	3 x 6	4 x 4	12 x 2	
141	34A	14	6	6	6	6	2	1 x 14	4 x 6	27 x 2		
142	34B	14	6	6	6	2	2	1 x 14	4 x 6	27 x 2		
143	34C	10	6	6	6	2	2	1 x 10	3 x 6	28 x 2		
144	35A	16	6	6	6	4	4	1 x 16 11 x 0	4 x 6	4 x 4	12 x 2	
145	35B	14	6	6	6	4	4	1 x 14 12 x 0	4 x 6	4 x 4	11 x 2	
146	35C	10	6	6	4	4	4	1 x 10	3 x 6	4 x 4	12 x 2	

								12 x 0						
147	35D	12	6	6	6	4	4	1 x 12 12 x 0	4 x 6	3 x 4	12 x 2			
148	36A	16	6	6	6	4	4	1 x 16 9 x 0	3 x 6	6 x 4	13 x 2			
149	36B	14	6	6	6	4	4	1 x 14 10 x 0	3 x 6	6 x 4	12 x 2			
150	36C	12	6	6	6	4	4	1 x 12 10 x 0	3 x 6	5 x 4	13 x 2			
151	36D	14	6	6	4	4	4	1 x 14 10 x 0	3 x 6	6 x 4	12 x 2			
152	36E	10	6	6	4	4	4	1 x 10 10 x 0	2 x 6	6 x 4	13 x 2			
153	37A	14	6	6	6	2	2	1 x 14	4 x 6	27 x 2				
154	37B	10	6	6	2	2	2	1 x 10	3 x 6	28 x 2				
155	38A	14	6	6	4	4	4	1 x 14 8 x 0	2 x 6	8 x 4	13 x 2			
156	38B	16	6	6	4	4	4	1 x 16 7 x 0	2 x 6	8 x 4	14 x 2			
157	38C	12	6	6	4	4	4	1 x 12 8 x 0	2 x 6	7 x 4	14 x 2			
158	38D	10	6	4	4	4	4	1 x 10 8 x 0	1 x 6	8 x 4	14 x 2			
159	39A	16	6	6	4	4	4	1 x 16 7 x 0	2 x 6	8 x 4	14 x 2			
160	39B	12	6	6	4	4	2	1 x 12 8 x 0	2 x 6	7 x 4	14 x 2			
161	39C	14	6	6	4	4	4	1 x 14 8 x 0	2 x 6	8 x 4	13 x 2			
162	39D	14	6	6	4	4	2	1 x 14 8 x 0	2 x 6	8 x 4	13 x 2			
163	39E	10	6	4	4	4	2	1 x 10 8 x 0	1 x 6	8 x 4	14 x 2			

164	40A	14	6	4	4	4	4	1	x	14	1	x	6	10	x	4	14	x	2
								6	x	0									
165	40B	12	6	4	4	4	4	1	x	12	1	x	6	9	x	4	15	x	2
								6	x	0									
166	40C	16	6	4	4	4	4	1	x	16	1	x	6	10	x	4	15	x	2
								5	x	0									
167	40D	10	4	4	4	4	4	1	x	10	10	x	4	15	x	2	6	x	0
168	41A	15	5	5	5	5	5	1	x	15	6	x	5	10	x	3	15	x	1
169	41B	13	5	5	5	5	5	1	x	13	6	x	5	9	x	3	16	x	1
170	41C	11	5	5	5	5	5	1	x	11	5	x	5	10	x	3	16	x	1
171	42A	15	5	5	5	5	5	1	x	15	6	x	5	10	x	3	15	x	1
172	42B	11	5	5	5	5	3	1	x	11	5	x	5	10	x	3	16	x	1
173	42C	13	5	5	5	5	5	1	x	13	6	x	5	9	x	3	16	x	1
174	42D	13	5	5	5	5	3	1	x	13	6	x	5	9	x	3	16	x	1
175	42E	15	5	5	5	5	3	1	x	15	6	x	5	10	x	3	15	x	1
176	43A	15	5	5	5	5	5	1	x	15	6	x	5	10	x	3	15	x	1
177	43B	11	5	5	5	5	1	1	x	11	5	x	5	10	x	3	16	x	1
178	43C	13	5	5	5	5	1	1	x	13	6	x	5	9	x	3	16	x	1
179	44A	13	5	5	5	5	3	1	x	13	4	x	5	15	x	3	12	x	1
180	44B	13	5	5	5	3	3	1	x	13	4	x	5	15	x	3	12	x	1
181	44C	15	5	5	5	5	3	1	x	15	4	x	5	16	x	3	11	x	1
182	44D	11	5	5	5	3	3	1	x	11	3	x	5	16	x	3	12	x	1
183	45A	16	4	4	4	4	4	1	x	16	16	x	4	15	x	0			
184	45B	12	4	4	4	4	4	1	x	12	15	x	4	16	x	0			
185	46A	14	4	4	4	4	4	1	x	14	12	x	4	15	x	2	4	x	0

186	46B	12	4	4	4	4	2	1	x	12	11	x	4	16	x	2	4	x	0
187	46C	16	4	4	4	4	2	1	x	16	12	x	4	16	x	2	3	x	0
188	47A	16	4	4	4	4	4	1	x	16	16	x	4	15	x	0			
189	47B	12	4	4	4	4	0	1	x	12	15	x	4	16	x	0			
190	48A	12	4	4	4	4	4	1	x	12	15	x	4	16	x	0			
191	48B	16	4	4	4	4	4	1	x	16	16	x	4	15	x	0			

Appendix F

SYMMETRY TYPES

Here the different two variable symmetry types, as defined by Edwards et al[24] are briefly presented. The symmetries associated with a Hamming distance of two are the nonequivalence symmetry, the equivalence symmetry and the multiform symmetry, whilst the single variable symmetry involves a Hamming distance of one.

Nonequivalence Symmetry

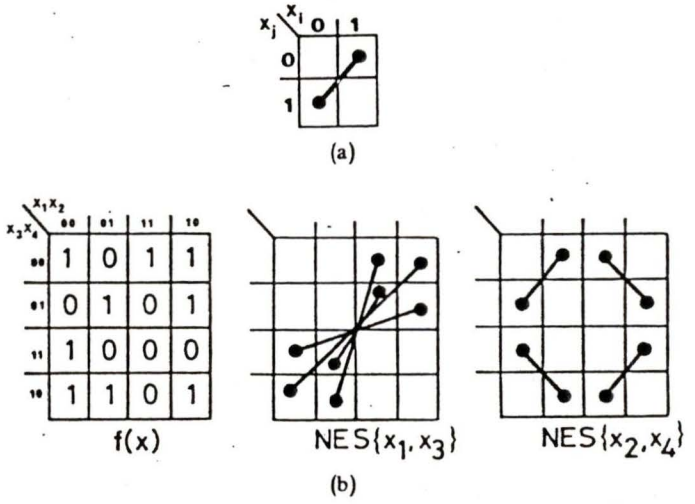


Fig. 1 : Non-equivalence Symmetry:

(a) General description of NES $\{x_i, x_j\}$, where the bar symbol links like-valued minterm areas.

(b) Karnaugh-map plot of function f(x) which is NES $\{x_1, x_3\}$ and also NES $\{x_2, x_4\}$.

A function $f(x_1,\ldots,x_i,x_j,\ldots,x_n)$ exhibits nonequivalence symmetry in its input variables x_i, x_j if

$$f(x_1,\ldots,0,1,\ldots,x_n) = f(x_1,\ldots,1,0,\ldots,x_n) \qquad \ldots (1)$$

that is the function output f(x) remains invariant under the interchange $\{x_i = 0, x_j = 1\}$ and $\{x_i = 1, x_j = 0\}$. The symmetry is termed 'nonequivalence' because a function exhibiting this property is identical in the n-spaces where x_i is not equal to x_j. We will refer to this type of symmetry in x_i, x_j by NES $\{x_i, x_j\}$.

Fig. 1(a) illustrates the n-spaces which by definition are identical under NES $\{x_i, x_j\}$, while Fig. 1(b) covers a four variable example which is nonequivalence symmetric in both $\{x_1, x_3\}$ and in $\{x_2, x_4\}$. As we are dealing with pairs of input variables, it will be clear that there are three possible non-equivalence symmetries in 3-variable functions, six in 4-variable functions, and so on.

Equivalence symmetry

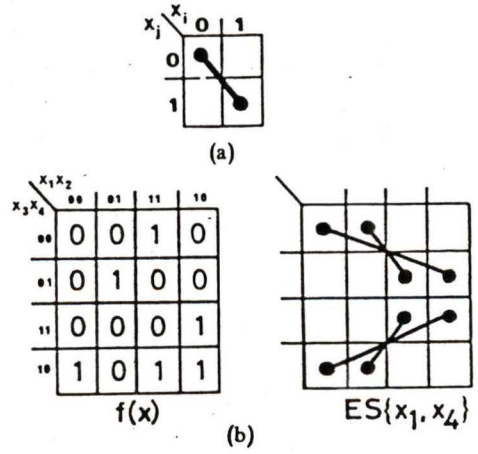


Fig. 2 : Equivalence Symmetry :

(a) General definition of ES $\{x_i, x_j\}$.

(b) Function f(x) with ES $\{x_1, x_4\}$.

A function $f(x_1,\ldots,x_i,x_j,\ldots,x_n)$ exhibits equivalence symmetry in its input variables x_i, x_j if

$$f(x_1,\ldots,0,0,\ldots,x_n) = f(x_1,\ldots,1,1,\ldots,x_n) \qquad \ldots (2)$$

that is the function output f(x) remains invariant under the interchange $\{x_i = 0, x_j = 0\}$ and $\{x_i = 1, x_j = 1\}$. This symmetry is termed 'equivalence' because a function exhibiting this property is identical in the n-spaces where $x_i = x_j$. We will refer to this type of symmetry in x_i, x_j by ES $\{x_i, x_j\}$.

Fig. 2(a) illustrates the n-spaces which by definition are identical under ES $\{x_i, x_j\}$, with Fig. 2(b) giving a 4-variable example.

Multiform Symmetry

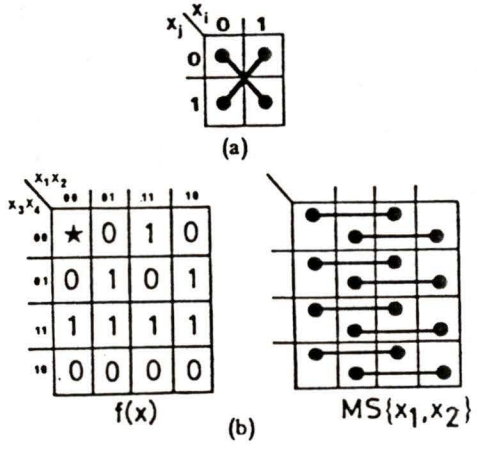


Fig. 3 : Multiform Symmetry :

(a) General definition of MS $\{x_i, x_j\}$.

(b) Function f(x) with MS $\{x_1, x_2\}$, don't care minterm * allocated logic value 1.

A function $f(x_1,\ldots,x_i,x_j,\ldots,x_n)$ exhibits multiform symmetry in its input variables x_i, x_j, if the two relationships (1) and (2) above simultaneously hold, that is,

$$f(x_1,\ldots,0,1,\ldots,x_n) = f(x_1,\ldots,1,0,\ldots,x_n)$$

and

$$f(x_1,\ldots,0,0,\ldots,x_n) = f(x_1,\ldots,1,1,\ldots,x_n) \qquad \ldots (3)$$

We will refer to this type of symmetry in x_i, x_j by MS $\{x_i, x_j\}$. Fig. 3 illustrates this relationship in general and specific form.

Single-variable Symmetry

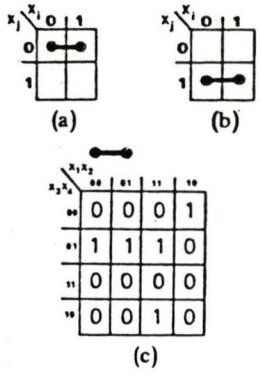


Fig. 4 : Single variable Symmetries :

(a) General definition of {SVS x_i} $\bar{x}_j$}.

(b) General definition of {SVS x_i} x_j}.

(c) Function f(x) with {SVS x_2}$\bar{x}_1$.

A function $f(x_1,\ldots,x_i,x_j,\ldots,x_n)$ exhibits a single variable symmetry in its input variable x_i if

$$f(x_1,\ldots,0,0,\ldots,x_n) = f(x_1,\ldots,1,0,\ldots,x_n) \quad \ldots (4)$$

or if

$$f(x_1,\ldots,0,1,\ldots,x_n) = f(x_1,\ldots,1,1,\ldots,x_n) \quad \ldots (5)$$

that is, the function output remains invariant under the interchange $\{x_i = 0, x_j = 0\}$ and $\{x_i = 1, x_j = 0\}$, or under the interchange $\{x_i = 0, x_j = 1\}$ and $\{x_i = 1, x_j = 1\}$. In the former case we have a single-variable symmetry in x_i in the n-space $x_j = 0$ of a function, which we will refer to as $\{\text{SVS } x_i\}\bar{x}_j$, while in the latter case we have a single-variable symmetry in x_i in the n-space $x_j = 1$. We will refer to the latter as $\{\text{SVS } x_i\}x_j$.

Fig. 4 illustrates these two possible forms of symmetry in x_i.

Appendix G

DETECTION OF SYMMETRIES

A method of detecting symmetries[25], applicable to both totally and partially specified functions is briefly reviewed in this section. The two variable symmetries in variables x_i and x_j mentioned earlier, involve minterm areas 00, 01, 11 and 10 for the respective variables concerned. Hence, a function f(x) will be represented by four columns, where each column represents a quarter of the function f(x). If we were to widen our search to look for n variable symmetries, the function f(x) would have to be divided into 2^n columns, each column containing one minterm[26]. However our search is limited to two variable symmetries.

A function f(x) is said to be decomposable if a pair of columns have identical entries in every row. When a function is partially specified, the columns corresponding to a symmetry may contain 'don't-care' conditions. The columns are not required to be identical. Rather, there must be an assignment to the don't-care conditions for which the columns become identical. An equivalent, but more-easily verified, condition requires that the columns contain no conflicting entries, i.e. there is no row for which one column has a 0 and the other has a 1. For columns with don't-cares, this test can be performed and columns satisfying this condition are said to be compatible. A formal definition for compatibility is given below.

Definition :

Two assignments to the variables x_i and x_j, $1 \le i, j \le n$, denoted $\langle \alpha_i, \alpha_j \rangle$ and $\langle \beta_i, \beta_j \rangle$, are compatible if, and only if,

$$f(x_1,..,\alpha_i,..,\alpha_j,..,x_n) = f(x_1,..,\beta_i,..,\beta_j,..,x_n) \quad .. (1)$$

for the entire space over which f(x) is defined. Otherwise $\langle \alpha_i, \alpha_j \rangle$ and $\langle \beta_i, \beta_j \rangle$ are incompatible. Compatibility is written as $\langle \alpha_i, \alpha_j \rangle \sim \langle \beta_i, \beta_j \rangle$ incompatibility is written as $\langle \alpha_i, \alpha_j \rangle \not\sim \langle \beta_i, \beta_j \rangle$.

We can enumerate the symmetries exhibited by a function by identifying the compatible input assignments for each pair of variables. This can be done efficiently by using the cube notation introduced by Roth[27]. A vertex is an n-tuple of 0's and 1's representing an assignment to all the variables $x_1, x_2, \ldots, x_n$. A cube is an n-tuple of 0's, 1's and -'s representing a product term where -'s are variables that do not appear in the product term. A cube represents a set of vertices found by replacing the -'s in the cube by 0's and 1's in all possible ways.

A function is specified by two arrays of cubes termed the 'on' and the 'off' arrays. The on-array is a set of cubes such that each assignment for which f(x) = 1 is included in at least one cube, and each cube only represents assignments for which f(x) = 1. The off-array is similarly defined with f(x) = 0. Assignments for which f(x) is a don't-care are omitted from both arrays.

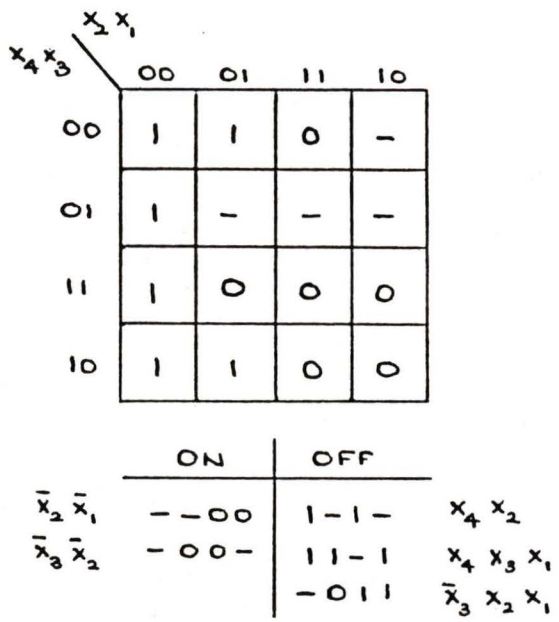


Fig. 1 : Example of cube notation.

As an example of this cube notation consider the function in Fig. 1. Each of the blocks of 0's or 1's on the map corresponds to a cube in the off or on array, respectively. For example, the leftmost column is represented by --00 . The bottom right block of four 0's is represented by 1-1-. These cubes in the 'on' array correspond to the product terms in a sum of products expression for the example function in Fig. 1. Cube --00 represents the product term $\bar{x}_2\bar{x}_1$ and cube 1-1- represents the product term x_4x_2. Clearly the arrays representing a function are not unique. We shall use A and B to denote the cubes. The symbols in these cubes are identified as a_i, b_i; a_i, $b_i \in \{0, 1, -\}$, $1<=i<=n$. The pair of symbols $<a_i, a_j>$ form a subcube of the cube A. This subcube represents a set of assignments to the variables x_i, x_j, with a_i representing the values taken by x_i and a_j representing the values taken by x_j.

$\wedge$	0	1	-
0	0	ϕ	0
1	ϕ	1	1
-	0	1	-

AND

$\vee$	ϕ	0	1	-
ϕ	ϕ	0	1	-
0	0	0	-	-
1	1	-	1	-
-	-	-	-	-

OR

*	0	1	-
0	ϕ	0	0
1	1	ϕ	1
-	1	0	-

*

Fig. 2 : Cube Operators.

The identification of compatibilities, in the form of two new sets of arrays, S and T, called compatibility tables, involves comparing each cube from the on-array with each cube from the off-array. Three simple operations over cubes are required in creating the compatibility tables. The cubes in the on and off arrays, and the S_i and T_i, $1<=i<=n$, are represented by bit strings. Two bits are required for each symbol. If ϕ, 0, 1, - are represented by 00, 01, 10, and 11, respectively, then the $\wedge$ operator is the logical AND of the bit strings representing the two cubes, and the $\vee$ operator is the logical OR. A logical definition of all the three operators are given in tabular form in Fig. 2 (a-c). The * operator is a little more complex and its implementation is given below.

The operation $B_1 * B_2$ is implemented as follows:

$OR(AND(M_1,AND(B_2,SHL(B_1))),AND(M_2,AND(B_2,SHR(B_1)))$ where

B_1, B_2 denote two bit strings representing cubes where the two two bits representing a symbol are adjacent, i.e. 011- is represented by 01101011.

M_1 is a bit string 1010....10 with the same number of bits as B_1 and B_2.

M_2 is a bit string 0101....01 with the same number of bits as B_1 and B_2.

The algorithm for constructing S and T using the above mentioned operations is given below. $A^1,A^2,\ldots, A^p$ denote the cubes of the on array. $B^1,B^2,\ldots, B^q$ denote the cubes of the off array.

Initialize S_i, T_i the rows of S and T, $1<=i<=n$, to n-tuples of ϕ's; Let S_{ij} and T_{ij} denote the j^{th} entry of S_i and T_i respectively for each pair i,j, $1<=i<=p$, $1<=j<=q$, and let d represent the distance between A^i and B^j. There are four possibilities:

(1) d = 0: error in the function specification in that a vertex appears in both the on and off arrays;

(2) d = 1: let k be the position where one of A^i, B^j is 1 and the other is 0.

$S_k = S_k \vee (A^i \wedge B^j)$.

if $A^i_k = 1$ then $T_k = T_k \vee (A^i * B^j)$

else $T_k = T_k \vee (B^j * A^i)$.

(3) d = 2: k,m denote the two positions where one of A^i, B^j is 1 and the other is 0.

if $A^i_k = 1$ then $T_{km} = T_{km} \vee (A^i_m * B^j_m)$

else $T_{km} = T_{km} \vee (B^j_m * A^i_m)$

(4) d => 3: ignore this pair.

We are interested in symmetries involving Hamming distances of only 1 and 2, hence, we are ignoring d greater than 2. The six possible symmetries for x_i, x_j, the corresponding compatibilities and the required tests are given in Table 1. All portions of a test must be true for the given compatibility and the corresponding symmetry to exist. For a further detailed discussion of the method, readers should consult Miller et al[25].

Symmetry	Compatibility	Test
ES $\{x_i, x_j\}$	00 $\backsim$ 11	$T_{ij} \neq 1$, $T_{ij} \neq -$, $T_{ji} \neq 1$, $T_{ji} \neq -$;
NES $\{x_i, x_j\}$	01 $\backsim$ 10	$T_{ij} \neq 0$, $T_{ij} \neq -$, $T_{ji} \neq 0$, $T_{ji} \neq -$;
$\{SVS\ x_i\}\bar{x}_j$	00 $\backsim$ 10	$S_{ij} \neq 0$, $S_{ij} \neq -$;
$\{SVS\ x_i\}x_j$	01 $\backsim$ 11	$S_{ij} \neq 1$, $S_{ij} \neq -$;
$\{SVS\ x_j\}\bar{x}_i$	00 $\backsim$ 01	$S_{ji} \neq 0$, $S_{ij} \neq -$;
$\{SVS\ x_j\}x_i$	10 $\backsim$ 11	$S_{ji} \neq 1$, $S_{ij} \neq -$.

Table 1: S_i and T_i tests for symmetry detection.

VITA

Surname: IYER Given Names: Balasubramanian Hariharan

Place of Birth: Coimbatore, India Date of Birth March 10, 1957

Educational Institutions Attended, with Dates of Entering and Leaving

Indian Institute of Technology, Delhi	1974	to	1979
The University of Manitoba, Winnipeg	1980	to	1982
University of Victoria	1982	to	1984

Degrees, Diplomas, Etc., Awarded, with Dates and Names of Institutions:

B. Tech (Honors)	1979	Indian Institute of Technology, Delhi

Honors and Awards:

University of Victoria Graduate Supplement

Publications:

none

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis or dissertation (the title of which is shown below) to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/~~Dissertation~~

CLASSIFICATION AND DIGITAL SYNTHESIS USING SPECTRAL TECHNIQUES.

Author [redacted]
(Signature)

BALASUBRAMANIAN. H. IYER
Name (in block letters)

24th Aug. 1984
(Date)