THE PIECEWISE LOGICAL WALSH-HADAMARD TRANSFORM AND APPLICATIONS OF H DIAGRAMS

by

Yung-Leung Henry MAR

Submitted to the Department of Electrical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Electrical Engineering
Faculty of Science and Engineering
The University of Ottawa,
Ottawa, Canada.

May 1973

ABSTRACT

The main object of this thesis is the logical Walsh-Hadamard transform and the applications of H diagrams for Walsh functions and Hadamard matrices. A brief survey on the principles and applications of the orthogonal transformation is outlined. The logical Walsh-Hadamard transform is introduced and defined. A combinatorial network for obtaining this transform is worked out for N = 2, 4 and 8. The logical transform is also extended to cover the case of the length N of the binary data string becomes greater than 8. This is achieved either by increasing the dimensionality or by employing the piecewise concept.

The H diagram is first applied for realizing the logical Walsh-Hadamard transform by switching functions. It is also used for factoring the Hadamard matrix, thereby providing a new approach for the fast Walsh-Hadamard transform. Finally, a spectral analysis is performed on the H diagram.

ACKNOWLEDGEMENTS

The author is most grateful to his supervisor, Professor C. L. Sheng for his encouragement, understanding and guidance throughout the course of the research.

Gratitude is also expressed to Professor M. Krieger, Professor W Steenaart and Professor S. G. S. Shiva for stimulating discussions and critical comment.

Special thanks go to the National Research Council of Canada, and the Department of University Affairs of the Province of Ontario for their financial supports.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Walsh-Hadamard Transforms In Image Processing[1]

Since detailed comprehension of the physical world depends critically upon vision and a picture is worth better than 1000 words, it is not surprising that investigators from many different fields are currently exploring image processing. Applications include radiography, cartography, earth resources data collected by satellites, spectrum analyzers, speech analyzers, television pictures, facsimile, remotely piloted vehicles, and others. Image processing is becoming of increasing importance for several reasons. Among them there are firstly the advent of the laser, which makes Fourier processing in the optical domain a practical reality[2], and secondly the invention of the Fast Fourier Transform and other fast algorithms[3,4], which makes digital image processing being suitably handled by computer and thus a practical reality. Nevertheless, the objectives of image processing are typically data compression and image enhancement.

An image, or a picture, can be defined as a monochrome continuous non-negative function f(x, y) defined over a finite region of two-dimensional space and in which the value of the function at any point is given by the brightness level of the image, which is usually called the gray level[4]. In general this brightness level varies from black to white with a continuous distribution. When a picture is processed by a digital computer, it is usually regarded as several discrete arrays of numbers, i.e., as a matrix, rather than as a function. Indeed, any matrix having real, non-negative elements can be thought of as defining a "piecewise constant" picture function.

In digital picture processing, it is often desirable to assume that a picture function can take on only a finite set of values, in other words, that its gray levels are quantized. It is not difficult to show that any picture function is indistinguishable from a quantized picture function, providing that many sufficient levels are allowed. An important special case is that of a binary-valued picture function, which can take on only two values, i.e., "black" and "white" or 0 and 1.

Shannon[6] has defined redundancy as "that fraction of a message or datum which is unnecessary and hence repetitive in the sense that if it were missing the message would still be essentially complete, or at least could be completed." Therefore, the objective of data compression is to reduce the quantity of data in the original signal to a minimum with no loss of information. A familiar example, applicable to a signal composed in part of constant values, is to compress by recording only the positions and magnitudes of significant changes. Another example from the field of image processing is a photograph of a checkboard pattern, which would be highly redudant if transmitted sample by sample.

The use of transforms to accomplish data compression is based on an "unbalancing" effect observed when a transform is applied to data. In the original domain (before transformation) the importance of any datum is relatively constant, while the redundancy is present in the form of the interrelationships between individual data. In the domain of the transform coefficients, however, there is a great disparity in the significance or amount of information in a particular transform coefficient. If the transformation is chosen properly, the information bearing transform coefficients are in many orders of magnitude larger than those reflecting redundancy. The redundancy is reduced by removal of these insignificant terms.

Taking advantage of this basic fact, compression can be accomplished in several ways. A common way is to assume that the low frequencies or sequencies* are important while the high ones represent only noise. The high frequencies or sequencies may then be deleted in order to compress the data. This is usually called zonal filtering. Another technique which has been found more successful for picture compression is that of thresholding or threshold coding. The transformed coefficients are truncated at a threshold and all of those whose absolute values are less than this threshold are set to zero. The employment of a constant threshold value exhibits an adaptive behavior on compression ratio, which is generally defined as the total number of samples per the number of samples transmitted. Crowded areas automatically receive less compression while smooth areas receive high compression. Various combinations of these two techniques are possible.

The basic image compression scheme can be summarized in the following steps:

1. Modeling the original image to a regular-rectangular grid of points by specifying its values (gray levels) $u(x,y)$ at each spatial coordinate (x,y).
2. Applying a transformation to the sampled image.
3. Compressing the transformed samples to a fixed number of bits by either zonal filtering or threshold coding, or a combination of these two.
4. The image is reconstructed by applying the inverse transformation to the compressed samples.

---

* see p. 15 for definition

Now suppose that a picture has been obtained through some image or transmission process that has degraded it. In many cases the degradation can be mathematically inverted and "restored" to its original condition. Furthermore, a two dimensional filter can be accomplished by multiplying the transform of an image, point by point, with a two dimensional filter function,it is able to improve the quality of the image.

To measure the "quality" of a picture, two quantities are commonly used. The first of these called "resolution" relates to the distinguishability of close objects, while the second called "acutance" is concerned with the sharpness of edges(4). They are not necessarily dependent. Specific measures of resolution can be defined by specifying the objects to be seen as distinct as well as the contrast that must exist between the objects and the spaces between them before they can be regarded as distinct. Acutance is usually measured by the average of the square of the rate of change of the gray levels across an edge divided by the total contrast across the edge. Besides, the human factor, i.e. the sensitivity of the eye, plays an important role in the determination of the quality of a picture.

It is often necessary to determine how well two pictures match one another, or conversely how significant they differ from one another. These problems arise most commonly in the area of pictorial pattern recognition and for the comparison of different orthogonal transformations. There are many possible ways of measuring the difference between two pictures or picture functions. Among them the simplest one is to compute their average absolute difference, i.e., the sum of absolute differences at each point divided by the total area of the picture. Another common way is to compute the mean-square error, i.e. their average squared difference.

So far, the performance of different transforms is usually measured by their rate versus distortion functions[7], i.e. the total number of bits required to achieve a particular fidelity criterion. Fidelity criteria considered are the mean-square error between the original and reconstructed images, and the subjective quality of the reconstructed image relative to the original.

There exist many different kinds of transforms which can serve the above-mentioned purpose. Some frequently encountered are the zero-order hold, Fourier, Haar, slant, Karhunen-Loeve , and Walsh-Hadamard transforms. Among these the Fourier transform is no doubt the most well-known and is being applied in almost all possible areas. However, because of the ease and the tremendous speed for the computations involved, currently the Walsh-Hadamard transform attracts more and more researchers to work in this field. Transmitting the Walsh-Hadamard transform of an image rather than the spatial representation of the image provides a potential tolerance to channel errors and the possibility of reduced bandwidth transmisssion.

1.2 Walsh-Hadamard Transforms In Feature Extraction[8,9]

The essence of pattern recognition resides mainly in the selection of few and good features. This applies not only to the main portion of information processing in pattern recognition but also to the preliminary choice of observed phenomena in data collection, as well as to the final stage of decision making. In fact, the decision making can be characterized as selecting a single particular function of observed data such that its positive and negative values should indicate whether each object belongs or does not belong to a class. Even in the descriptive pattern recognition, the descriptive relations between component-parts are nothing but a set of highly nonlinear functions of the observed data. The only difference from the usual method lies in the fact that in the descriptive approach the possible classes become so large that sometimes they are not countable at the outset.

We assume that we are given a collection of objects each of which is described by a vector. Each component of the vector is the measured value of a certain feature. The selection of observed features is arbitrary, but we accept their values as our starting point, and try for a further selection of features from among the linear combinations of these given data.

We assume that there will be K classes S_k, indexed by $k=1, 2, \ldots, K$. Let the vector stated above have N components, in other words, the dimensionality of our pattern space is equal to N. Let N be indexed by $n=1, 2, \ldots, N$. Let the pattern space be designated by the subscript x and the transform space by subscript A. Then assuming that a second-order statistic (a covariance function) can be measured or modeled for each class S_k, let it be designated $\underline{\Phi}_x^{(k)}$. Then a generalized covariance matrix becomes

$$\underline{\Phi}_x = \sum_{k=1}^{K} P(S_k)\, \underline{\Phi}_x^{(k)} \qquad (1.1)$$

where the $P(S_k)$ is the a priori class distribution.

A linear transformation provided by a unitary operator $\underline{A}$ will map the pattern space into a transform space whose basis vectors are the orthogonal columns of the matrix $\underline{A}$. The features in the new space are linear combinations of the original axes according to the structure of $\underline{A}$. The second-order statistics in the transform space become

$$\underline{\Phi}_A = \underline{A}^T \underline{\Phi}_x \underline{A} \tag{1.2}$$

The objective now is to find a set of features such that the distribution of importance becomes as uneven as possible, so that we can ignore eventually those features which are not important.

The different kinds of transforms mentioned in the previous section can all be used for this purpose simply because all of them are orthogonal transformations. Among them the Karhunen-Loeve transform is optimum in a variety of criteria (10, 11), of which mean-square truncation error and entropy interpretation are just two. The transformation results in a covariance function that is diagonal with entries equal to the eigenvalues of $\underline{\Phi}_x$:

$$\underline{\psi}_x^T \underline{\Phi}_x \underline{\psi}_x = \underline{\lambda} \tag{1.3}$$

where $\underline{\psi}_x$ and $\underline{\lambda}$ are matrices of eigen-vectors and eigenvalues of $\underline{\Phi}_x$. The coefficients in the transform space become statistically uncorrelated and for Gaussian statistics become statistically independent. However, the transformation requires considerable computation to form and diagonalize the covariance matrices. It becomes unpractical when the covariance matrix is relatively large.

The Walsh-Hadamard transform is particularly appealing in this respect for a variety of reasons. First, it is implementable in $N \log_2 N$ additions without any multiplications (12, 13). Both the Walsh functions and the Hadamard matrix are binary in nature, making the transform quite appealing for semiconductor special-purpose computer implementation. It has an analogy between sequency and the conventional frequency. Thus, this kind of transform has been used widely for feature extraction[14-16].

1.3 The Logical Walsh-Hadamard Transform

Assume that the data being transformed is restricted to be binary. This assumption is logical since the computer processes mainly with binary data internally. Because of the sampling theory it is also general in the sense that a frequency or sequency band-limited signal can be completely represented by the discrete samples taken at a rate that is at least twice the highest frequency or sequency present in the signal[17]. Furthermore, the samples are quantized into levels which are used to be a power of two. A good example is the gray levels mentioned in the first section. For this reason Searle[18] in 1970 modified the conventional Walsh-Hadamard transform and introduced the so-called "Logical Walsh-Hadamard Transform". This kind of modification appears to be useful in the following respects.

Firstly, the coefficients of the transform are supposed to contain no more information than the data. This is due to the fact that the amount of information contained in each sample is proportional to its energy while the Walsh-Hadamard transform obeys the law of conservation of energy. However, some of these coefficients contain more bits than a single datum.

Example

Data	in decimal	3	2	0	3
	in binary	11	10	00	11
Coeffs.	in decimal	8	-2	2	4
	in binary	1000	-10	10	100

If each datum occupies one computer word, the coefficients may require more space. This forms a certain type of redundancy. In logical Walsh-Hadamard transform this kind of redundancy is removed, hence a logical transform contains no more bits than the original data.

Secondly, both the elements of Hadamard matrix and the data are of a binary nature, it would be more appropriate to compute with the Boelean operations rather than the arithmetic operations. In this thesis a fast Boolean algorithm is obtained for this logical transform which would require little or no intermediate storage.

Finally, since the logical Walsh-Hadamard transform has exactly the same number of bits as the original data, it becomes a fixed-length operation and can be easily implemented by hardware. Furthermore, for fixed-length operations the computer can handle it with ease and higher speed.

However, the general techniques used by the conventional Walsh-Hadamard transform for data compression or bandwidth reduction, namely, the zonal filtering and the threshold coding as mentioned in the first section, can not apply in the logical transform case. This is due to the fact that the logical transform takes on values of zeroes and ones only and is restricted to binary data. In fact, the logical transform has included the thresholding operation in performing the transformation. But, by virtue of the run-length coding technique[20,21] a saving in either time or bandwidth is possible. The basic idea is to transmit only the lengths of the one and zero runs alternatively instead of transmitting the message bit by bit.

1.4 Organization of the Thesis

The organization of the thesis is as follows. Each chapter except the first three and the last is preceded by an introductory section describing its contents and the work carried out.

The basic material that is common to all chapters is presented in Chapter 2.

Chapter 3 presents the logical Walsh-Hadamard transform, its origin and its implementation. The chapter begins with the historic background and the definition of the logical Walsh-Hadamard transform in Section 3.1. The transform is listed for N=2 and 4 cases with its combinatorial network provided in Section 3.2. Section 3.3 deals with N=8 case. Analysis of the 8-bit logical Walsh-Hadamard transform is made in Section 3.4. Section 3.5 provides the reasoning for the threshold setting of the logical transform. A short summary of the chapter is contained in Section 3.6.

The piecewise logical Walsh-Hadamard transform is introduced in Chapter 4. The idea is to extend the logical transform for longer and reasonable length by increasing the dimension or performing permutation. The two-dimensional logical Walsh-Hadamard transform is first introduced and defined in Section 4.1. In this section a theorem of uniqueness and the combinatorial network for the two-dimensional case are also provided. Section 4.2 gives a formal proof of the well-known theorem relating two-dimensional with one-dimensional Walsh-Hadamard transforms. In Section 4.3 we define the piecewise logical Walsh-Hadamard transform and give its relationship with the two-dimensional logical transform. We look into the practical application of the logical transform in image processing in Section 4.4. Section 4.5 covers the higher dimensional case by using permutation. A short summary of the chapter is also provided in Section 4.6.

From Chapter 5 on, we make a detour from the logical transform and study the applications of the H diagram. Chapter 5 presents a new approach for obtaining the fast Walsh-Hadamard transform by using H diagrams. The chapter starts with the relationship between the Hadamard matrix and the Walsh-Hadamard transform in Section 5.1. Section 5.2 gives a modified factorization method for the Hadamard matrix. An algorithm for obtaining the fast Walsh-Hadamard transform with illustrating examples is shown in Section 5.3. Section 5.4 concludes the chapter with a summary.

Chapter 6 presents the relations among the different characteristic parameters of Walsh functions. By using H diagrams, certain spectral analysis concerning axis-symmetry is also provided. The dyadic translation is first introduced and defined in Section 6.1. Then, the axis of dyadic translation is related to those of H diagram. In Section 6.2 the axis-symmetry of a function is defined, and it is shown that each Walsh function possesses a unique axis-symmetry. Section 6.3 shows the relations of sequency, axis-symmetry, row-index and period of a Walsh function. Analogous to the sequency or frequency spectrum, an axis-symmetry spectrum is introduced and defined in Section 6.4. It can be easily obtained on an H diagram. The spectrum is analyzed by using H diagrams in Section 6.5 for those functions that possess axis-symmetries. A short summary of the chapter is included in Section 6.6.

Finally, Chapter 7 concludes with a discussion and comments for further studies.

CHAPTER 2

FUNDAMENTALS AND DEFINITIONS

In this chapter we are going to introduce some basic material which will be considered common to all the following chapters. The reader is referred to references (17, 19, 23, 42) for additional details.

2.1 Orthogonality And Linear Independence

Definition 2.1 : A system f(j, x) of real and almost everywhere nonvanishing functions f(0, x), f(1, x), ... is called orthogonal in the interval $x_0 \le x \le x_1$ if the following condition holds :

$$\int_{x_0}^{x_1} f(j, x)\, f(k, x)\, dx = X_j\, \delta_{jk}$$

$$\delta_{jk} = 1 \text{ for } j = k, \qquad \delta_{jk} = 0 \text{ for } j \neq k$$

The functions are called orthogonal and normalized, or orthonormal, if the constant X_j is equal to 1. A nonnormalized system of orthogonal functions may always be normalized.

Definition 2.2 : A system f(j, x) of m functions is called linearly dependent, if the equation

$$\sum_{j=0}^{m-1} c(j)\, f(j, x) \equiv 0 \qquad (2.1)$$

is satisfied for all values of x without all constants c(j) being zero. The functions f(j, x) are called linearly independent, if Eq. (2.1) is not satisfied.

Functions of an orthogonal system are always linearly independent. And a system of m linearly independent functions can always be transformed into a system of m orthogonal functions.

Definition 2.1.3 A square matrix with real elements is called an orthogonal matrix if its inverse and its transpose are equal except for a factor.

Definition 2.4 Let an arbitrary function F(x) be expanded in a series of the orthogonal functions f(j, x) :

$$F(x) = \sum_{j=0}^{\infty} a(j)\, f(j, x) \tag{2.2}$$

The value of the coefficients a(j) may be obtained by multiplying Eq. (2.2) by f(k, x) and integrating the products in the interval of orthogonality $x_0 \leq x \leq x_1$:

$$\int_{x_0}^{x_1} F(x)\; f(k, x)\, dx = a(k)$$

A series $\sum_{j=0}^{m-1} a(j)\, f(j, x)$ having m terms is called the representation of F(x) with least mean square deviation.

Definition 2.5 The mean square deviation Q of F(x) from its representation is defined as :

$$Q = \int_{x_0}^{x_1} [F(x) - \sum_{j=0}^{m-1} b(j)\, f(j, x)]^2\, dx$$

$$= \int_{x_0}^{x_1} F^2(x)dx - \sum_{j=0}^{m-1} a^2(j) + \sum_{j=0}^{m-1} [b(j) - a(j)]^2$$

The last term vanishes for b(j) = a(j) and the mean square deviation assumes its minimum.

Definition 2.6 The system f(j, x) is called orthonormal and complete, if the mean square deviation Q converges to zero with increasing m for any function F(x) that is quadratically integrable in the interval $x_0 \le x \le x_1$:

$$\lim_{m \to \infty} \int_{x_0}^{x_1} [\, F(x) - \sum_{j=0}^{m-1} a(j)\, f(j,\, x)\,]^2 \, dx = 0$$

This follows that

$$\sum_{j=0}^{\infty} a^2(j) = \int_{x_0}^{x_1} F^2(x)dx$$

which is known as completeness theorem or Parseval's theorem. Its physical meaning is the conservation of energy under orthogonal transformations.

2.2 Walsh Functions And Hadamard Matrices

Definition 2.7 : Following Paley's modification [42], the Rademacher functions are defined as

$$R_0(m) = 1 \qquad 0 \le m \le 1$$

$$R_1(m) = \begin{cases} 1 & 0 \le m < 1/2 \\ -1 & 1/2 \le m < 1 \end{cases}$$

and $R_1(m+1) = R_1(m)$, $R_s(m) = R_1(2^{s-1}m)$ $(s = 2,3,\dots)$

The Rademacher functions form an incomplete orthonormal set. If we extend these functions to a complete orthonormal set by multiplication, we will obtain the Walsh functions.

Definition 2.8 : The Walsh functions are defined as

$$\text{wal}\,(s,m) = [R_n(m)]^{g_n} [R_{n-1}(m)]^{g_{n-1}} \dots [R_2(m)]^{g_2} [R_1(m)]^{g_1}$$

where $(g_n g_{n-1} \dots g_2 g_1)$ is the Gray code representation of s and $R_i(m)$ is the ith Rademacher function defined in Definition 2.7.

The product of two Walsh functions yields another Walsh function:

$$\text{wal}(h,m)\ \text{wal}(k,m) = \text{wal}(h \oplus k, m)$$

where sign $\oplus$ stands for an addition modulo 2 while both h and k are written as binary numbers.

Definition 2.9 : Sequency is defined as the number of zero crossings or sign changes within the unit interval defined for each Walsh function.

The sequency is a parameter which characterizes a Walsh function. Hence, it is unique to a specific Walsh function. Various analogies exist between the Fourier and Walsh-Hadamard transforms with sequency playing the role of frequency.

Definition 2.10 : Walsh functions can be further classified as "sal" (sine-Walsh) or "cal"(cosine-Walsh) based on the odd or even symmetry of wal (s,m) about the axis at m = 1/2, i.e.,

$$\text{wal}(2s-1,m) = \text{sal}(s,m) \qquad \text{for s being odd}$$

$$\text{wal}(2s,m) = \text{cal}(s,m) \qquad \text{for s being even.}$$

Definition 2.11 : Let $\underline{A}$ be an (NxL) matrix with its element at the ith row and jth column denoted as a_{ij} and let $\underline{B}$ be another matrix of any dimension. The Kronecker-product of $\underline{A}$ and $\underline{B}$ denoted by $\underline{A} \otimes \underline{B}$ is defined as :

$$\underline{A} \otimes \underline{B} = \begin{bmatrix} a_{11}\underline{B} & a_{12}\underline{B} & \cdots & a_{1L}\underline{B} \\ a_{21}\underline{B} & a_{22}\underline{B} & \cdots & a_{2L}\underline{B} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N1}\underline{B} & a_{N2}\underline{B} & \cdots & a_{NL}\underline{B} \end{bmatrix}$$

The Kronecker-product is also called the direct product.

Definition 2.12 : A Hadamard matrix is a symmetric square matrix of elements plus one and minus one. Its rows (and columns) are mutually orthogonal. For $N = 2^n$, where n is an integer, a Hadamard matrix can be constructed by recursively applying the Kronecker-product. With $N = 2^n$ and

$$\underline{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\underline{H}_N = \underbrace{\underline{H}_2 \otimes \underline{H}_2 \otimes \cdots\cdots \otimes \underline{H}_2}_{n}$$

$$= \underline{H}_2^{[n]}$$

where the bracketed exponent means that the Kronecker-product is performed n times upon $\underline{H}_2$.

2.3 H Diagrams:

An H diagram is a geometric model which can serve as an effective visual aid in the analysis of binary functions. It is based on a geometric transformation of the coordinates of a hypercube onto a two-dimensional plane. The two variable case takes the form of a figure "H" from which the name H diagram is derived.

Starting at a central origin, the first binary variable x_1 and its complement are represented by vectors along the horizontal axis. The next variable x_2 is added by constructing members parallel to the vertical axis, as shown in Fig. 2.1(a)*. In each case, the conventional positive and negative directions are chosen, respectively, for the unprimed variable and its complement. The extremities of

* Fig. 2.1 is adopted from Fig. 1, p 1193 in Marihugh's paper[19]

the H diagram represent the canonical terms of a Boolean function in the same manner that vertices are used in the hypercube representation. The location of each term is given by the vector sum of the corresponding variables. In the hypercube case a third variable x_3 can be added, shown by the solid lines of Fig. 2.1(b), generating the coordinate framework of a cube. In the H diagram, the $(x_3, \bar{x}_3)$ vectors are rotated into a common plane as shown in Fig. 2.1(c).

Fig. 2.1(d) shows the extension of the H diagram to four variables. The numerical sequence follows a regular pattern that alternates about successive variables. From 0, the sequence first alternates about the least significant variable x_4 to cell 1. The next two numbers occur in the same geometric order, but shifted about the next significant variable x_3. By repeating the process in the same fashion a numerical sequence as in Fig. 2.1(d) can be obtained.

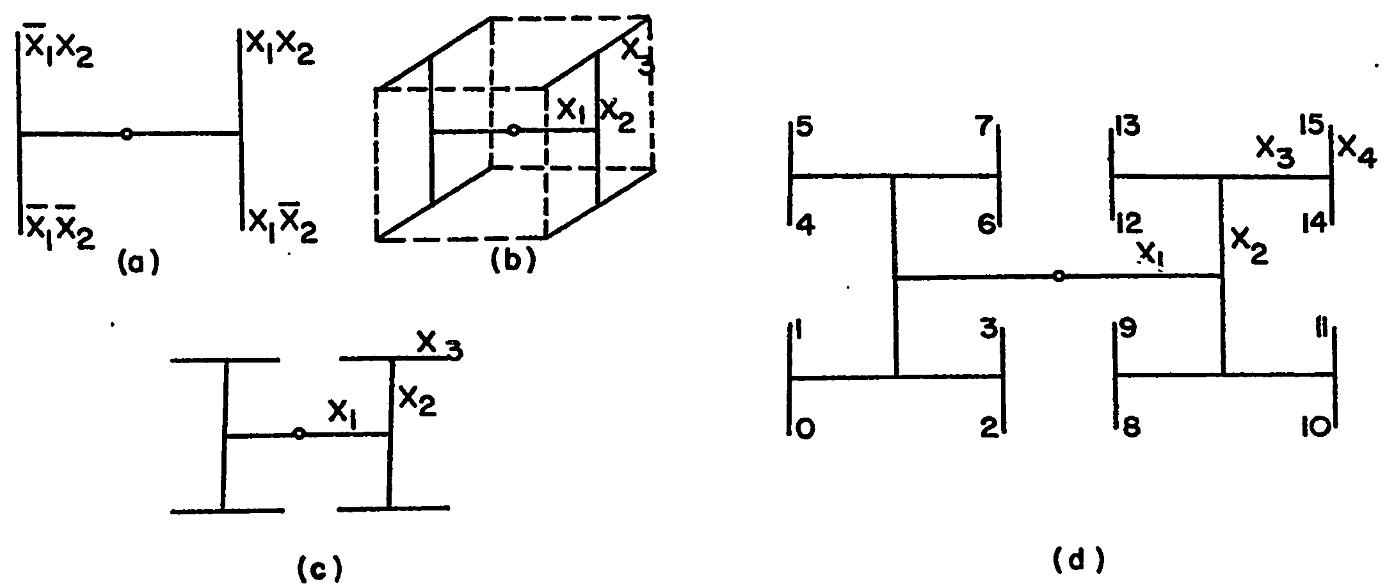


Fig. 2.1 Construction of H diagrams.

CHAPTER 3

THE LOGICAL WALSH-HADAMARD TRANSFORM

3.1 Historic Background

In order to eliminate redundancy resulting from the application of the Walsh-Hadamard transform to binary data, and to take advantages of the fact that both the input data and the elements of an Hadamard matrix are binary, Searle [18] modified the conventional Walsh-Hadamard transform and thereby introduced the so-called logical Walsh-Hadamard transform (abbreviated LWHT). He pointed out that if a binary data string of length N = 4, or 8 was transformed by multiplying it by an Hadamard matrix of order N, and that if the first bit of the data string is one, then only the sign bits of its transform are necessary in order to completely restore the information of the original data. Furthermore, the original data can be recovered by applying the same procedure to the sign bits of the transform. The set of ordered sign bits is called the LWHT of the original data.

Since both the data set and the sign bits of the coefficients of the transform are binary, it is sometimes convenient to denote them by their decimal representations. Let N be the length of the input data string, and it is fixed for both the data and its transform coefficient bits. If the input and output bits are denoted by the ordered sets $\{x_0, x_1, \ldots, x_{N-1}\}$ and $\{z_0, z_1, \ldots, z_{N-1}\}$ respectively, then their decimal representations will be denoted correspondingly as X and Z. The relationships among them will be

$$X = \sum_{i=0}^{N-1} x_i \; 2^{N-i-1} \tag{3.1}$$

$$Z = \sum_{k=0}^{N-1} z_k \; 2^{N-k-1} \tag{3.2}$$

and

$$Z = \text{LWHT}\ (X) \tag{3.3}$$

The coefficient bits $\{z_k\}$ for $k = 1, 2, \ldots N-1$ are defined by the operation of thresholding :

$$z_k = \begin{cases} 1 & \text{if } \sum_{i=0}^{N-1} h_{ki} x_i > 0 \\ 0 & \text{if } \sum_{i=0}^{N-1} h_{ki} x_i \leq 0 \end{cases} \tag{3.4}$$

where h_{ki} is the element in the kth row and the ith column of the (NxN) Hadamard matrix.

A mapping function can be defined instead to take into account this thresholding operation :

$$f_M : \left\{ \begin{matrix} \text{The set of} \\ \text{integers} \end{matrix} \right\} \longrightarrow \left\{ \begin{matrix} \text{The set of} \\ \text{binary numbers} \end{matrix} \right\}$$

and

$$f_M(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \end{cases} \tag{3.5}$$

Therefore,

$$z_k = f_M\left(\sum_{i=0}^{N-1} h_{ki} x_i \right) \quad \text{for } k=1, 2, \ldots N-1 \tag{3.6}$$

For $k = 0$, and $x_0 = 1$, z_k is always equal to unity since $h_{0i} = 1$ for all i's and x_i takes on values 0 and 1 only. Correspondingly the zeroth order bit of the data as mentioned before is restricted to be unity. Hence,

$$z_k = x_k \qquad \text{for } k = 0 \tag{3.7}$$

For the case that $x_0 = 0$, the operation of complementation has to be introduced. Let $\bar{x}_i$, $\bar{z}_i$, $\bar{X}$ and $\bar{Z}$ be the complements of x_i, z_i, X and Z respectively. Then $\bar{X}$ and $\bar{Z}$ will be defined as :

$$\bar{X} = 2^N - 1 - X = \sum_{i=0}^{N-1} \bar{x}_i \, 2^{N-i-1} \tag{3.8}$$

$$\bar{Z} = 2^N - 1 - Z = \sum_{k=0}^{N-1} \bar{z}_k \, 2^{N-k-1}$$

To obtain the LWHT in this case the data is required to be complemented twice, one before and one after the transformation. Using the notations defined as above,

$$Z = \overline{\text{LWHT}\ (\overline{X})} \quad \text{or} \quad \overline{Z} = \text{LWHT}\ (\overline{X}) \tag{3.9}$$

If all the above-mentioned rules are followed and N is restricted to 2, 4 or 8, then the transform will be reversible. That is,

$$X = \text{LWHT}\ (Z) = \text{LWHT}\ [\text{LWHT}\ (X)] \tag{3.10}$$

However, for $N \geq 16$, these simple properties and relationships no longer hold as pointed out by Parkyn Jr. and Cash(24). In this thesis the LWHT is extended to cover the $N \geq 16$ case without losing its simplicity and reversibility. This is achieved by increasing the dimension of the transform and using the LWHT's for N = 2, 4 and 8 as the building blocks. The LWHT's for N=2, 4 and 8 are first implemented by a set of switching functions each and realized by a combinatorial network using majority gates and NAND gates. According to the nature of a combinatorial network, the output LWHT is instantaneous, i.e., almost immediately after the input data is applied. Hence, the transform will be given directly without any computation.

3.2 The LWHT for $N \leq 8$

Tables 3.1(a) and 3.1(b) list the LWHT's for N=2 and 4 respectively. If the x_i's and the z_k's are considered as the inputs and outputs of a set of switching functions, then the tables are nothing but the truth tables for the switching functions. For these two cases the minimal forms of the output functions can easily be obtained by using either Karnaugh maps or tabular method. However, when N is raised to 8, both of them will become cumbersome and inefficient. Hence the output functions will be constructed by first plotting the true vertices for each output in a separated H diagram[19] as shown in Figs. 3.1 and 3.2.

Table 3.1(a)

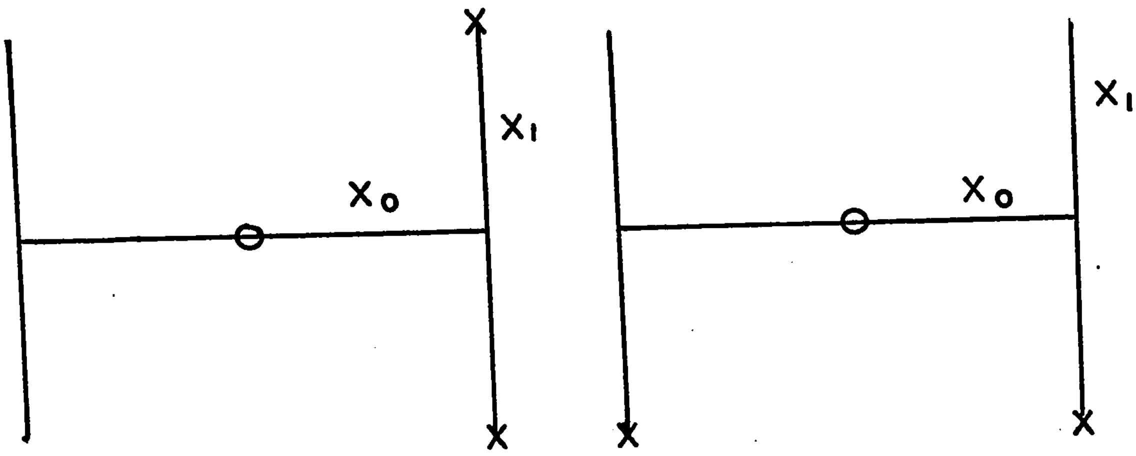
LWHT For N = 2 Case

X	x_0	x_1	z_0	z_1	Z
0	0	0	0	1	1
1	0	1	0	0	0
2	1	0	1	1	3
3	1	1	1	0	2

Table 3.1(b)

LWHT For N = 4 Case

X	x_0	x_1	x_2	x_3	z_0	z_1	z_2	z_3	Z
0	0	0	0	0	0	1	1	1	7
1	0	0	0	1	0	0	0	1	1
2	0	0	1	0	0	1	0	0	4
3	0	0	1	1	0	1	0	1	5
4	0	1	0	0	0	0	1	0	2
5	0	1	0	1	0	0	1	1	3
6	0	1	1	0	0	1	1	0	6
7	0	1	1	1	0	0	0	0	0
8	1	0	0	0	1	1	1	1	15
9	1	0	0	1	1	0	0	1	9
10	1	0	1	0	1	1	0	0	12
11	1	0	1	1	1	1	0	1	13
12	1	1	0	0	1	0	1	0	10
13	1	1	0	1	1	0	1	1	11
14	1	1	1	0	1	1	1	0	14
15	1	1	1	1	1	0	0	0	8



(a) z_o (b) z_1

Fig. 3.1 H Diagram of z_k's For N = 2 Case

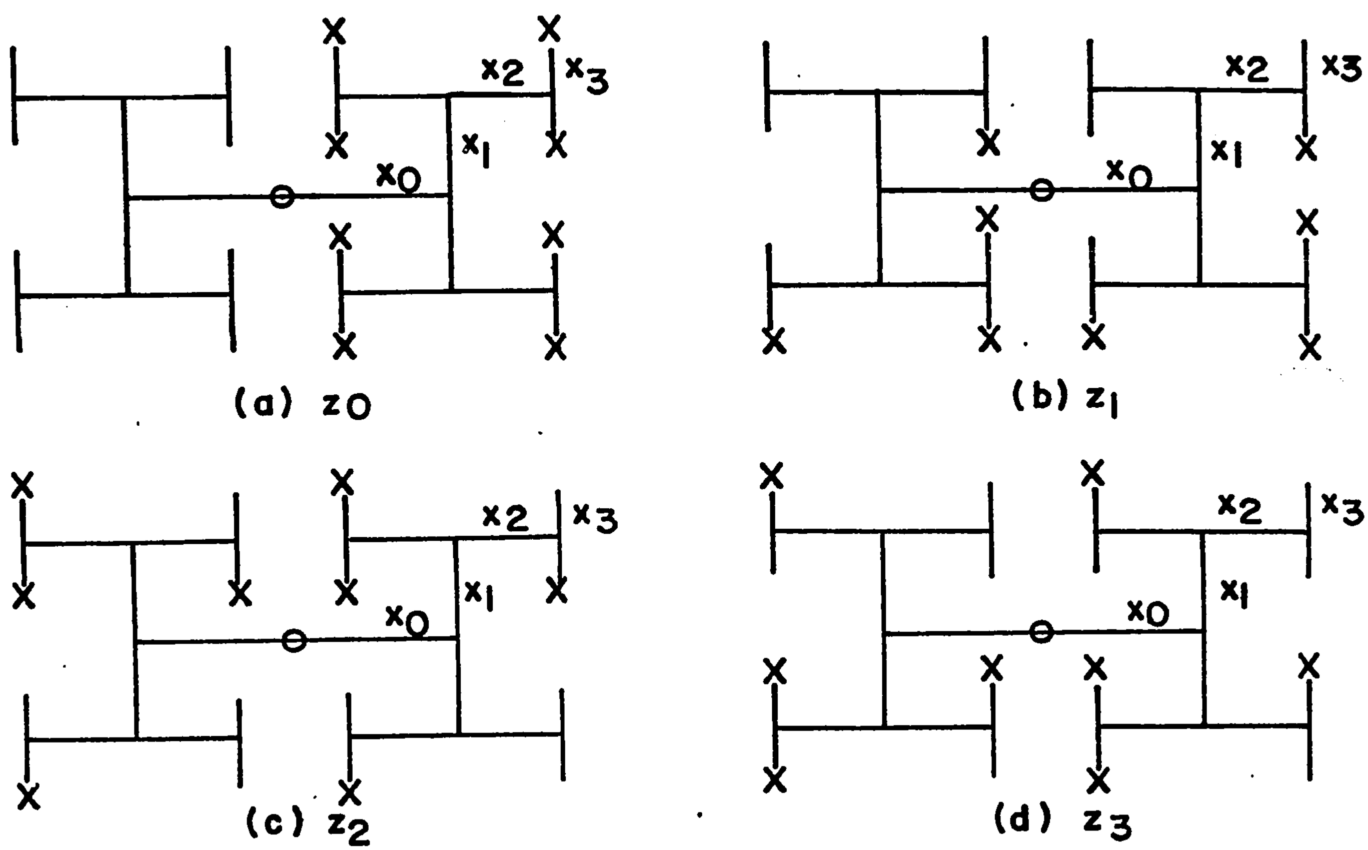


Fig. 3.2 H Diagrams of z_r's For N = 4 Case

The switching functions can be obtained in minimal forms from these H diagrams.

For N = 2

$$z_0 = x_0$$
$$z_1 = \bar{x}_1 \qquad (3.11)$$

For N = 4

$$z_0 = x_0$$
$$z_1 = \bar{x}_1 x_2 + \bar{x}_1 \bar{x}_3 + x_2 \bar{x}_3$$
$$z_2 = x_1 \bar{x}_2 + x_1 \bar{x}_3 + \bar{x}_2 \bar{x}_3 \qquad (3.12)$$
$$z_3 = \bar{x}_1 \bar{x}_2 + \bar{x}_1 x_3 + \bar{x}_2 x_3$$

These two sets of switching functions are consistent to the definition of the LWHT given before such that z_0 is equal to x_0, and that the rest of the outputs are independent of x_0. Moreover, for N = 4, z_1, z_2 and z_3 are totally symmetric functions, and also δ-functions.(21) For the reason that the arguments of the δ-function might be omitted for successive uses without causing any ambiguity, the notation of the δ-function will be modified. This is simply to denote a δ-function - $\delta_m(x_1, x_2, \ldots, x_n)$, by $M_{m/n}(x_1, x_2, \ldots, x_n)$, where n refers to the number of variables and m is the original subscript for δ, hence m is always less than or equal to n. Only by such a slight modification the comparison of the structure of different switching functions for the N=8 case can be made with a quick glance.

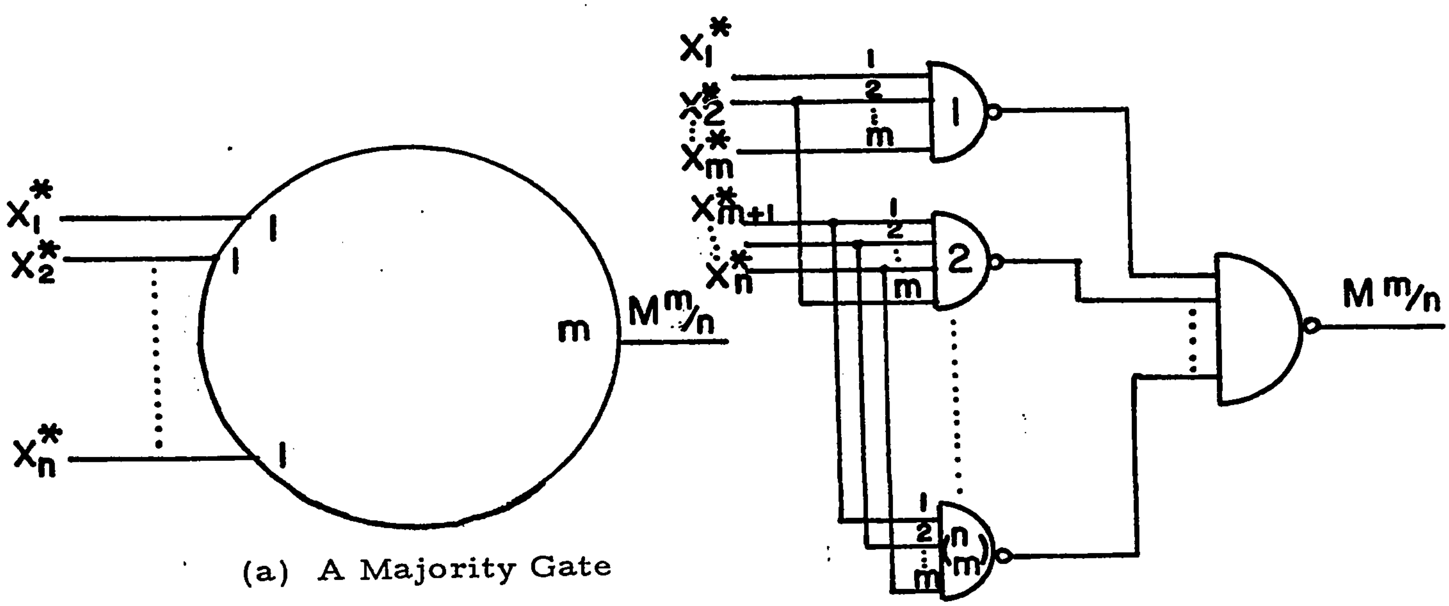
Now if the δ-function is defined in terms of symmetric function as :

$$M_{m/n}(x_1^*, x_2^*, \ldots x_n^*) = S_{\{n, (n-1), \ldots, m\}}(x_1^*, x_2^*, \ldots x_n^*) \qquad (3.13)$$

where M and S represent the δ-function and the symmetric function respectively, the stars denote that the variables are in either complemented or uncomplemented form, n is the total number of input variables, m is a positive integer less than n and the subscript for S indicates the set of a-numbers of the symmetric function[21, 22]. The set of equations in (3.12) can thus be rewritten as :

$$
\begin{aligned}
z_0 &= x_0 \\
z_1 &= M_{2/3}(\bar{x}_1, x_2, \bar{x}_3) \\
z_2 &= M_{2/3}(x_1, \bar{x}_2, \bar{x}_3) \\
z_3 &= M_{2/3}(\bar{x}_1, \bar{x}_2, x_3)
\end{aligned}
\tag{3.14}
$$

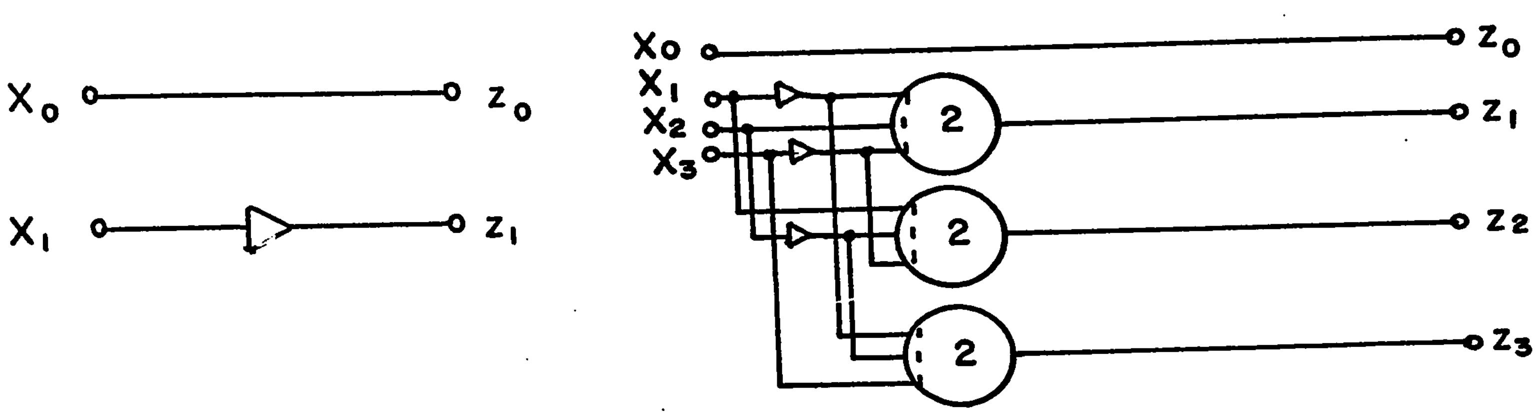
The δ-function defined in (3.13) can easily be implemented by a threshold element —— the so called 'majority gate'[22], or by a two-layer NAND network as shown in Fig. 3.3. The combinatorial network for the 4-bit LWHT is then shown in Fig. 3.4(b). The combinatorial network for the 2-bit LWHT is obvious, however, for the completeness of the work it is shown as well in Fig. 3.4(a).



(a) A Majority Gate

(b) A Two-Layer NAND Network

Fig. 3.3 Implementation of $M_{m/n}(x_1^*, x_2^*, \ldots\ldots x_n^*)$.



(a) N = 2 (b) N = 4

Fig. 3.4 A Combinatorial Network For The LWHT

3.3 The LWHT For N = 8

The dynamic range of X and Z as defined in equations (3.1) and (3.2) increases rapidly as N increases from 4 to 8. To simplify the process a bit, we are going to take advantage of the basic properties of the LWHT, i.e., $z_0 = x_0$ and that z_i is independent of x_0 for all $i \neq 0$. Therefore, Table 3.2 lists only the LWHT of N = 8 for $2^{N-1} \leq X \leq 2^N - 1$ in descending order. The H diagrams for z_1 to z_7 in terms of x_1 to x_7 are shown in Fig. 3.5.

TABLE 3.2

LOGICAL WALSH-HADAMARD TRANSFORM FOR N = 8 CASE

INPUT DATA								TRANSFORM OUTPUT							
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	1
1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	0	1	0	1	0	0	0
1	1	1	1	1	0	1	1	1	1	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0
1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0
1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1
1	1	1	1	0	1	1	0	1	0	0	0	1	0	0	1
1	1	1	1	0	1	0	1	1	0	0	0	1	1	0	0
1	1	1	1	0	1	0	0	1	0	1	0	1	1	0	1
1	1	1	1	0	0	1	1	1	0	0	0	1	0	1	0
1	1	1	1	0	0	1	0	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	1	1	0	0	1	1	1	1	0
1	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0
1	1	1	0	1	1	1	1	1	1	1	0	0	1	1	0
1	1	1	0	1	1	1	0	1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0
1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0
1	1	1	0	1	0	1	1	1	1	0	0	0	0	1	0
1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0
1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	0
1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0
1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	0
1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	1
1	1	1	0	0	1	0	1	1	0	1	0	1	1	1	0
1	1	1	0	0	1	0	0	1	0	1	0	1	1	0	0
1	1	1	0	0	0	1	1	1	1	0	0	1	1	1	0
1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0
1	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	0	0	1	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1	1	0	1	1	0	0	1	1
1	1	0	1	1	1	1	0	1	0	1	0	0	0	0	1
1	1	0	1	1	1	0	1	1	0	1	1	0	0	0	0
1	1	0	1	1	1	0	0	1	0	1	1	1	0	0	1
1	1	0	1	1	0	1	1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	1	0	1	1	1	1	1	0	1	1
1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	0
1	1	0	1	1	0	0	0	1	0	1	1	1	0	0	0

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

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

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

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

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

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

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

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

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

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

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

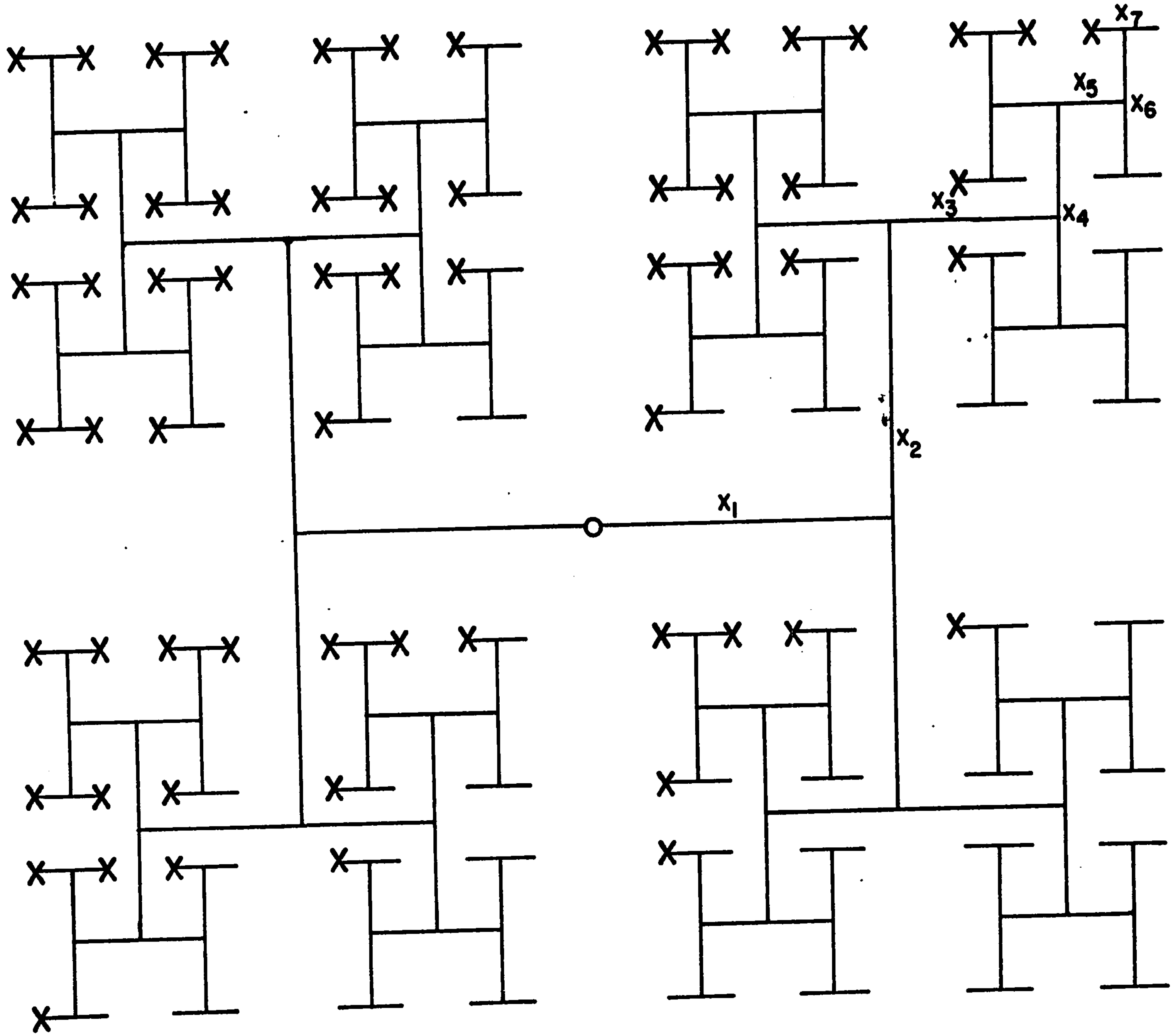


FIG.3-5(a) H-DIAGRAM FOR Z_1

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

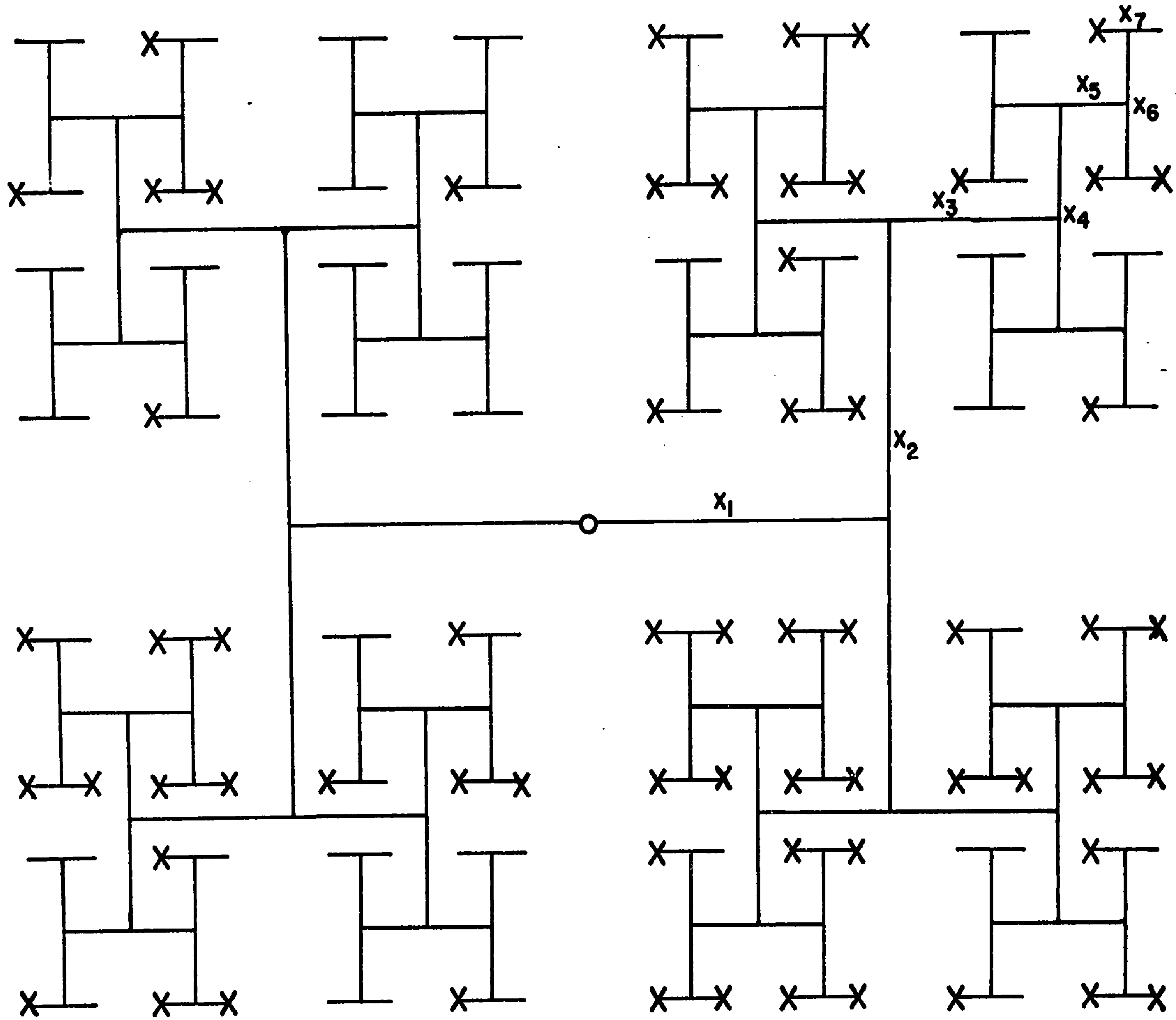


FIG.3-5(b) H-DIAGRAM FOR Z_2

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

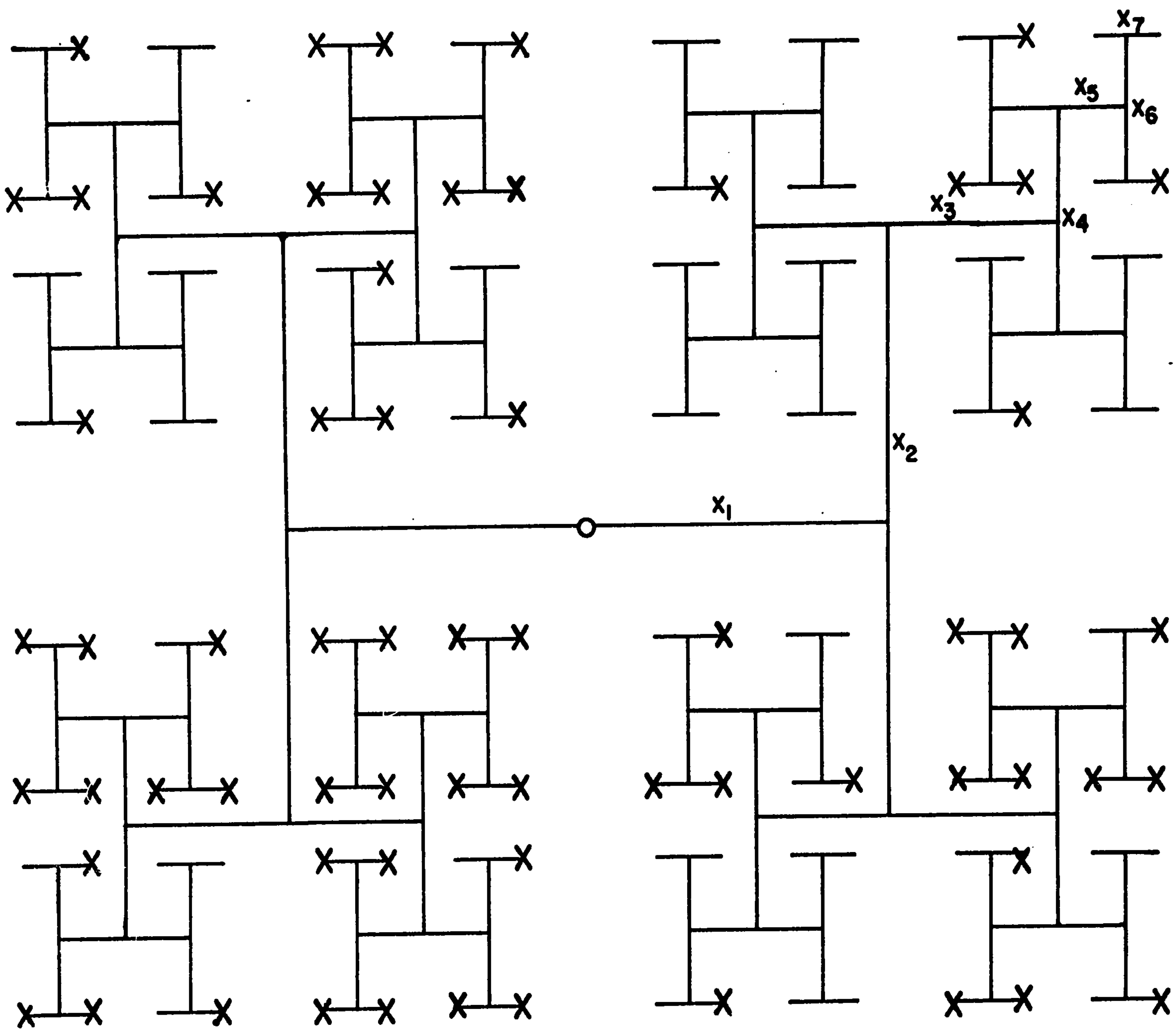


FIG.3'-5(c) H-DIAGRAM FOR Z_3

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

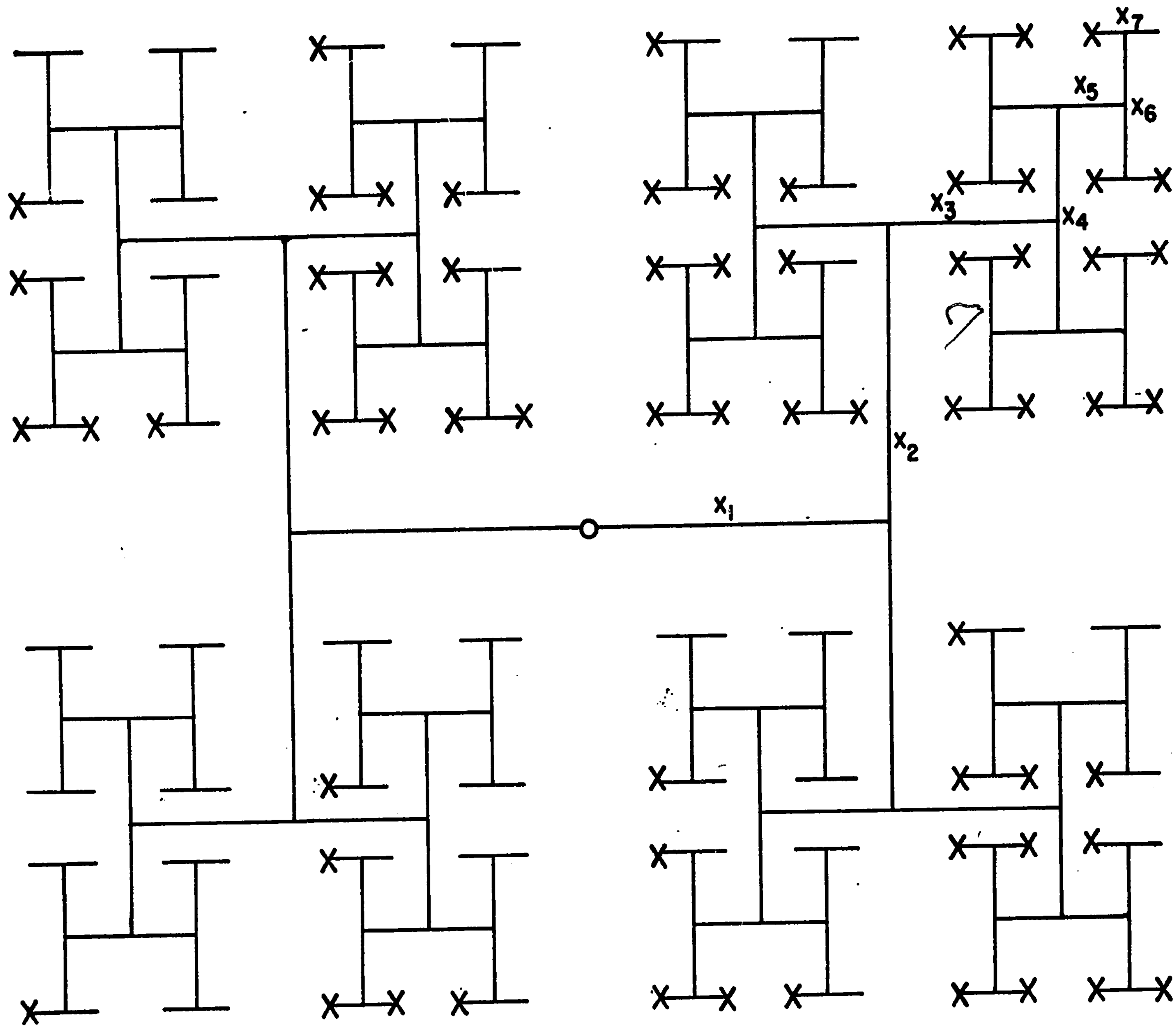


FIG.3-5(d) H-DIAGRAM FOR Z_4

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

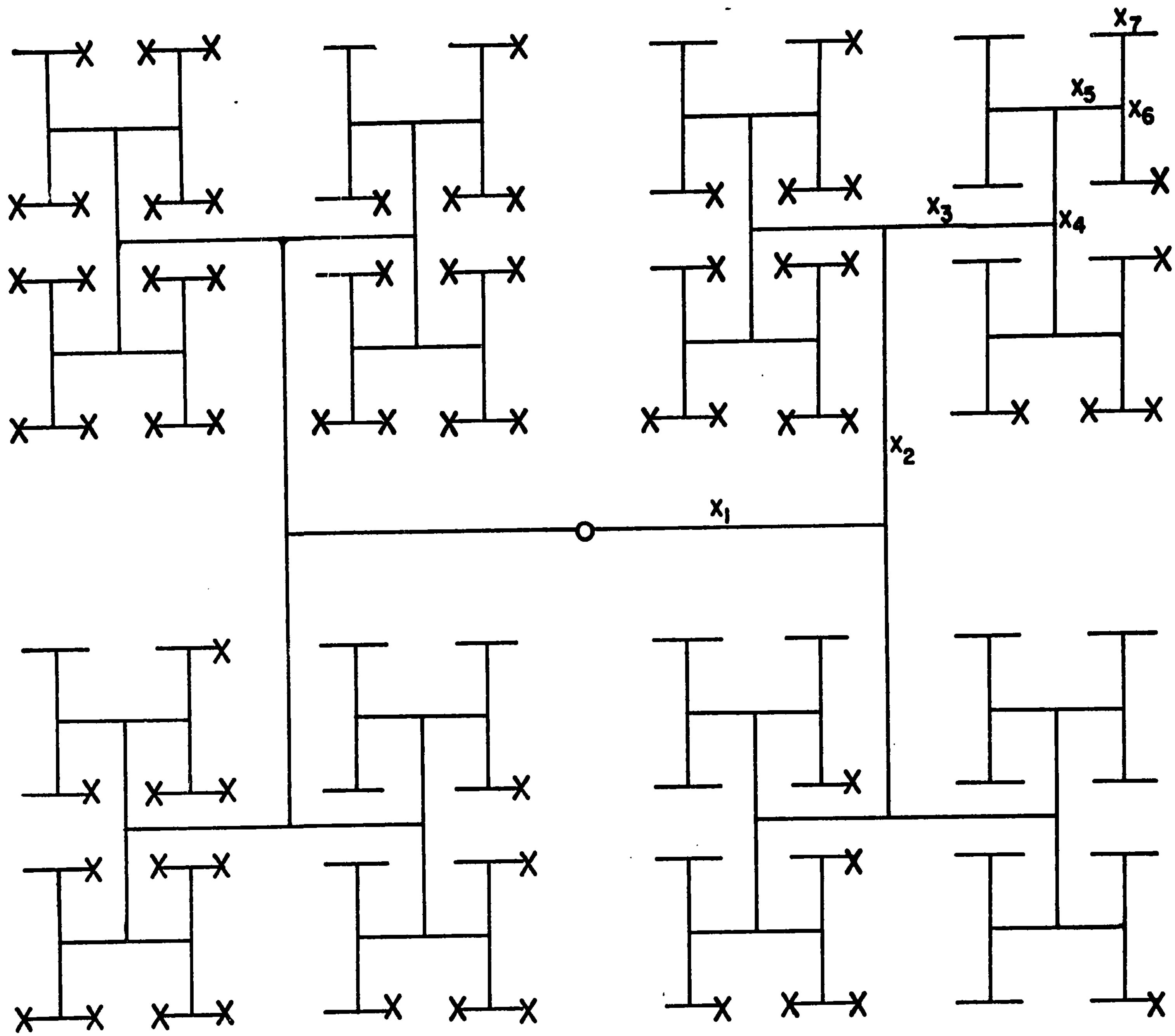


FIG.3-5(e) H-DIAGRAM FOR Z_5

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

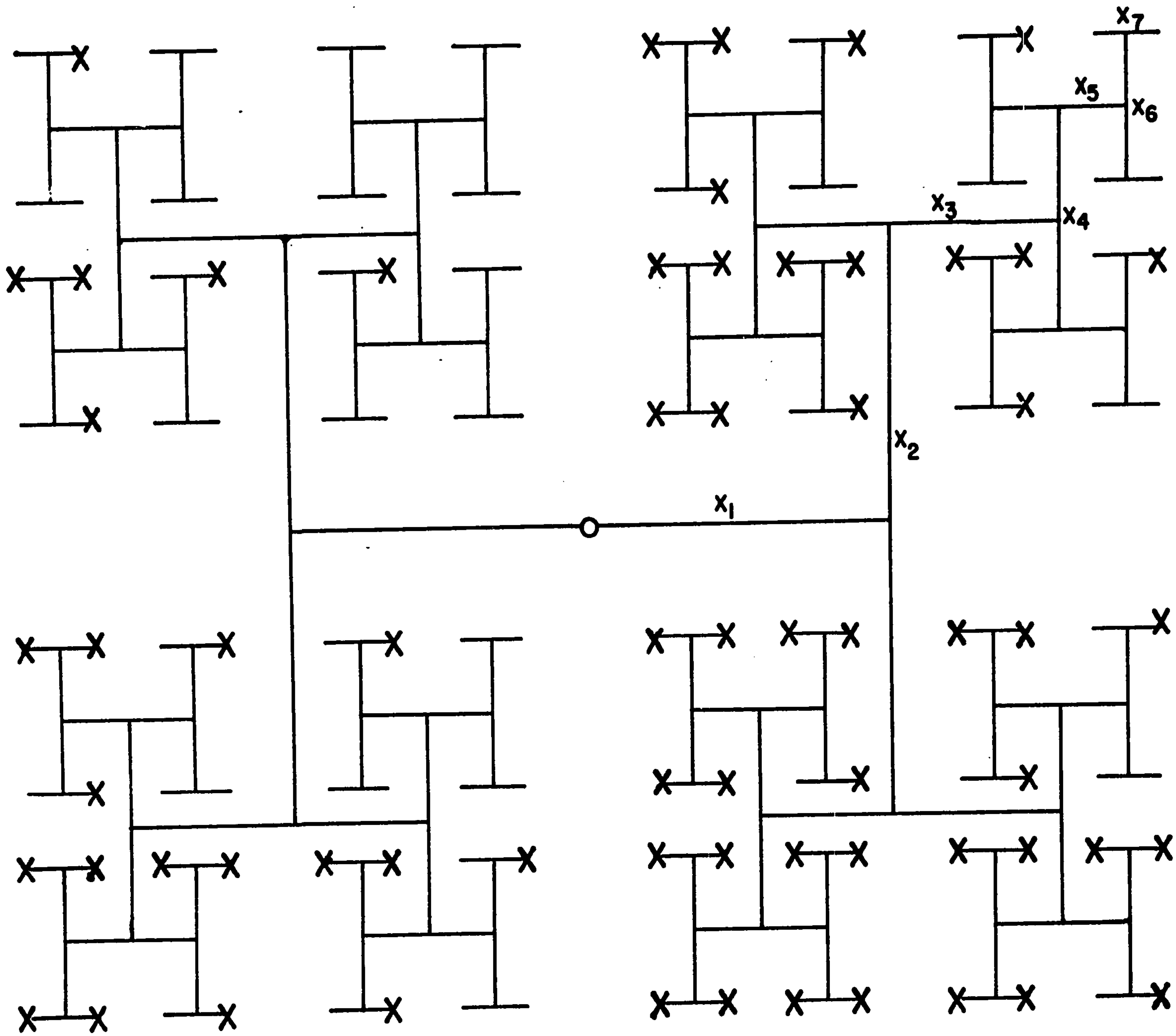


FIG.3-5(f) H-DIAGRAM FOR Z6

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

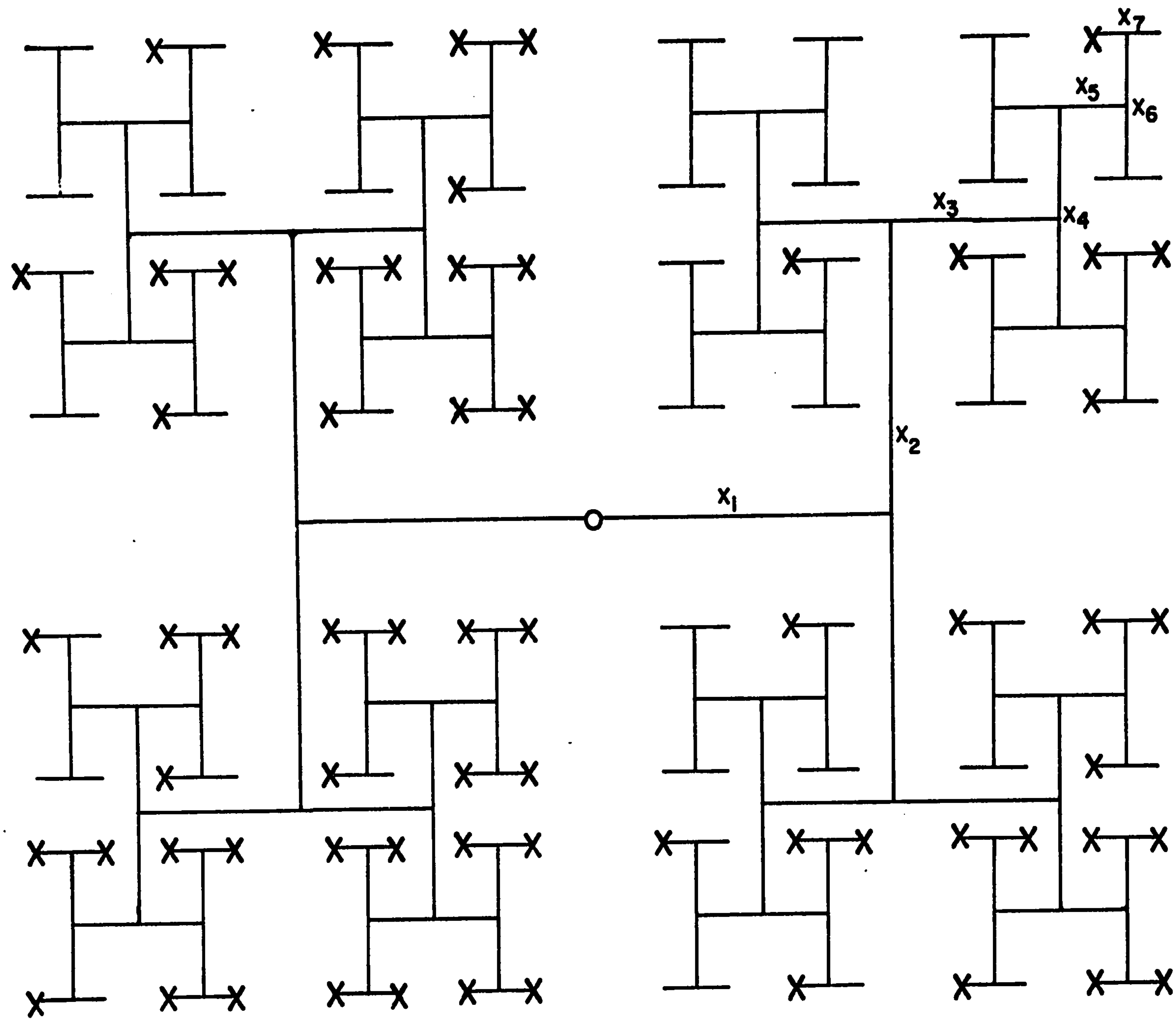


FIG.3-5(g) H-DIAGRAM FOR Z_7

NOTE: A CROSS REPRESENTS A TRUE VERTEX, OTHERWISE A FALSE VERTEX

From Fig. 3.5 the set of switching functions for N = 8 can be obtained as follows :

$$\begin{aligned}
z_0 &= x_0 \\
z_1 &= M_{3/3}(\bar{x}_1, x_2, \bar{x}_3)M_{1/4}(x_4, \bar{x}_5, x_6, \bar{x}_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \\
z_2 &= M_{3/3}(x_1, \bar{x}_2, \bar{x}_3)M_{1/4}(x_4, x_5, \bar{x}_6, \bar{x}_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \\
z_3 &= M_{3/3}(\bar{x}_1, \bar{x}_2, x_3)M_{1/4}(x_4, \bar{x}_5, \bar{x}_6, x_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \\
z_4 &= M_{3/3}(x_1, x_2, x_3)M_{1/4}(\bar{x}_4, \bar{x}_5, \bar{x}_6, \bar{x}_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \qquad (3.15) \\
z_5 &= M_{3/3}(\bar{x}_1, x_2, \bar{x}_3)M_{1/4}(\bar{x}_4, x_5, \bar{x}_6, x_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \\
z_6 &= M_{3/3}(x_1, \bar{x}_2, \bar{x}_3)M_{1/4}(\bar{x}_4, \bar{x}_5, x_6, x_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4} \\
z_7 &= M_{3/3}(\bar{x}_1, \bar{x}_2, x_3)M_{1/4}(\bar{x}_4, x_5, x_6, \bar{x}_7) + M_{2/3}M_{2/4} + M_{1/3}M_{3/4} + M_{4/4}
\end{aligned}$$

Instead of being totally symmetric as the N = 4 case, the z_i's for $i \neq 0$ are partially symmetric in two subsets, namely, the subset of the first three variables $\{x_1^*, x_2^*, x_3^*\}$ and that of the final four variables $\{x_4^*, x_5^*, x_6^*, x_7^*\}$. Also, the arguments of the two δ-functions for each individual output are the same throughout. Therefore, they are shown only once at the beginning and omitted in the rest of the expression. Other points worth mentioning are that the subscripts for the δ-functions are homogeneous, and that all these seven output functions have the same structure.

The combinatorial network for the 8-bit LWHT is shown in Fig. 3.6. With modern-day semiconductor technology this combinatorial network can be easily implemented with moderate cost. The outputs can be obtained serially by using only a single copy of the circuit and applying the different sets of inputs literals successively, i.e., the arguments of the δ-functions, one at a time. This can also be done in parallel by applying multiple copies of the circuits. This is the trade-off between the speed and the cost. Nevertheless, the LWHT for $N \leq 8$ can be obtained directly by using hardware only and without any computation.

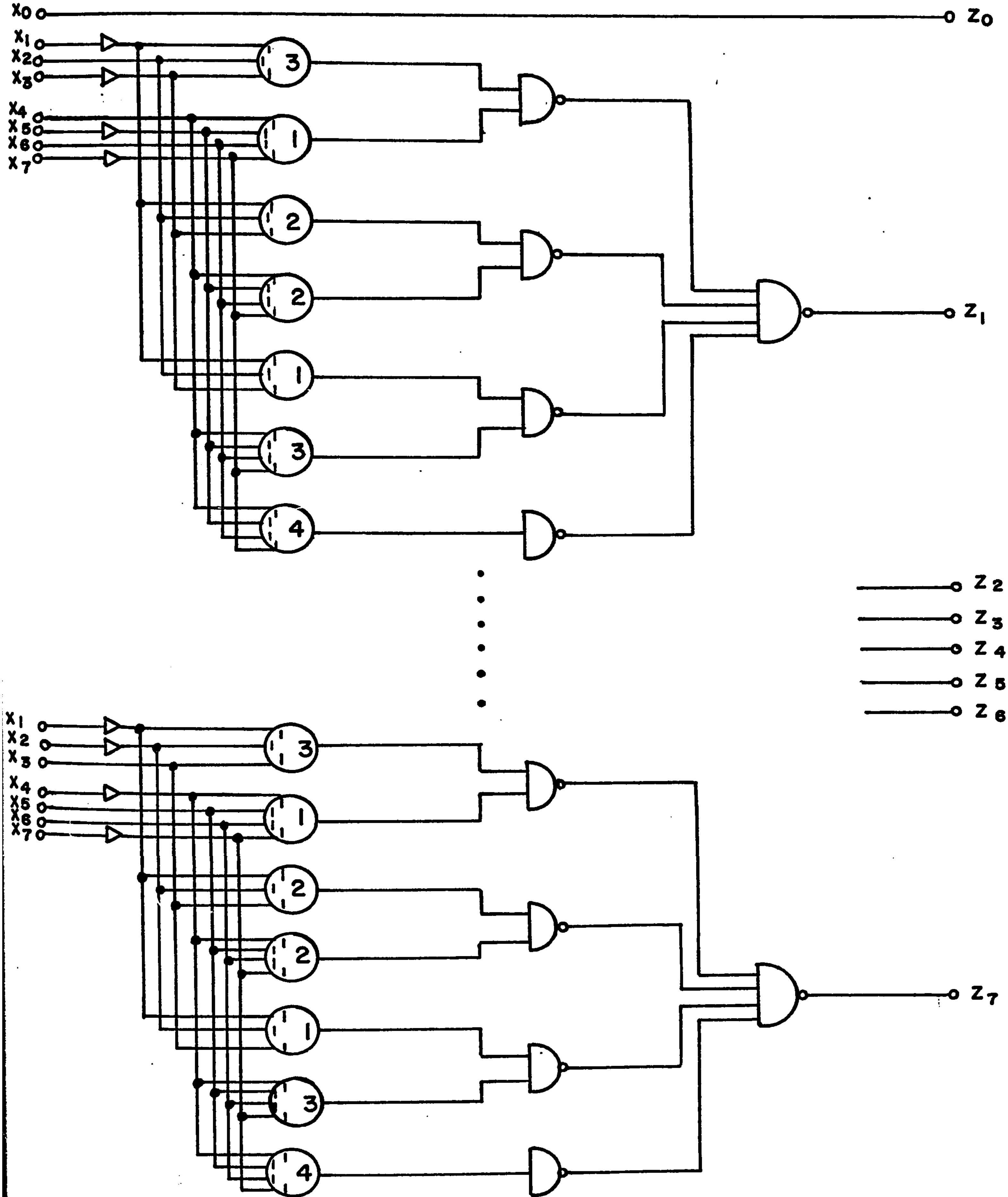


Fig. 3.6 Combinatorial Network For The 8-Bit LWHT

3.4 Analysis of the 8-Bit LWHT

First we look into the distribution of data strings that are their own transforms, i.e., LWHT (X) = X. There are none in the 2-bit case, 4 in the 4-bit case, and 16 in the 8-bit case. It is quite obvious that these data strings form full complementation pairs according to (3.9). Full complementation changes all 1-bits to 0 and 0-bits to 1. It is partial complementation[20] if the first bit is unchanged. These data strings starting with first bit equal to 1 and being their own transforms are listed in Table 3.3.

Table 3.3

Data Strings For Which LWHT (X) = X
Paired By Partial Complementation

(a) N = 4		(b) N = 8	
1 0 0 1	1 1 1 0	1 1 1 1 1 0 0 0	1 0 0 0 0 1 1 1
		1 1 1 0 1 1 0 0	1 0 0 1 0 0 1 1
		1 1 1 0 1 0 1 0	1 0 0 1 0 1 0 1
		1 1 1 0 1 0 0 0	1 0 0 1 0 1 1 1

Next, for a data string beginning with a 1 and repeating the first half bits in the last half, its LWHT will have all zeroes in the last half of the bits. In notation form, it is,

$$\text{LWHT } (1, x_1, x_2, x_3, 1, x_1, x_2, x_3) = (1, z_1, z_2, z_3, 0, 0, 0, 0) \qquad (3.16)$$

This will be shown in the Appendix I. By taking the inverse as well as the complementation of (3.16), we have,

$$\text{LWHT } (0, \bar{x}_1, \bar{x}_2, \bar{x}_3, 0, \bar{x}_1, \bar{x}_2, \bar{x}_3) = (0, \bar{z}_1, \bar{z}_2, \bar{z}_3, 1, 1, 1, 1)$$

$$\text{LWHT } (1, z_1, z_2, z_3, 0, 0, 0, 0) + (1, x_1, x_2, x_3, 1, x_1, x_2, x_3) \qquad (3.16a)$$

$$\text{LWHT } (0, \bar{z}_1, \bar{z}_2, \bar{z}_3, 1, 1, 1, 1) = (0, \bar{x}_1, \bar{x}_2, \bar{x}_3, 0, \bar{x}_1, \bar{x}_2, \bar{x}_3)$$

Also, for a data string beginning with a 1 and having the last half bits being a complementation of the first half, we have,

$$\text{LWHT}\ (1,x_1,x_2,x_3,0,\overline{x}_1,\overline{x}_2,\overline{x}_3) = (1,0,0,0,z_1,z_2,z_3,z_4) \qquad (3.17)$$

Similarly, this will be shown in the Appendix I. And we obtain equations in (3.17a) by taking the inverse and the complementation of (3.17).

$$\text{LWHT}\ (0,\overline{x}_1,\overline{x}_2,\overline{x}_3,1,x_1,x_2,x_3) = (0,1,1,1,\overline{z}_1,\overline{z}_2,\overline{z}_3,\overline{z}_4)$$

$$\text{LWHT}\ (1,0,0,0,z_1,z_2,z_3,z_4) = (1,x_1,x_2,x_3,0,\overline{x}_1,\overline{x}_2,\overline{x}_3) \qquad (3.17a)$$

$$\text{LWHT}\ (0,1,1,1,\overline{z}_1,\overline{z}_2,\overline{z}_3,\overline{z}_4) = (0,\overline{x}_1,\overline{x}_2,\overline{x}_3,1,x_1,x_2,x_3)$$

At length, we list two more set of interesting transform pairs in (3.18) and (3.19).

$$(1,x_1,x_2,x_3,0,x_1,x_2,x_3) \Longleftrightarrow (1,z_1,z_2,z_3,1,1,1,1)$$

$$(0,\overline{x}_1,\overline{x}_2,\overline{x}_3,1,\overline{x}_1,\overline{x}_2,\overline{x}_3) \Longleftrightarrow (0,\overline{z}_1,\overline{z}_2,\overline{z}_3,0,0,0,0) \qquad (3.18)$$

$$(1,x_1,x_2,x_3,1,\overline{x}_1,\overline{x}_2,\overline{x}_3) \Longleftrightarrow (1,1,1,1,z_1,z_2,z_3,z_4)$$

$$(0,\overline{x}_1,\overline{x}_2,\overline{x}_3,0,x_1,x_2,x_3) \Longleftrightarrow (0,0,0,0,z_1,z_2,z_3,z_4) \qquad (3.19)$$

3.5 Threshold Setting for the LWHT

Although at the very beginning of this chapter the threshold for the LWHT has been set, now in this section we will show that this threshold is the only one which can retain the reversibility as well as the binary nature of the transform. For a binary data string of length N, its Walsh-Hadamard transform coefficients take on values between plus and minus N. If the binary data string always starts with a 1, then its transform coefficients will lie between N and $-(\frac{N}{2} - 1)$. Table 3.4 lists the frequency count of Walsh-Hadamard transform coefficients for N = 2, 4 and 8, i.e., how often does each coefficient take on the particular value inside the bound. The first column lists the values of the coefficients and the

Table 3.4

Frequency Count Of Walsh-Hadamard Transform Coefficients

FREQUENCY COUNT OF W.H.T. COEFFS. FOR N = 2

2	1	0
1	1	1
0	0	1

FREQUENCY COUNT OF W.H.T. COEFFS. FOR N = 4

4	1	0	0	0
3	3	0	0	0
2	3	1	1	1
1	1	3	3	3
0	0	3	3	3
-1	0	1	1	1

FREQUENCY COUNT OF W.H.T. COEFFS. FOR N = 8

8	1	0	0	0	0	0	0	0
7	7	0	0	0	0	0	0	0
6	21	0	0	0	0	0	0	0
5	35	0	0	0	0	0	0	0
4	35	1	1	1	1	1	1	1
3	21	7	7	7	7	7	7	7
2	7	21	21	21	21	21	21	21
1	1	35	35	35	35	35	35	35
0	0	35	35	35	35	35	35	35
-1	0	21	21	21	21	21	21	21
-2	0	7	7	7	7	7	7	7
-3	0	1	1	1	1	1	1	1

subsequent columns are the frequency count for the corresponding coefficients.

The distribution of every coefficient is binomial. And the distributions for all the coefficients except the first one are the same. If a threshold is applied between 0 and 1 as shown in Table 3.4, the first coefficient will always pass the threshold and result in a 1, and the other coefficients will just be balance, i.e., half of them is over the threshold and half of them below. This matches what we defined at the beginning of this chapter. Conversely, any other threshold being set, imbalance will be created. Since there are exactly the same number of zeroes as the ones in each bit except the first one of the data to be counted, an imbalance means that the data will not be reversible.

3.6 Summary

In this chapter we have introduced the LWHT, its definition, its origin as well as the recent development. An exhaustive study has been provided for the $N = 2, 4, 8$ cases. For $N < 16$, the LWHT is simple, reversible, and can be realized by a set of switching functions. A combinatorial network for these switching functions is given. Some analyses of the 8-bit LWHT and on the setting of the threshold are contained. With all these being furnished we can then increase the dimension of the transform to cover the $N \geq 16$ cases. This will be the main topic of the next chapter.

CHAPTER 4

THE PIECEWISE LOGICAL WALSH-HADAMARD TRANSFORM

Recent results by Sakrison(27) and Tasto and Wintz(28) indicate that a decrease in rate-distortion by a factor of 2 or 3 is achievable from the use of two-dimensional processing encoding over one-dimensional line-by-line processing. However, performing a two-dimensional transform on an (NxN) array of image intensity samples with a relatively large and meaningful N requires fairly complicated implementation. Partition of the two-dimensional image has been proposed (7,29-31) as a method for obtaining the advantages on the rate-distortion as well as on the memory requirements, and furthermore for reducing the implementation complexity.

Since the one-dimensional logical Walsh-Hadamard transform is restricted to $N \leq 8$, the image is first broken up into a number of pieces of size (8x8) elements or less and then transform taken. In this chapter, we are going to extend the LWHT to two-dimensional case and to establish an algorithm so that it is not necessary to know in advance the exact size of the entire image. Higher dimensional LWHT will be proposed, and it will be proved that the conventional one-dimensional and two-dimensional Walsh-Hadamard transform are interchangeable under a special ordering will be included as well.

4.1 The Two-Dimensional LWHT

In the last chapter a combinatorial network for obtaining the LWHT for $N \leq 8$ was given. It was obtained from the three sets of switching functions given in (3.11), (3.14) and (3.15). Each switching function is nothing but a mapping from an n-dimensional binary space to a one-dimensional binary space, where n is the number of input variables. In our case, $n = N$, and each set in (3.11), (3.14) and (3.15) has exactly N functions. Therefore, either of these three sets of switching functions defines a mapping from an N-dimensional binary space onto itself. Moreover, the mapping is one-to-one, i.e., for each input combination the output combination is unique. In other words, for $N \leq 8$ each binary data string has a unique LWHT.

As mentioned before this kind of relationship breaks down for $N \geq 16$. Parkyn Jr. and Cash have given a nice analysis on this for $N = 16$[24]. However, if we first arrange a binary data string of length greater than or equal to 16 to a two-dimensional binary matrix, and next define a two-dimensional LWHT for it, then we can retain whatever we have obtained in the one-dimensional case - the instantaneous output without computation and the reversibility of the transform. In this section we confine ourselves to the two-dimensional case. Nevertheless, the same kind of reasoning can be extended to any higher dimensions.

Now, the definitions of the mapping function f_M shall be extended to include that both its domain and its range can be a matrix, i.e., given an (NxL) matrix $\underline{A}$, $f_M(\underline{A})$ is defined as:

$$f_M(\underline{A}) = \begin{pmatrix} f_M(a_{11}) & f_M(a_{12}) & \cdots & f_M(a_{1L}) \\ f_M(a_{21}) & f_M(a_{22}) & \cdots & f_M(a_{2L}) \\ \vdots & \vdots & \ddots & \vdots \\ f_M(a_{N1}) & f_M(a_{N2}) & \cdots & f_M(a_{NL}) \end{pmatrix} \qquad (4.1)$$

For N, $L \leq 8$ and both of them being a power of 2, the two-dimensional

LWHT will be defined as :

$$\underline{Z} = \text{2-LWHT}(\underline{X}) = f_M\ \{\underline{H}_N f_M(\underline{X}\,\underline{H}_L)\} \qquad (4.2)$$

and

$$\underline{X} = \text{2-LWHT}^{-1}(\underline{Z}) = f_M\ \{f_M(\underline{H}_N\,\underline{Z})\,\underline{H}_L\} \qquad (4.3)$$

where both $\underline{X}$ and $\underline{Z}$ are (NxL) binary matrices, and $\underline{H}$ is the Hadamard matrix, also $x_{ij} = 1$ for $i = 1$ or $j = 1$. For x_{1j} or x_{i1} equals to zero, the complementation defined in (3.9) for the one-dimensional case shall be applied to the corresponding column or row respectively.

By the definition of f_M as in (3.5) and (4.1), for $\underline{A}$ and a being binary we have,

$$f_M(\underline{A}) = \underline{A} \quad \text{and} \quad f_M(a) = a$$

Therefore, for N=1, $\underline{X}$ becomes a row vector and its LWHT will be

$$\begin{aligned} \underline{Z} &= f_M\ \{\underline{H}_1\ f_M(\underline{X}\ \underline{H}_L)\} \\ &= f_M\ \{f_M(\underline{X}\ \underline{H}_L)\} \\ &= f_M(\underline{X}\ \underline{H}_L) \end{aligned}$$

and

$$\underline{X} = f_M(\underline{Z}\ \underline{H}_L)$$

Similarly, for L = 1, $\underline{X}$ becomes a column vector and its LWHT will be

$$\underline{Z} = f_M(\underline{H}_N\ \underline{X})$$

and

$$\underline{X} = f_M(\underline{H}_N\ \underline{Z})$$

This is exactly the same as the one-dimensional case defined in (3.6) but the data and its transform are now in matrix or vector form. Actually, the two-dimensional LWHT defined is nothing but one-dimensional LWHT applied twice; first rowwise and then columnwise. The inverse 2-D LWHT is just the reverse, first columnwise and then rowwise.

Theorem 4. 1 The two-dimensional LWHT defined as in (4.2) is unique, for N, L $\le$ 8.

Proof: Assume that $\underline{X}_1$, $\underline{X}_2$, $\underline{Y}_1$, $\underline{Y}_2$, $\underline{Z}_1$, $\underline{Z}_2$ are (NxL) matrices for N, L $\le$ 8, and

$$\underline{Z}_i = \text{2-LWHT}\ (\underline{X}_i) \qquad \text{for } i = 1, 2$$

$$\underline{Y}_i = f_M(\underline{H}_N\ \underline{X}_i) \qquad \text{for } i = 1, 2$$

$$\underline{Z}_i = f_M(\underline{Y}_i\ \underline{H}_L) \qquad \text{for } i = 1, 2$$

We have already shown that for the one-dimensional case the LWHT is unique, or in mathematical forms :

$$\underline{Z}_1 = \underline{Z}_2 \qquad \text{iff } \underline{Y}_1 = \underline{Y}_2$$

and $\underline{Y}_1 = \underline{Y}_2$ iff $\underline{X}_1 = \underline{X}_2$

Therefore, $\underline{Z}_1 = \underline{Z}_2$ iff $\underline{X}_1 = \underline{X}_2$

Q.E.D.

A combinatorial network for the 2-D LWHT of an (8x8) matrix is shown in Fig. 4.1. However, 64 is the maximum length for the two-dimensional case. For binary data string of N > 64 higher dimensional LWHT is needed.

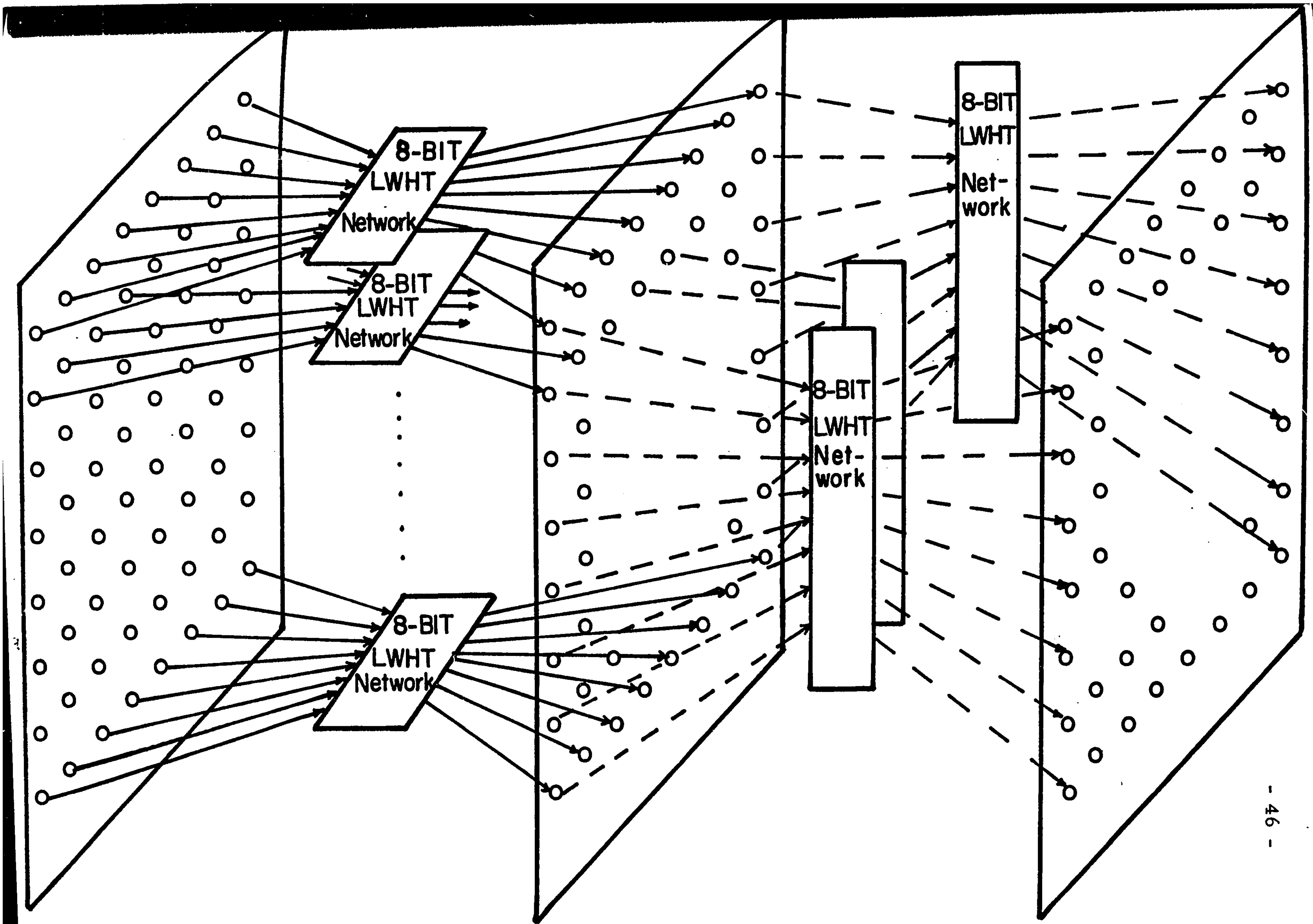


Fig. 4.1 Combinatorial Network For 2-D LWHT

Example 4.1 Find the 2-D LWHT of an (8x8) binary matrix

Data $\underline{X} = \begin{bmatrix} 1&0&1&0&1&0&1&0 \\ 1&0&0&1&1&0&0&1 \\ 1&1&1&1&0&0&0&0 \\ 1&1&0&0&0&0&1&1 \\ 1&0&1&0&0&1&1&0 \\ 1&0&1&0&1&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&0&1&1&1&0&0&0 \end{bmatrix}$

Apply 8-bit LWHT rowwise $\begin{bmatrix} 1&1&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 1&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0 \\ 1&1&0&0&0&1&0&1 \\ 1&1&1&1&0&1&0&0 \\ 0&1&0&1&1&1&1&1 \\ 0&1&0&1&1&0&0&0 \end{bmatrix}$

Apply 8-bit LWHT columnwise $\underline{Z} = 2\text{-LWHT}(\underline{X}) = \begin{bmatrix} 1&1&0&0&0&0&0&0 \\ 0&1&0&0&1&1&1&1 \\ 1&1&1&1&0&1&0&1 \\ 0&1&0&0&0&0&1&1 \\ 1&0&0&0&0&0&1&0 \\ 0&1&1&1&1&0&0&0 \\ 0&1&0&1&1&0&1&1 \\ 0&1&1&1&0&1&1&1 \end{bmatrix}$

Example 4.2 Find the inverse 2-D LWHT of $\underline{Z}$ given in Example 1. $\underline{X}$ can be obtained simply reversing the order of the procedure in Example 1, i.e., work from the bottom upwards.

4.2 A Theorem Relating Two-Dimensional With One-Dimensional Walsh-Hadamard Transforms

In image processing, either two-dimensional processing encoding or one-dimensional line-by-line processing can be used. Practically speaking, the latter is more desirable since it is consistent with conventional scanning methods thus resulting in considerable reduction in implementation complexity and storage requirements. However, as mentioned before, the former gives a better performance as far as the rate-distortion is concerned. For Walsh-Hadamard transforms applying the one-dimensional one is actually equivalent to taking the two-dimensional transform. This is a well-known fact for most of the researchers in this field, but a rigorous mathematical proof is not existing in the literature. The following theorem will show the equivalence, and we give this teorem a proof.

Theorem 4.2 Assume that the matrix $\underline{g}$ shown below represents the values of a two-dimensional image or pattern,

$$\underline{g} = [\underline{g}_1, \underline{g}_2, \ldots\ldots, \underline{g}_L] = \begin{bmatrix} g_{11} & g_{12} & \cdots\cdots & g_{1L} \\ g_{21} & g_{22} & \cdots\cdots & g_{2L} \\ \vdots & \vdots & & \vdots \\ g_{N1} & g_{N2} & \cdots\cdots & g_{NL} \end{bmatrix}$$

where all g_{ij}'s are real numbers, $\underline{g}_i$'s are column vectors and N, L are powers of two. Then, the two-dimensional Walsh-Hadamard transform will produce the same coefficients as the one-dimensional one applying to

$$\underline{f} = \begin{bmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \cdot \\ \cdot \\ \cdot \\ \underline{g}_L \end{bmatrix}$$

$$= [\, g_{11}, g_{21}, \ldots, g_{N1};\ g_{12}, g_{22}, \ldots, g_{N2}; \ldots ;\ g_{1L}, \ldots, g_{NL}]^T$$

where the superscript T denotes the transpose of a matrix or a vector.

Proof: Let $\underline{G} = \underline{H}_N \, \underline{g} \, \underline{H}_L$ be an (NxL) matrix where $N = 2^n$, $L = 2^\ell$ and n, ℓ are integers. And let

$$\underline{F} = \underline{H}_{(NL)} \, \underline{f}$$

$$= \begin{bmatrix} \underline{F}_1 \\ \underline{F}_2 \\ \vdots \\ \underline{F}_L \end{bmatrix}$$

be a column vector of (NL) elements where each $\underline{F}_i$ is a column vector of N elements. Hence, if

$$\underline{F}' = [\, \underline{F}_1, \underline{F}_2, \ldots\ldots \underline{F}_L] \,,$$

Then $\underline{F}'$ is an (NxL) matrix. We will show that

$$\underline{F}' = \underline{G}$$

Since $\underline{H}_L = [h_{ij}]_{LxL}$ and $h_{ij} \in (-1, 1)$ for i, j equal to 1 to L,

$$\underline{H}_{(NL)} = \underline{H}_N \otimes \underline{H}_L = \underline{H}_L \otimes \underline{H}_N = \begin{bmatrix} h_{11}\underline{H}_N & h_{12}\underline{H}_N & \cdots & h_{1L}\underline{H}_N \\ h_{21}\underline{H}_N & h_{22}\underline{H}_N & \cdots & h_{2L}\underline{H}_N \\ \vdots & \vdots & \ddots & \vdots \\ h_{L1}\underline{H}_N & h_{L2}\underline{H}_N & \cdots & h_{LL}\underline{H}_N \end{bmatrix}$$

where $\otimes$ denotes the Kronecker-product operation [35].

Then,

$$\underline{F} = \underline{H}_{(NL)}\,\underline{f} = \begin{bmatrix} h_{11}\underline{H}_N & h_{12}\underline{H}_N & \cdots & h_{1L}\underline{H}_N \\ h_{21}\underline{H}_N & h_{22}\underline{H}_N & \cdots & h_{2L}\underline{H}_N \\ \vdots & \vdots & \ddots & \vdots \\ h_{L1}\underline{H}_N & h_{L2}\underline{H}_N & \cdots & h_{LL}\underline{H}_N \end{bmatrix} \begin{bmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \vdots \\ \underline{g}_L \end{bmatrix}$$

$$= \left[\sum_{i=1}^{L} h_{1i}\underline{H}_N\underline{g}_i,\ \sum_{i=1}^{L} h_{2i}\underline{H}_N\underline{g}_i,\ \ldots\ldots,\ \sum_{i=1}^{L} h_{Li}\underline{H}_N\underline{g}_i \right]^T$$

Therefore,

$$\underline{F}_j = \sum_{i=1}^{L} h_{ji}\,\underline{H}_N\,\underline{g}_i \qquad \text{for } j = 1, 2, \ldots. L,$$

and

$$\underline{F}' = [\sum_{i=1}^{L} h_{1i}\underline{H}_N \underline{g}_i, \sum_{i=1}^{L} h_{2i}\underline{H}_N \underline{g}_i, \ldots\ldots, \sum_{i=1}^{L} h_{Li}\underline{H}_N \underline{g}_i]$$

$$= [\underline{H}_N \sum_{i=1}^{L} h_{1i}\underline{g}_i, \underline{H}_N \sum_{i=1}^{L} h_{2i}\underline{g}_i, \ldots\ldots, \underline{H}_N \sum_{i=1}^{L} h_{Li}\underline{g}_i]$$

$$= \underline{H}_N [\sum_{i=1}^{L} h_{1i}\underline{g}_i, \sum_{i=1}^{L} h_{2i}\underline{g}_i, \ldots\ldots, \sum_{i=1}^{L} h_{Li}\underline{g}_i]$$

$$= \underline{H}_N \ \underline{A}$$

where the element in the kth row and the jth column of $\underline{A}$ will be

$$a_{kj} = \sum_{i=1}^{L} h_{ji} \ g_{ki} = \sum_{i=1}^{L} g_{ki} \ h_{ij}$$

This is due to the fact that the Hadamard matrix is symmetric, i.e., $h_{ji} = h_{ij}$ for all i's and j's.

Hence,

$$\underline{A} = \underline{g} \ \underline{H}_L$$

$$\underline{F}' = \underline{H}_N \ \underline{g} \ \underline{H}_L$$

$$= \underline{G}$$

Q.E.D.

4.3 The Piecewise Logical Walsh-Hadamard Transform

In Section 4.1 we have introduced the two-dimensional LWHT. For a two-dimensional LWHT, the length of the data string is restricted to be less than 128 and has to be a power of two. The first restriction can be lifted by introducing higher dimensional LWHT, which will be discussed in the next section. The second one can also be extended to some other cases, i.e., instead of being a power of two between 8 and 64 the length of the data string can be any even numbers. However, to be of practical use the length will be confined to a multiple of eight between 8 and 64.

Consider a binary data string of length $8 \leq N \leq 64$, where N is also a multiple of 8. This data string will first be partitioned into n blocks of 8 bits each. Obviously, $n = N/8$. Each block will then be applied to the LWHT defined in (3.6) and (3.9). Since LWHT preserves the length of the original data, the transform thus has the same length N as before. An octal number of two octal digits can be used to represent the ordering of the data string. Namely, the eight elements in the first block will be assigned the number 00, 01 to 07, with the second block the numbers 10, 11 to 17, and in the last block the numbers (n-1)0, (n-1)1 to (n-1)7. If we give a weight of n to the second octal digit and a weight of one to the first one, then the weighted values of these octal numbers will lie between zero and N-1 without any repetition. Now, a permutation can be performed on the transform coefficients according to the ascending order of the weighted value of its corresponding octal number. Apply once again the 8-bit LWHT to each block of the transform after permutation. This will be called the piecewise logical Walsh-Hadamard transform (abbreviated PLWHT) of the original data.

Actually, the PLWHT is a two-dimensional transform. The one-dimensional 8-bit LWHT has to be applied before and after the permutation. The only difference between the PLWHT and the two-dimensional LWHT lies in that the n in PLWHT can be any integer while in the two-dimensional LWHT case it has to be a power of two. Three examples for obtaining the PLWHT are given as follows.

Example 4.3 N = 16, n = 16/8 = 2

Data	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0
8-Bit LWHT	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	1
Octal No. Ordering	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
Weighted Values	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13	15
Permutation Of LWHT	0	1	0	1	0	1	1	1	0	0	1	0	0	0	1	1
PLWHT	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0

Example 4.4 N = 24, n = 24/8 = 3

Data	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1
8-Bit LWHT	1	0	0	0	0	1	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	0
Octal No. Ordering	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17	20	21	22	23	24	25	26	27
Weighted Values	0	3	6	9	12	15	18	21	1	4	7	10	13	16	19	22	2	5	8	11	14	17	20	23
Permutation Of LWHT	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	1	1	1	0
PLWHT	1	1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	1

Example 4.5 N = 16, n = 2

Data	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0
8-Bit LWHT	0	1	0	1	0	1	1	1	0	0	1	0	0	0	1	1
Octal No. Ordering	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
Weighted Values	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13	15
Permutation Of LWHT	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	1
PLWHT	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0

For n being a power of two, the PLWHT can be obtained from the two-dimensional LWHT by simply transposing the resultant matrix of the LWHT and arranging the matrix row by row into a single array. In Example 3.5, we showed the inverse transform of the PLWHT of Example 3.3. It turns out to be that the PLWHT is its own inverse. Thus, the original data can be recovered by simply applying the PLWHT twice.

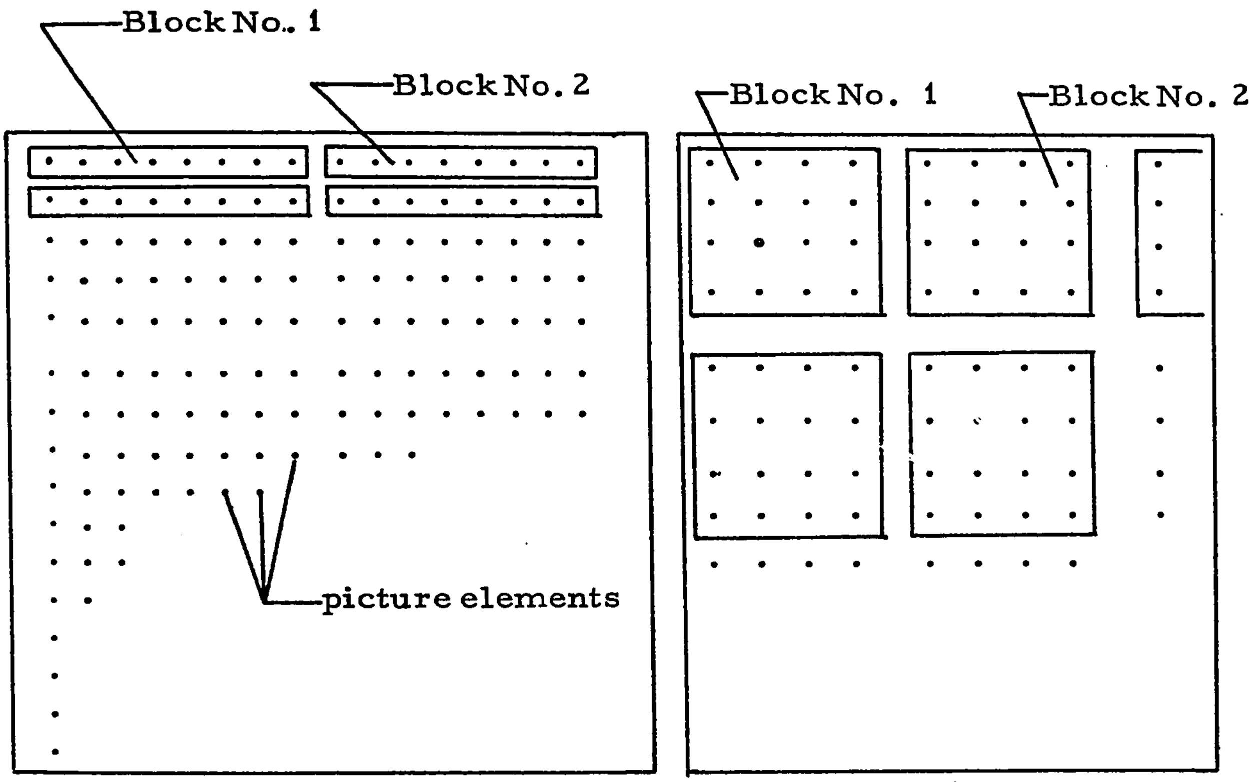
The operations involved in the PLWHT is nothing but twice the 8-bit LWHT's and a permutation in between. Since the 8-bit LWHT can be done by using a combinatorial network and the rule for permutation is quite simple, the speed for obtaining PLWHT will be as fast as the two-dimensional LWHT. Besides, the length of the data string or the overall size of the image is not necessary to be known in advance for performing the PLWHT. The number of blocks - n, can be easily counted while the input data is entering the register for the first 8-bit LWHT.

4.4 The Logical Transform For A Two-Dimensional Image Of Larger Size

With the materials introduced in sections 4.1 and 4.3, namely, the two-dimensional logical Walsh-Hadamard and the piecewise logical Walsh-Hadamard transforms, we are now in the position to look into the practical application of the logical transform in image processing. Previously, we confined ourselves to the data string of length less than or equal to 64, or an image with elements no more than 64. Actually, both the two-dimensional LWHT and the PLWHT are defined within this bound. However, practically speaking, the number of picture elements in an image is always exceeding this bound. The size of an image is usually 16x16 and up. Therefore, we have to partition the image into blocks of small units. The recommended size of this kind of unit will be 64 in the two-dimensional LWHT case and 8 in the PLWHT case. The other restriction such as the length or the number of picture elements has to be a power of two or a multiple of eight will not have much effect. Because of the broad use of the computer in digital image processing the size of the image is usually to be a power of two. For a number larger than eight, this implies that it is a multiple of eight.

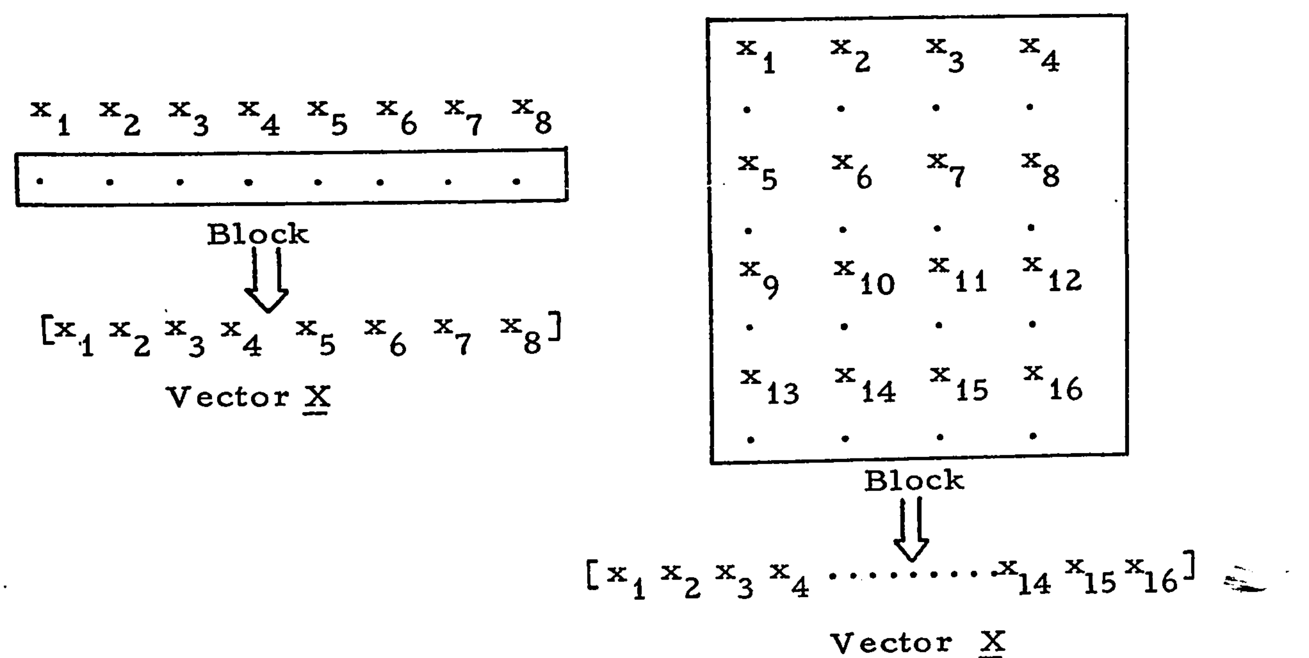
The partition of an image into blocks of picture elements is best shown by illustrations. Fig. 4.2(a) shows the arrangement for the PLWHT case and Fig. 4.2(b) shows the arrangement for the two-dimensional LWHT case. In Fig. 4.2(b), because of the size limit of this sheet we are showing a block of (4x4) elements instead of the often mentioned size of (8x8). In Fig. 4.3 we also show the way we used in this thesis to arrange the picture element in the blocks into vectors.

Comparing the two schemes shown in the Fig. 4.2(a) and (b), we will easily notice that the PLWHT needs only one-dimensional combinatorial network while in the other case two-dimensional combinatorial network is required. Furthermore, the first scheme is more compatible to the current scanning devices. Therefore, as far as the implementation complexity concerned, the PLWHT will be superior to the two-dimensional LWHT.



(a) (b)

Fig. 4.2 Partition of images into blocks for (a) PLWHT and (b) 2-D LWHT



(a) (b)

Fig. 4.3 Arrangement of picture elements into vectors for (a) one-dimensional and (b) two-dimensional blocks

However, if we compare them in terms of rate-distortion, speed and intermediate memory requirements, then we will find just the opposite, i.e., the two-dimensional LWHT is better than the PLWHT in all respects. Having mentioned in the beginning of this chapter, the rate-distortion can be decreased by a factor of 2 or 3 in favor of the two-dimensional scheme[27,28]. Secondly, for two images containing exactly the same number of picture elements, one of them will be partitioned into N_1 blocks according to the scheme (a) while the other one is partitioned into N_2 blocks according to the scheme (b). In general, N_1 is greater than N_2. If the time delay introduced by the conventional logic gate or the majority gate is negligible, then the time required to obtain the logical transform for each block, either in (a) or in (b), will be the same. Hence, for the overall image, the transform can be obtained faster by a factor N_1/N_2 by using scheme (b) -- the two-dimensional LWHT, than using scheme (a) -- the PLWHT. Also, because of partitioning, the computer has to assign a portion of its memory to keep track of the ordering of the blocks. The larger the number of blocks, the more intermediate memory is required. In addition to this, for the PLWHT there is a permutation involved which also requires a certain amount of memories to take care of this matter.

Hence, there will be a trade-off between the implementation complexity and the performance. But due to the inherent limitation of the logical transform on the size in one direction, in other words, the maximum number of elements in a one-dimensional block is 8, 64 in a two-dimensional block, 512 in a three-dimensional block, and so on, the two-dimensional LWHT in scheme (b) will turn out to be the better choice.

A problem identified with partitioning is edge effects. Because just like the Fourier transform, the Walsh-Hadamard transform is equivalent to an expansion of Walsh functions, differences in brightness between opposite edges act like a discontinuity requiring a substantial number of spectral components to represent it. Errors in these discontinuities appear as newly introduced lines, or contours along the boundaries of the blocks in the recovered picture. A possible solution proposed[32] was to add an extra row and column in order to permit interpolation. Introduction of the extra row and column reduces the sharpness of the discontinuities due to brightness differences between opposite edges of the two-dimensional blocks. In order to accomodate this extra row and column the image has to be partitioned into blocks of size one less than the block size used for the combinatorial network in each dimension. For instance, if the combinatorial network designed is of size (8x8), then the image should be partitioned into blocks of (7x7) each. The block size will be consistent with the size of the network after interpolating the neighbouring rows and columns between the blocks and annexing the extra row and the extra column to the end of the preceeding block.

4.5 The Higher Dimensional Case

In order to have a unique logical transform for a binary data string of longer length we have to raise the dimension of the transform. For instance, for data string of length greater than eight we have to use two-dimensional LWHT instead of the one-dimensional LWHT. Similarly, for a data string of length greater than 64, a three-dimensional LWHT is required. However, in doing so we have first to define a three-dimensional binary array and the multiplication for such an array to the Hadamard matrices. Furthermore, for a dimension higher than three this kind of extension will be more and more complicated and very hard to grasp. Even through this kind of concept might eventually work out it will not have much practical meaning.

In section 4.3 we introduced the piecewise logical Walsh-Hadamard transform. By applying one-dimensional LWHT twice and performing a permutation in between, the PLWHT obtained had a similar effect as the two-dimensional LWHT. This is to say that a unique high dimensional LWHT can be achieved by using lower dimensional LWHT more than once and performing permutations in between. As suggested in the last section there are two different ways -- one using one-dimensional building blocks and the other using two-dimensional building blocks. A comparison of these two different methods was also considered in that particular section . Next, we are going to consider the case of an image consisting of (16x16) elements. This size of the image is specially chosen for the ease of illustration.

In Fig. 4.4 the building block is one-dimensional and the block size is eight. Since there are altogether 256 elements, 32 blocks are required. Octal numbers are used throughout the figure. The number on the top of each block is the block number of that corresponding block and it is invariant of the permutation. The three octal digit numbers inside the block denote the position or the address of the elements with reference to the original layout. For example, the second element of the third block of the original image will have an address 032 (Note: The blocks as well as the elements inside the block will start with the zeroth order) after the first permutation this element will move to the third position of the second block, and after

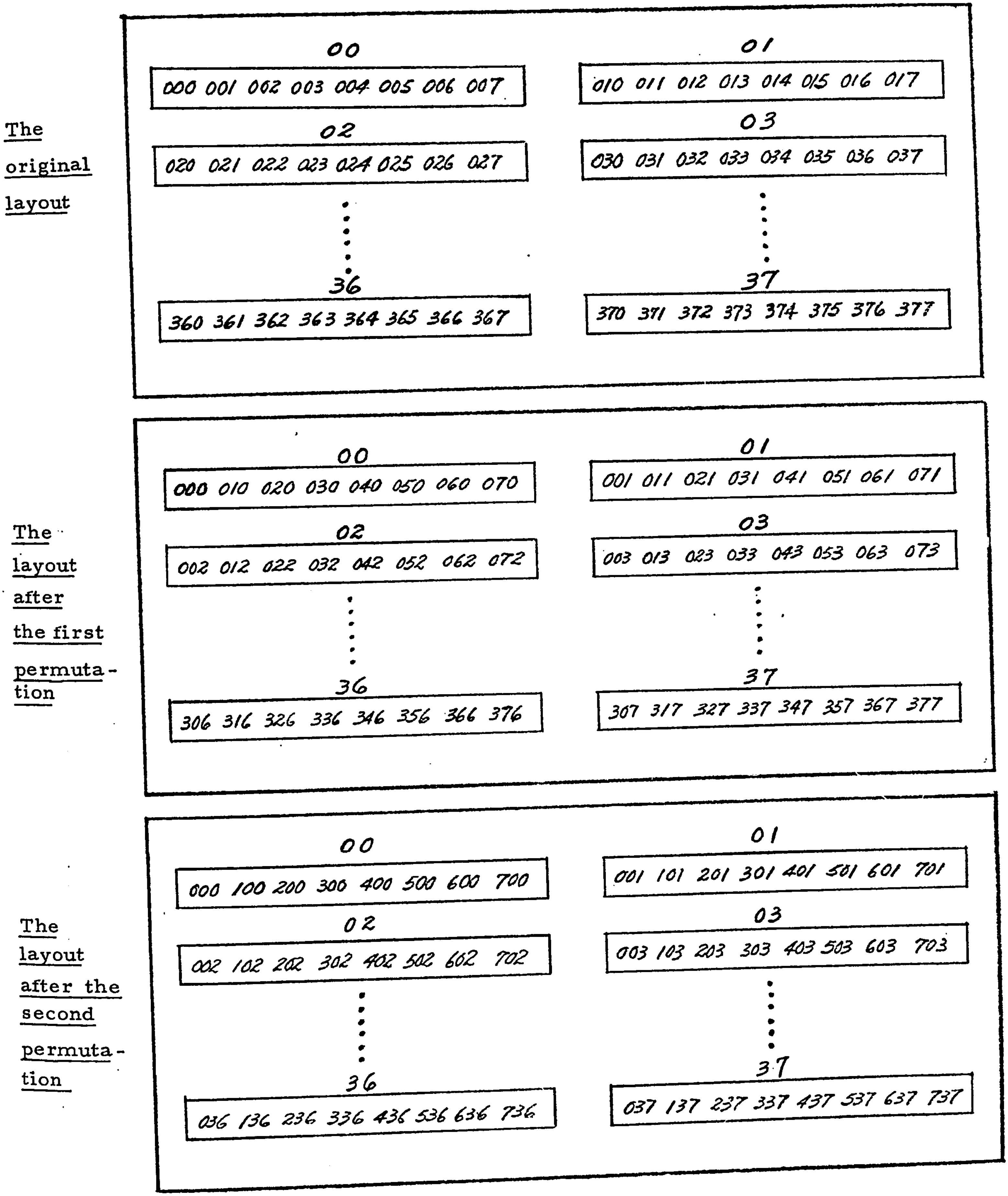


Fig. 4.4 Arrangement of picture elements for a 3-D LWHT by using 1-D building blocks.

the second permutation this element will move further to the zeroth position of the twenty-sixth (32 in octal representation) block. At the original layout of the image the first two octal digits of the three digit number denote the block number in octal representation and the last digit denotes the location within the block.

In Fig. 4.5 the building block is two-dimensional and the block size is (4x4), i.e., 16 elements in each block. There are altogether 16 blocks. Instead of using the octal numbers as in Fig. 4.4 we use hexadecimal representation, i.e., in addition to the decimal number using the alphabets A to F to represent 10 to 15 respectively. The hexadecimal digits are necessary to denote uniquely, the element location the first digit for the block number and the second digit for the position within the block. As in Fig. 4.4 the location number shown in the block is always with reference to the original layout.

Since the logical transform retains exactly the same size as its original data. No matter if the transform is one-dimensional or two-dimensional, it is an 'in place' operation. This is to say that the transform will occupy the same location as its original data after the transformation, or the data is replaced by its transform in the same location. Therefore, the LWHT will not be shown in Figs. 4.4 and 4.5. But it has to be understood that in Fig. 4.4 a one-dimensional LWHT shall be performed before and after each permutation while in Fig. 4.5 a two-dimensional LWHT shall be performed instead.

The permutation rule we chose is a very simple one. In Fig. 4.4

The original layout

The layout afters permutation

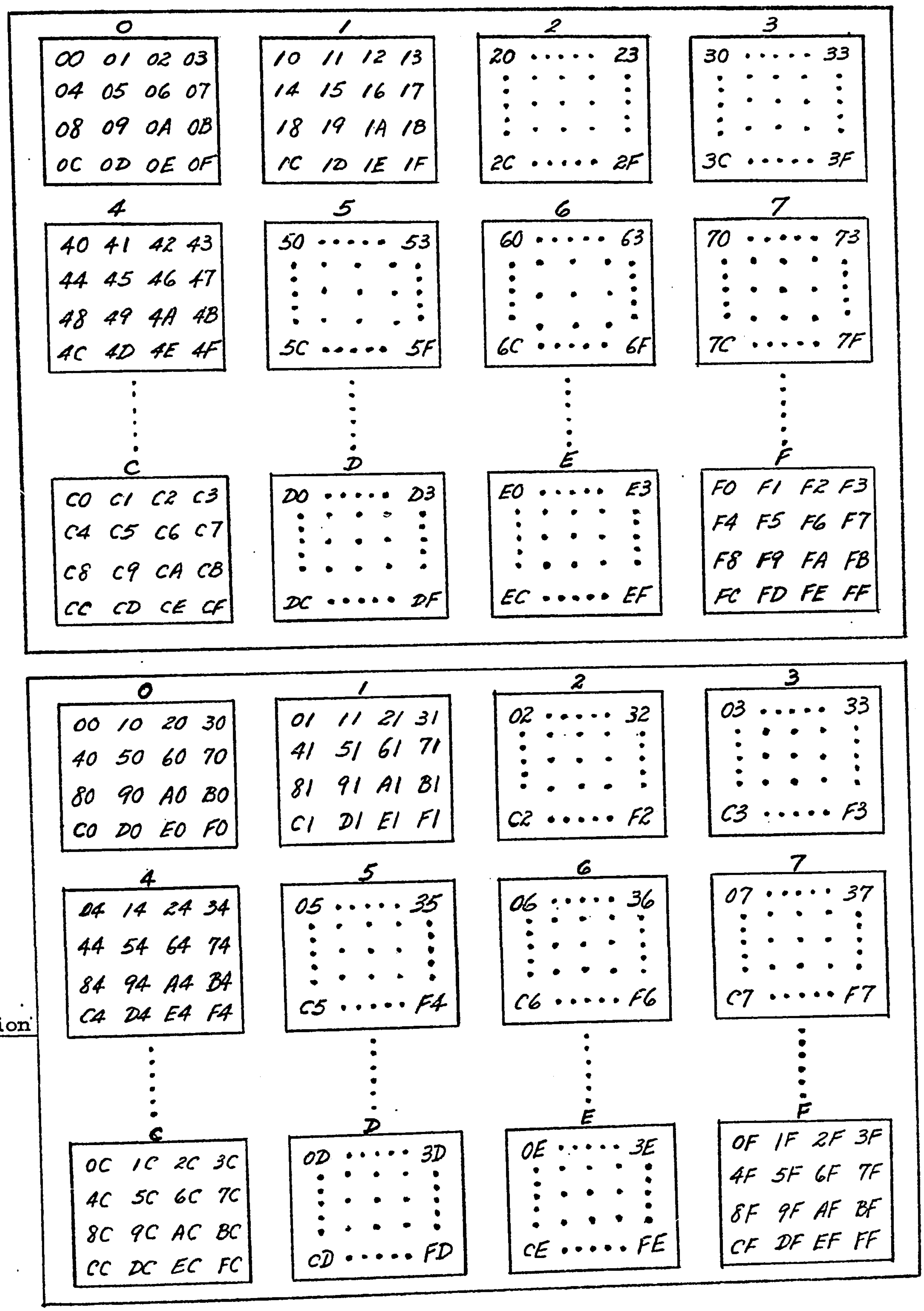


Fig. 4.5 Arrangement of picture elements for a 3-D LWHT by using 2-D building blocks.

the first two digits are fixed for each block in the original layout, the first and the third digits are fixed for each block in the layout after the first permutation, and the last two digits are fixed for each block after the second permutation. In Fig. 4.5 the first digit is fixed for each block in the original layout and the last digit is fixed for each block after the permutation. A permutation rule for the general case can be found in the same fashion.

For the case as in Fig. 4.4 the one-dimensional LWHT will be performed three times. And for the case as in Fig. 4.5 the two-dimensional LWHT will be performed twice. As we noticed before the two-dimensional LWHT is equivalent to applying one-dimensional LWHT twice with one permutation in between. Therefore, for both these two cases as in Figs. 4.4 and 4.5 the LWHT obtained will be a three-dimensional one. To extend this kind of reasoning, an n-dimensional LWHT for any number of n can be obtained by either applying the one-dimensional LWHT n times and a permutation in between any of the two operations of the LWHT, or applying the two-dimensional LWHT (n-1) times and a permutation in between any two operations of the LWHT.

4.6 Summary

In this chapter we extended the LWHT to the cases that the length of a data string or the number of elements in an image is considerably large. There were two ways to achieve this --- one way is to increase the dimension of the transform and the other way is to introduce a permutation and repeat the LWHT several times. Both of these were tried and worked out in some detail. The dimension was only increased to two and we were convinced that higher dimensional transform in the structural sense would not have much practical meaning. We did obtain the higher dimensional transform by using permutations. Partition of an image was also introduced and its effects were discussed.

CHAPTER 5

THE FAST WALSH-HADAMARD TRANSFORM BY USING H DIAGRAMS

As stated in the first chapter the Walsh-Hadamard transform has been recently applied in communication, namely in the transmission of digital images[17], and also in pattern recognition for image processing and feature extraction[16,33,34]. The Walsh-Hadamard transform replaces the orthogonal system of sine-cosine functions of the Fourier transform by another system of orthogonal functions known as Walsh functions. Each row (or column) of the Hadamard matrix is a Walsh function defined in the interval $(-\frac{1}{2}, \frac{1}{2})$[17]. The elements of a Hadamard matrix take on values of plus and minus one only. This leads to simple implementation with semiconductor technology, and simplifies the analysis by digital computer. Furthermore, fast Walsh-Hadamard transform can be used by factoring the Hadamard matrix[35,36]. This in turn reduces the number of required operations and provides a faster computer implementation.

In this chapter we introduce a modified and simple factorization method of the Hadamard matrix, and interprets the factorization in terms of operations on an H diagram, a geometric model originally used for the synthesis of combinatorial logic circuits[19]. This method not only maintains the advantages of the fast Walsh-Hadamard transform by reducing the number of storage elements and the number of arithmetic operations, it also has the advantage of providing a visual aid for the computational procedure.

5.1 The Hadamard Matrix And The Walsh-Hadamard Transform

The Hadamard matrix is a symmetric square matrix of elements plus one and minus one. Its rows (and columns) are mutually orthogonal. Let $\underline{H}_N$ be a Hadamard matrix of order N, then,

$$\underline{H}_N \underline{H}_N = N \underline{I}_N \qquad (5.1)$$

where $\underline{I}_N$ is the identity matrix of order N.

For $N = 2^n$, where n is an integer, a Hadamard matrix can be constructed by recursively applying the Kronecker-product. Let $\underline{H}_N$ be a Hadamard matrix of order $N = 2^n$, then the matrix

$$\begin{aligned} \underline{H}_{2N} &= \begin{bmatrix} \underline{H}_N & \underline{H}_N \\ \underline{H}_N & -\underline{H}_N \end{bmatrix} \\ &= \underline{H}_2 \otimes \underline{H}_N = \underline{H}_2 \otimes \underbrace{\underline{H}_2 \otimes \cdots \otimes \underline{H}_2}_{n} \\ &= \underline{H}_2^{[n+1]} \end{aligned} \qquad (5.2)$$

$$\text{with } \underline{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (5.3)$$

where the operation $\otimes$ is a Kronecker-product and the bracketed exponent means that the Kronecker-product is performed (n + 1) times upon H_2.

The order of a Hadamard matrix need not be a power of two. It has been conjectured that a Hadamard matrix of order N exists for all N equal to a multiple of four[31].

In this thesis, however, we will restrict ourselves to those Hadamard matrices with orders equal to powers of two.

The Hadamard matrix can be used to obtain the one or two-dimensional Walsh-Hadamard transform. For a given column vector $\underline{f}$ of N elements, the Walsh-Hadamard transform is defined as

$$\underline{F} = \underline{H}_N \underline{f} \tag{5.4}$$

and the vector $\underline{f}$ can be recovered from its transform by

$$\underline{f} = \frac{1}{N} \underline{H}_N \underline{F} \tag{5.5}$$

For the two-dimensional case, given a matrix g_{NxM}, the Hadamard transform is defined as

$$\underline{G}_{NxM} = \underline{H}_N \; g_{NxM} \underline{H}_M \tag{5.6}$$

$$g_{NxM} = \frac{1}{NM} \underline{H}_N \underline{G}_{NxM} \underline{H}_M \tag{5.7}$$

Since the Hadamard matrix is a matrix of elements plus and minus one, addition and subtraction will be sufficient to calculate the coefficients of the Walsh-Hadamard transform. However, for $N = 2^n$, a number of algorithms (34,38,40) have been developed using matrix factorization(35) to reduce the number of operations from N(N-1) to $N\log_2 N$ or nN. This not only reduces the number of computational operations, but also leads to considerable savings in storage requirements. Hence the name of fast Walsh Hadamard transform (FWHT) applies.

5.2 A Factorization Method For The Hadamard Matrix

It is well known that the matrix factorization* method has long been established(35) and a number of algorithms have been derived. Generally, to achieve 'in place' computation, the existing methods require a shuffle right after each operation has been performed(41).

---

* The matrix factorization is not for the Kronecker product.

In order to apply the fast Walsh-Hadamard transform directly on an H diagram and still retain the property of 'in place' computation we will first introduce a modified factorization method by stating the following theorem. In order to apply the fast Walsh-Hadamard transform directly on an H diagram and still retain the property of ' in place ' computation we will first introduce a modified factorization method by stating the following theorem.

Theorem 5.1 Given a Hadamard matrix $\underline{H}_N$ of order $N = 2^n$, for $n = 2, 3, \ldots$ then this matrix can be factored into a product of n $(N \times N)$ matrices of the following forms

$$\underline{H}_N = \underbrace{\begin{bmatrix} \underline{H}_2 & & & 0 \\ & \underline{H}_2 & & \\ & & \ddots & \\ 0 & & & \underline{H}_2 \end{bmatrix}}_{1} \underbrace{\begin{bmatrix} \underline{I}_2 & \underline{I}_2 & & & & 0 \\ \underline{I}_2 & -\underline{I}_2 & & & & \\ & & \underline{I}_2 & \underline{I}_2 & & \\ & & \underline{I}_2 & -\underline{I}_2 & & \\ & & & & \ddots & \\ 0 & & & & \underline{I}_2 & \underline{I}_2 \\ & & & & \underline{I}_2 & -\underline{I}_2 \end{bmatrix}}_{2} \cdots\cdots \underbrace{\begin{bmatrix} \underline{I}_{N/2} & \underline{I}_{N/2} \\ \underline{I}_{N/2} & -\underline{I}_{N/2} \end{bmatrix}}_{n}$$

$$\text{or} \quad \underline{H}_N = \prod_{i=1}^{n} \underline{T}_i \qquad \text{where } \underline{T}_i = \underline{I}_{2^{n-i}} \otimes \underline{H}_2 \otimes \underline{I}_{2^{i-1}}$$

Proof: The proof will be by induction.

For $n = 2$, $N = 4$

$$\underline{H}_4 = \begin{bmatrix} \underline{H}_2 & \underline{H}_2 \\ \underline{H}_2 & -\underline{H}_2 \end{bmatrix} = \begin{bmatrix} \underline{H}_2 & 0 \\ 0 & \underline{H}_2 \end{bmatrix} \begin{bmatrix} \underline{I}_2 & \underline{I}_2 \\ \underline{I}_2 & -\underline{I}_2 \end{bmatrix}$$

The theorem is true for this case.

Assuming that it is true for $n = k$ and $N = K = 2^k$, that is

$$\underline{H}_K = \begin{bmatrix} \underline{H}_2 & & & 0 \\ & \underline{H}_2 & & \\ & & \ddots & \\ 0 & & & \underline{H}_2 \end{bmatrix}_1 \begin{bmatrix} \underline{I}_2 & \underline{I}_2 & & & & 0 \\ \underline{I}_2 & -\underline{I}_2 & & & & \\ & & \underline{I}_2 & \underline{I}_2 & & \\ & & \underline{I}_2 & -\underline{I}_2 & & \\ & & & & \ddots & \\ 0 & & & & \underline{I}_2 & \underline{I}_2 \\ & & & & \underline{I}_2 & -\underline{I}_2 \end{bmatrix}_2 \cdots\cdots \begin{bmatrix} \underline{I}_{K/2} & \underline{I}_{K/2} \\ & \\ \underline{I}_{K/2} & -\underline{I}_{K/2} \end{bmatrix}_k$$

$$= \prod_{i=1}^{k} \underline{T}_i$$

Then, for $n = k+1$ and $N = 2K = 2^{k+1}$

$$\underline{H}_{2K} = \begin{bmatrix} \underline{H}_K & \underline{H}_K \\ \underline{H}_K & -\underline{H}_K \end{bmatrix} = \begin{bmatrix} \underline{H}_K & 0 \\ 0 & \underline{H}_K \end{bmatrix} \begin{bmatrix} \underline{I}_K & \underline{I}_K \\ \underline{I}_K & -\underline{I}_K \end{bmatrix} = (\underline{I}_2 \otimes \underline{H}_K) \begin{bmatrix} \underline{I}_K & \underline{I}_K \\ \underline{I}_K & -\underline{I}_K \end{bmatrix}$$

$$= [\, \underline{I}_2 \otimes (\prod_{i=1}^{k}) \,] \begin{bmatrix} \underline{I}_k & \underline{I}_k \\ \underline{I}_k & -\underline{I}_k \end{bmatrix}$$

By the property of the Kronecker-product[35], the first term can be expressed as $\prod_{i=1}^{k} (\underline{I}_2 \otimes \underline{T}_i)$. Hence,

$$
\underline{H}_{2K} = \underbrace{\left[\begin{array}{ccc|ccc}
\underline{H}_2 & & 0 & & 0 & \\
& \ddots & & & & \\
0 & & \underline{H}_2 & & & \\
\hline
& & & \underline{H}_2 & & 0 \\
& 0 & & 0 & \ddots & \\
& & & & & \underline{H}_2
\end{array}\right]}_{1}
\underbrace{\left[\begin{array}{cccc|cccc}
\underline{I}_2 & \underline{I}_2 & & 0 & & 0 & & \\
\underline{I}_2 & -\underline{I}_2 & & & & & & \\
& & \ddots & & & & & \\
0 & & \underline{I}_2 & \underline{I}_2 & & & & \\
& & \underline{I}_2 & -\underline{I}_2 & & & & \\
\hline
& & & & \underline{I}_2 & \underline{I}_2 & & 0 \\
& 0 & & & \underline{I}_2 & -\underline{I}_2 & & \\
& & & & & \ddots & & \\
& & & & & & \underline{I}_2 & \underline{I}_2 \\
& & & & & & \underline{I}_2 & -\underline{I}_2
\end{array}\right]}_{2}
$$

$$
\cdots \underbrace{\left[\begin{array}{cc|cc}
\underline{I}_{K/2} & \underline{I}_{K/2} & & 0 \\
\underline{I}_{K/2} & -\underline{I}_{K/2} & & \\
\hline
& & \underline{I}_{K/2} & \underline{I}_{K/2} \\
& 0 & \underline{I}_{K/2} & -\underline{I}_{K/2}
\end{array}\right]}_{k}
\underbrace{\begin{bmatrix}
\underline{I}_K & \underline{I}_K \\
\underline{I}_K & -\underline{I}_K
\end{bmatrix}}_{k+1}
$$

Q.E.D.

It is clear that each of the factor matrices $\underline{T}_k$ has exactly two non zero elements in each row (column). Therefore, for multiplication with a vector of N elements each factor matrix takes exactly N operations. There are altogether $n(= \log_2 N)$ factor matrices, hence the total number of operations is nN.

The factor matrices have the following properties

(1) $T_i^j = \begin{cases} \frac{j-1}{2} \, \underline{T}_i & \text{for } j \text{ an odd number} \\ \frac{i}{2} \, \underline{I}_N & \text{for } j \text{ an even number} \end{cases}$

(2) $\underline{T}_i^T = \underline{T}_i$ where $\underline{T}_i^T$ is the transpose of $\underline{T}_i$

Since $\underline{H}_N = \underline{H}_N^T = \prod_{i=1}^{n} \underline{T}_{n-i+1}^T = \prod_{i=1}^{n} \underline{T}_{n-i+1}$

That is $\prod_{i=1}^{n} \underline{T}_i = \prod_{i=1}^{n} \underline{T}_{n-i+1}$

5.3 Fast Walsh-Hadamard Transform By H Diagram

A simple, two-dimensional geometric model called H diagram was developed by Marihugh and Anderson[19] to serve as an effective visual aid in the analysis of binary functions. The method is based on a geometric transformation of the coordinates of a hypercube into a two-dimensional plane. Examples are shown in Fig. 5.1.

The extremities of the H diagram represent the canonical terms of a Boolean function in the same manner that vertices are used in the hypercube representation. The conventional positive and negative directions are adopted, respectively, for the unprimed variable and its complement. Assuming that x_1 is the most significant variable it is easier to map a function that is expressed in terms of the decimal equivalent. The corresponding decimal numbers are shown in Fig. 5.1 (b) and (c).

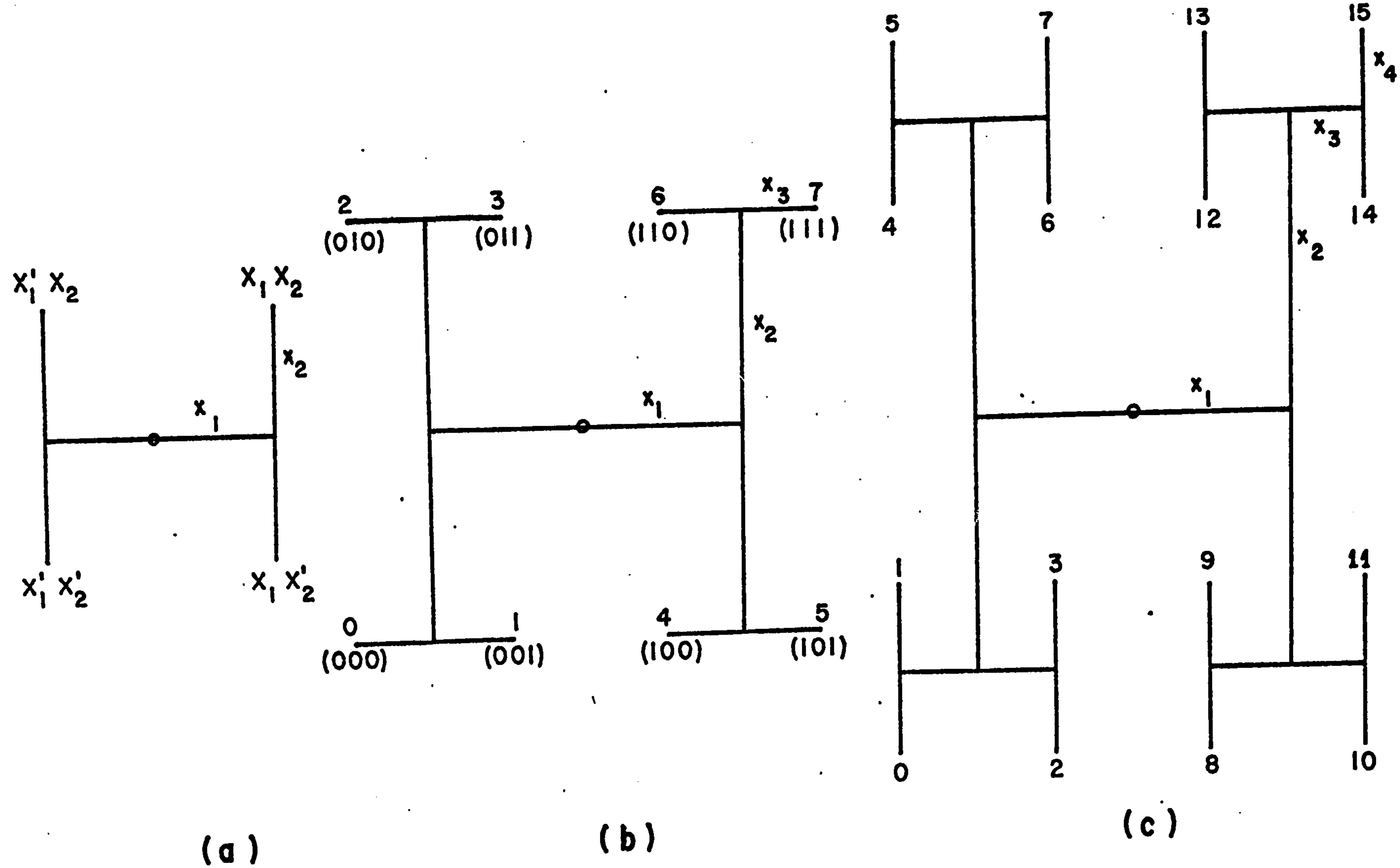


FIG. 5.1 H DIAGRAMS FOR 2, 3 AND 4 VARIABLES

Since there are 2^n extremities on an H diagram, a data vector, or a discrete function of 2^n elements can be represented on such an H diagram. This can be generalized to cover a data vector of any number of elements simply by adding zeroes and hence expanding the dimension of the vector to 2^n. The Walsh-Hadamard transform of such a vector can be obtained by multiplying a Hadamard matrix of the same order to this vector. Or we can think of the Hadamard matrix as an operator over the data vector, requiring additions and subtractions of the vector elements only. These operations may be performed along each of the axes of the H diagram successively. This is equivalent to factoring the Hadamard matrix as in Theorem 1.

To relate the factor matrix with the H diagram we will show that to pre-(post-) multiply each of the factor matrices with a row (column) vector is equal to adding and subtracting the corresponding vector elements about a particular axis of the H diagram. Let us examine the case for n = 3. The factor matrices are

$$\underline{T}_1 = \underline{I}_4 \otimes \underline{H}_2 \otimes \underline{I}_1 = \underline{I}_4 \otimes \underline{H}_2 = \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & 0 & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & 0 & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix}$$

$$\underline{T}_2 = \underline{I}_2 \otimes \underline{H}_2 \otimes \underline{I}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & & & & \\ 0 & 1 & 0 & 1 & & & & \\ 1 & 0 & -1 & 0 & & 0 & & \\ 0 & 1 & 0 & -1 & & & & \\ & & & & 1 & 0 & 1 & 0 \\ & 0 & & & 0 & 1 & 0 & 1 \\ & & & & 1 & 0 & -1 & 0 \\ & & & & 0 & 1 & 0 & -1 \end{bmatrix}$$

$$\underline{T}_3 = \underline{I}_1 \otimes \underline{H}_2 \otimes \underline{I}_4 = \underline{H}_2 \otimes \underline{I}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Let $\underline{f} = [a,b,c,d,e,f,g,h]^T$ be a real column vector to be operated on. Then

$$\underline{T}_1\underline{f} = \begin{bmatrix} a+b \\ a-b \\ c+d \\ c-d \\ e+f \\ e-f \\ g+h \\ g-h \end{bmatrix} \quad \underline{T}_2\underline{f} = \begin{bmatrix} a+c \\ b+d \\ a-c \\ b-d \\ e+g \\ f+h \\ e-g \\ f-h \end{bmatrix} \quad \underline{T}_3\underline{f} = \begin{bmatrix} a+e \\ b+f \\ c+g \\ d+h \\ a-e \\ b-f \\ c-g \\ d-h \end{bmatrix}$$

If we enter the elements of $\underline{f}$ into the H diagram according to the ascending order of the decimal numbering, we have the situation of Fig. 5.2

$\underline{f}$:

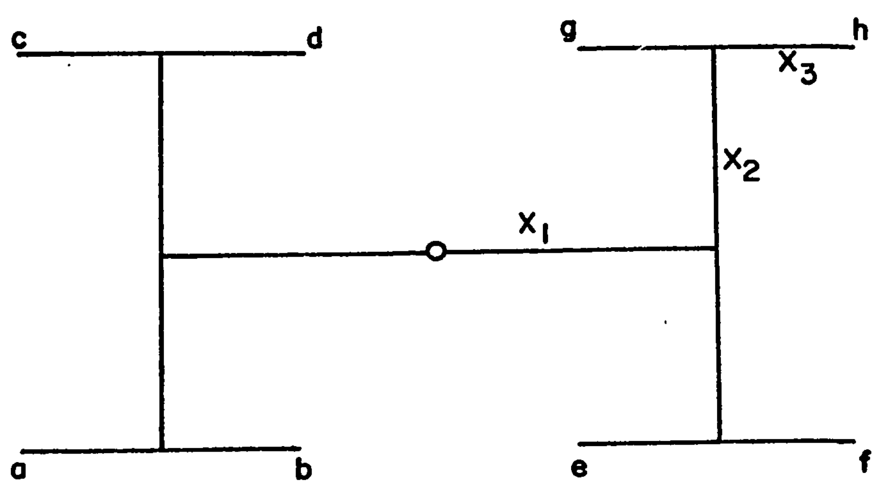


Fig. 5.2 The H Diagram of $\underline{f}$.

Post multiplying the matrices $\underline{T}_1$, $\underline{T}_2$, $\underline{T}_3$ with $\underline{f}$ is equivalent to adding and subtracting the two corresponding elements along the axes x_3, x_2, x_1 respectively as shown in Fig. 5.3.

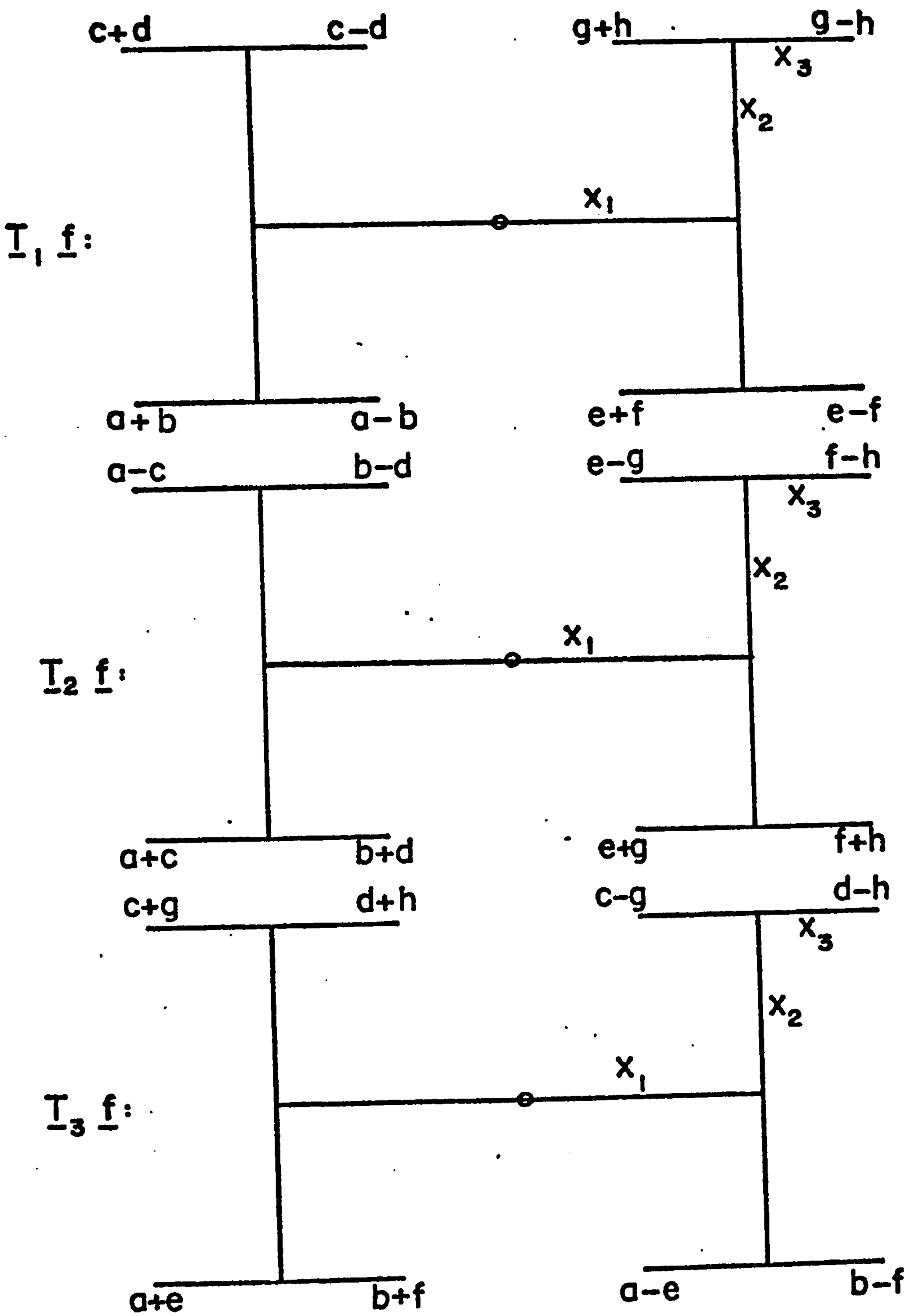


Fig. 5.3 The H diagrams of $\underline{f}$ premultiplied by factor matrices.

Since the Hadamard matrix is equal to the product of these factor matrices, the transform of a column vector can simply be obtained by performing the operations about each of the axes of the H diagram successively. And because of the symmetry property of the matrices the sequential order of the multiplications, and hence the order of operations is immaterial. However, as a rule we would prefer to operate from the axis corresponding to the least significant variable to the axis corresponding to the most significant one. This is shown in Fig. 5.4.

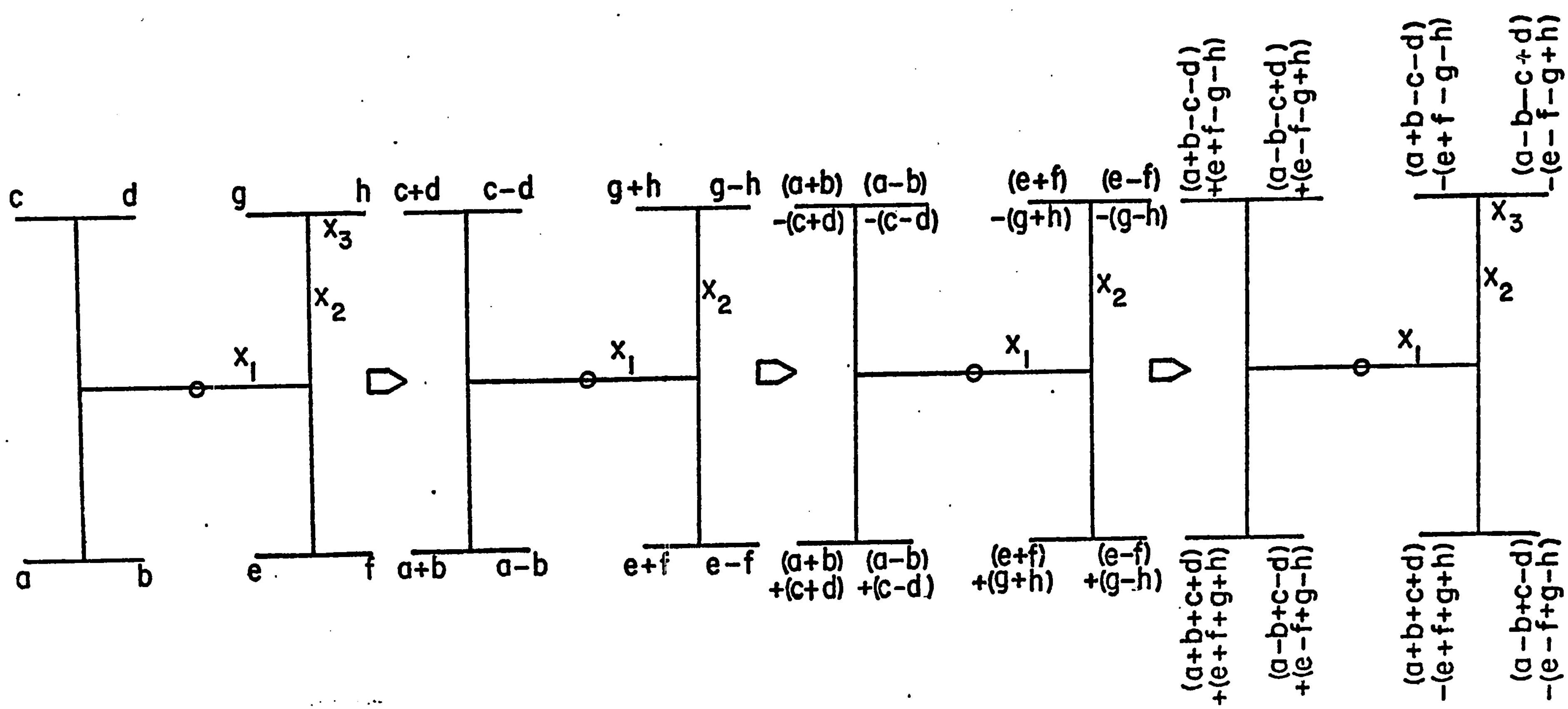
For the two dimensional case, let g be a (4 x 2) matrix.

$$\underline{g} = [g_1 \; g_2] = \begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}$$ where the elements are real numbers

and its Walsh-Hadamard transform is

$$\underline{G} = \underline{H}_4 \; \underline{g} \; \underline{H}_2$$

The premultiplication of $\underline{g}$ by $\underline{H}_4$ can be performed separately on the two H diagrams for $\underline{g}_1$ and $\underline{g}_2$ as shown in Fig. 5.5(b) and (c). To postmultiply the resultant by $\underline{H}_2$ is equivalent to linking the two separate diagrams by another axis between the center points and performing additions and subtractions about this new axis. This is shown in Fig. 5.5(d). If we follow the numbering convention of H diagrams, i.e. the zero cell is position invariant and always at the left-lowest corner, and the outermost axis corresponds to the least significant bit, then Fig. 5.5(d) and Fig. 5.4(d) give us the same results. This agrees with Theorem 5.1.



$\underline{f}$ (a) $\underline{T}_1\,\underline{f}$ (b) $\underline{T}_2\underline{T}_1\,\underline{f}$ (c) $\underline{T}_3\underline{T}_2\underline{T}_1\,\underline{f} = \underline{H}_8\underline{f} = \underline{\mathbf{F}}$ (d)

FIG 5.4 FWHT ON THE H DIAGRAM

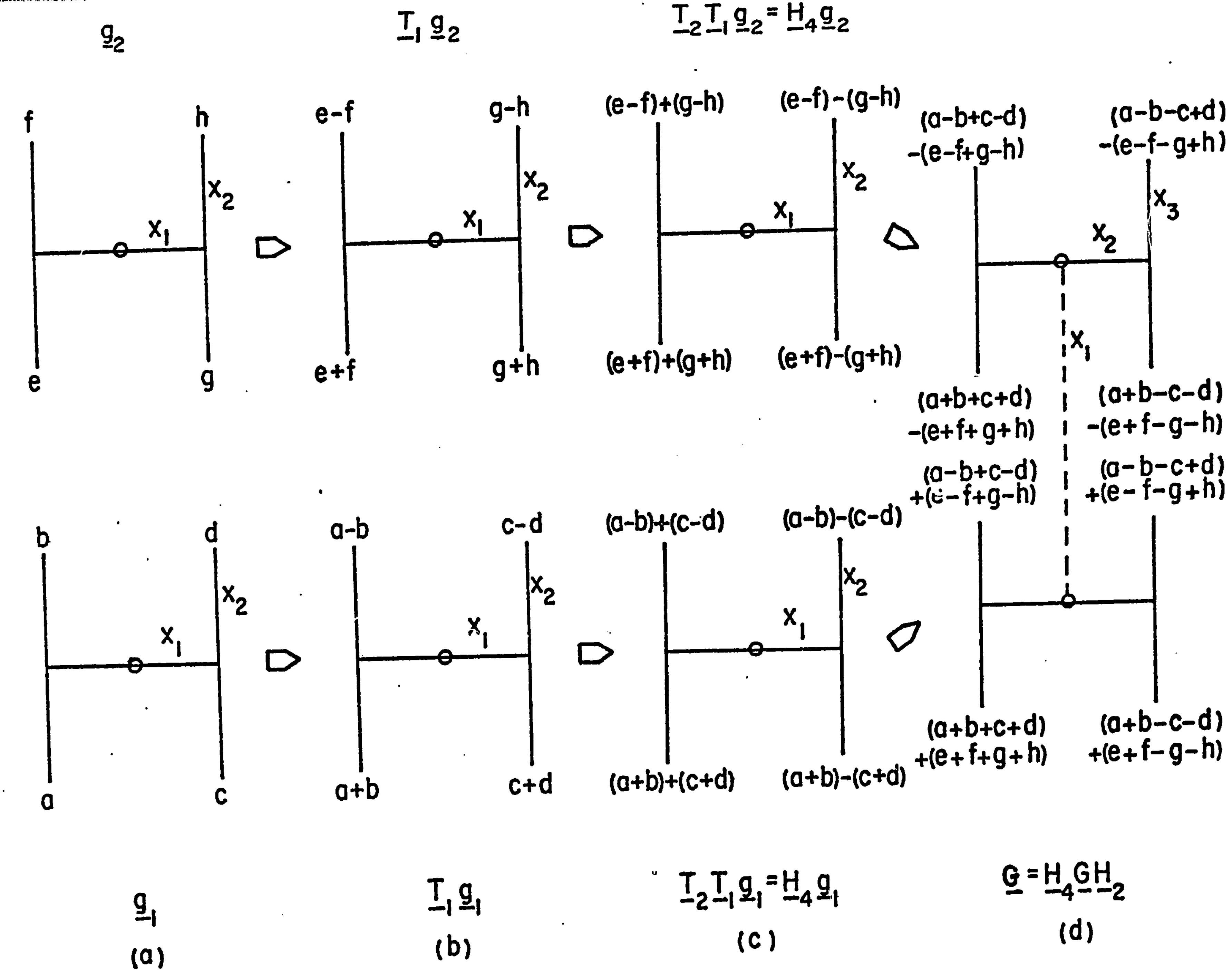


FIG. 5.5 FWHT ON THE H DIAGRAM (2-DIMENSIONAL CASE)

The procedure of getting the FWHT can now be summarized as follows:

1. Enter the discrete function f to an H diagram. If the number of its elements is not equal to a power of 2, increment the function by asserting zeros until the number of its elements is $N = 2^n$.

2. Set the index register to n, perform addition and subtraction about the axis corresponding to the content in the index register. Replace the sum in the lower or left location of the two operands and the difference in the upper or right location.

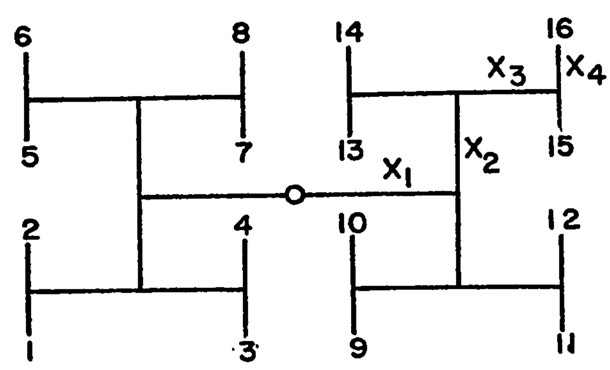
3. Reduce the index by one and repeat step 2 until the index becomes zero. Then the resultant will be the Walsh-Hadamard transform of function f.

Since we reuse the storage location of the original function to get its transform, the so-called ' in place ' computation is achieved. Thereby, the savings in storage is evident.

Example 5.1 Using the above-mentioned procedure, find the FWHT of the discrete function f = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16). In this case, N = 16 and $n = \log_2 N = 4$.

Step 1

The function f

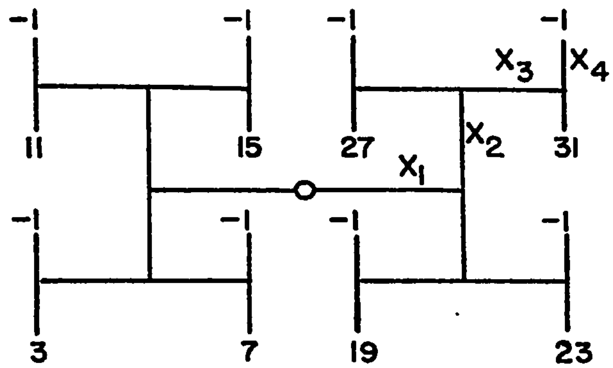


Step 2

Addition and subtraction about the axis x_4

0	0	4

Index Register



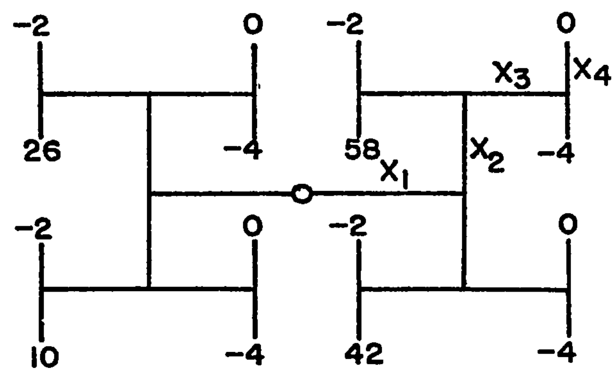
Step 3

Reducing the index by one

0	0	3

Step 2-1

About the axis x_3



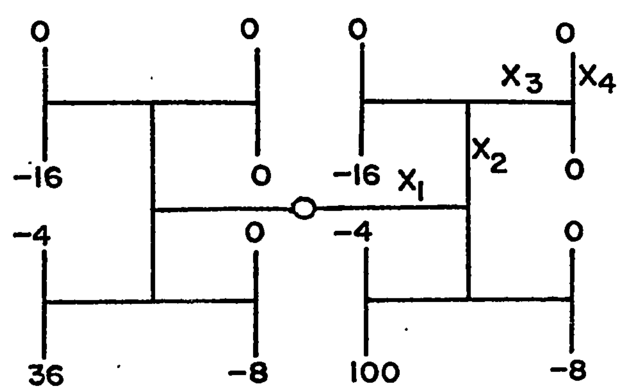
Step 3-1

Reducing the index by one

0	0	2

Step 2-2

About the axis x_2



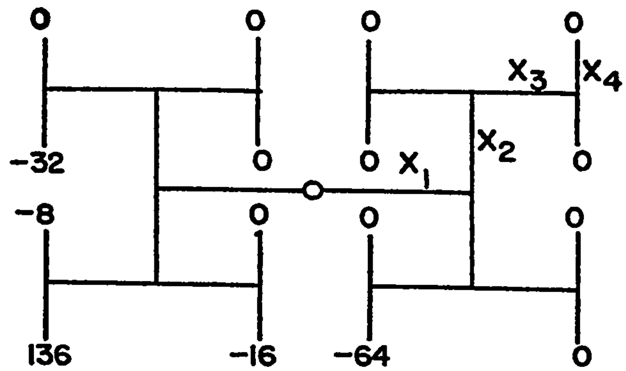
Step 3-2

Reducing the index by one

0	0	1

Step 2-3

About the axis x_1



Step 3-3

Reducing the index by one

0	0	0

Since the index becomes zero, the resultant is the Walsh-Hadamard transform of the function f, i.e., (136, -8, -16, 0, -32, 0, 0, 0, -64, 0, 0, 0, 0, 0, 0, 0).

5.4 Summary

In this chapter we have developed an algorithm for obtaining the fast Walsh-Hadamard transform on an H diagram by using ' in place ' computations. Comparing with the other methods, this algorithm seems to be simple, clear and straightforward. The H diagram not only provides a visual aid to the process of the computations, but also helps to supply a tool for the factorization of a Hadamard matrix. In case that the order of a Hadamard matrix becomes large, the H diagram can be decomposed into several H diagrams with fewer axes. Then the spectral coefficients can be obtained by getting the Hadamard transform on these subdiagrams, recombining them with some new axes, and eventually performing additions and substractions along these new axes. This is analogous to having a two-dimensional transform instead of a one-dimensional one. Higher-dimensional transforms can be achieved by repeating the decomposition and recombination processes. Another merit might be that the spectral analysis can be done directly on the H diagram as well. This will be the subject of Chapter 6.

CHAPTER 6

SPECTRAL ANALYSIS ON THE H DIAGRAMS

In this chapter, Walsh-Hadamard transforms of one-dimensional patterns are analyzed, an "axis-symmetry" parameter τ [42,47] and a row-index i [48] which can distinguish the members of the complete set of Walsh functions are introduced and defined. It is shown that there is a one-to-one correspondence among the row-index, the sequency and the axis-symmetry of each Walsh function . Through the row-index, the period [47,48] of a Walsh function can also be obtained easily.

The representation of a pattern in the Walsh domain is called "Walsh spectrum". It can be a sequency spectrum, an axis-symmetry or simply a Walsh-Hadamard transform depending on the ordering of the coefficients. It is a measure of the correlation of the pattern with the Walsh functions. The spectral analysis concerning axis-symmetries is stressed in this chapter by using H diagrams. And a simpler procedure for obtaining Walsh-Hadamard transform for those functions that possess axis-symmetries is shown by illustrating examples.

6.1 Axes Of Dyadic Translation And H Diagram

It is one of the important properties that Walsh functions possess even and odd symmetries about axes. Thus, while the Fourier functions constitute the natural representation of systems with translational symmetry, the Walsh functions has the natural representation of systems with dyadic symmetry. Dyadic convolution, dyadic invariance, and dyadic translations are terms used in the theory of linear systems analysis [43,44]. Recently, dyadic integration and differentiation are also introduced by different researchers[45,46]. Only dyadic translations are of interest here because they are closely related to symmetries.

When a function f(t) is sampled at $N = 2^n$ discrete points in time, the discrete function $\underline{f} = (f_o, f_1, \ldots, f_{N-1})$ is produced. A dyadic translation [44] f_τ of function f(t) gives the function $f(t \oplus \tau)$, where $t \oplus \tau$ represents the modulo-2 sum between the respective entries of the binary expressions of t and τ. Consider the ramp function f(t). Under such sampling procedure, f(t) is represented by the discrete function $\underline{f} = (0,1,2,3,4,5,6,7)$. This discrete ramp function with its dyadic translations x_τ for $\tau = 1, 2, \ldots, 7$ are shown in Fig. 6.1.* For $\tau = 0$, the function remains unchanged, i.e., f(t). Hence, dyadic translations are simple permutations of the sampled components of function f(t).

---

* Fig. 2, pp. 18 in Pichler's paper [44].

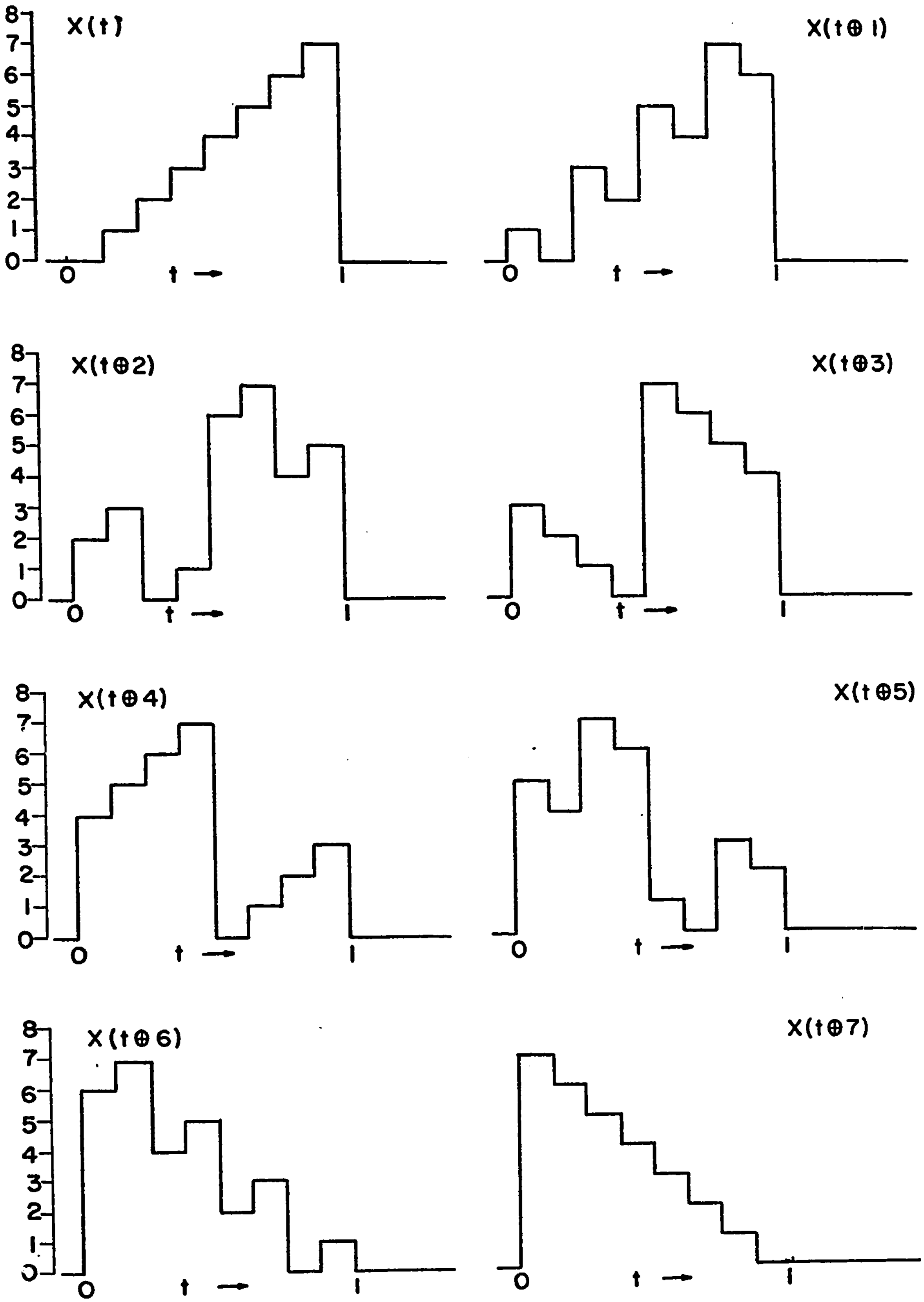


Figure 6.1 Dyadic translations f_τ, $\tau = 0, 1, \ldots, 7$ of a Sample Ramp Function $f(t)$.

A discrete function with $N = 2^n$ components defines n axes of dyadic translations. This is shown in Fig. 6.2[42] for $n = 1, 2$ and 3.

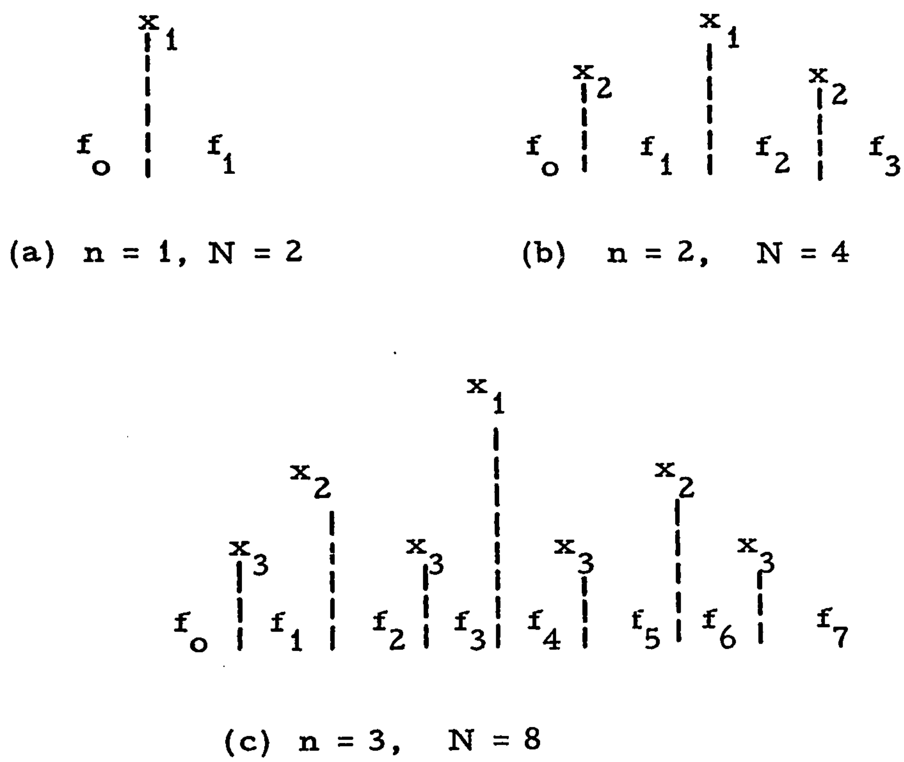


Figure 6.2 Axes Of Dyadic Translations.

From the definition of these axes, every dyadic translation of Fig. 6.1 is obtained by translating (or interchanging) the vector component about respective axes. The binary expressions of τ are associated with translations about axes x_3, x_2, x_1. A one for x_i indicates translations about axis x_i; a zero indicates no translation. For example, $\underline{f}\,(t \oplus 3)$ is generated from $\underline{f}$ by translating the entries of $\underline{f}$ about axes x_2 and x_1, which corresponds to the 1's in the binary expression of $\tau = 011$. Table 6.1 shows the functions which result when a discrete function of eight components undergoes dyadic translations.

Table 6.1

FUNCTIONS UNDER τ DYADIC TRANSLATIONS

$\underline{f}(t \oplus \tau) = \underline{f}_{\tau}$	$x_1 x_2 x_3$	Axes of dyadic translations: x_3 x_2 x_3 x_1 x_3 x_2 x_3							
$\underline{f}(t \oplus 0) = \underline{f}_0$	0 0 0	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$\underline{f}(t \oplus 1) = \underline{f}_1$	0 0 1	f_1	f_0	f_3	f_2	f_5	f_4	f_7	f_6
$\underline{f}(t \oplus 2) = \underline{f}_2$	0 1 0	f_2	f_3	f_0	f_1	f_6	f_7	f_4	f_5
$\underline{f}(t \oplus 3) = \underline{f}_3$	0 1 1	f_3	f_2	f_1	f_0	f_7	f_6	f_5	f_4
$\underline{f}(t \oplus 4) = \underline{f}_4$	1 0 0	f_4	f_5	f_6	f_7	f_0	f_1	f_2	f_3
$\underline{f}(t \oplus 5) = \underline{f}_5$	1 0 1	f_5	f_4	f_7	f_6	f_1	f_0	f_3	f_2
$\underline{f}(t \oplus 6) = \underline{f}_6$	1 1 0	f_6	f_7	f_4	f_5	f_2	f_3	f_0	f_1
$\underline{f}(t \oplus 7) = \underline{f}_7$	1 1 1	f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0

Now, if we enter the components of each function $\underline{f}_i$ for i = 0, 1, ..., 7 into an H diagram following the way as mentioned in Chapter 5, we will obtain eight different H diagrams representing the corresponding $\underline{f}_i$'s. Fig. 6.3 shows the H diagrams for these $\underline{f}_i$'s. It is not surprising to find out that the axes of H diagram coincide with the axis defined for dyadic translations, since in mathematics a dyadic group is a group of binary numbers and the H diagram is specially suitable for the coordinate representation of binary functions. However, the H diagrams provide a much better tool for obtaining the dyadic translation or detecting axis symmetries.

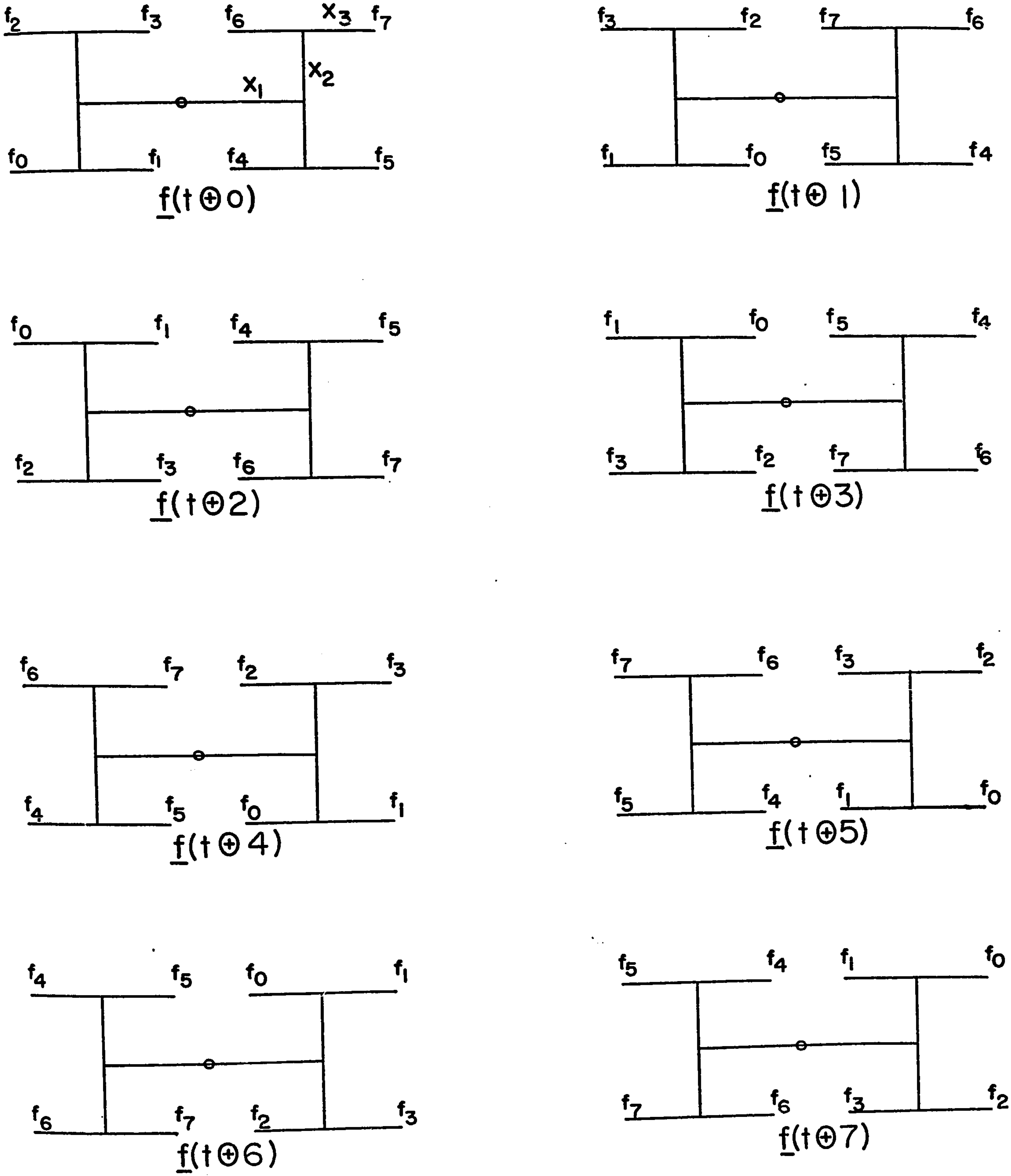


Fig. 6.3. H Diagram For Dyadic Translations.

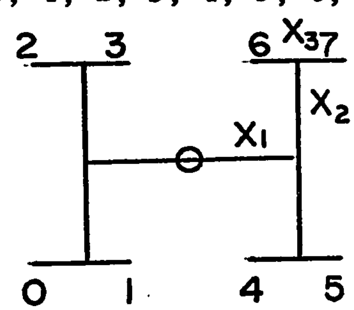
6.2 Axis-Symmetry And Walsh Functions

Definition 6.1[47]: A discrete function $\underline{f} = (f_0, f_1, \ldots, f_{N-1})$ possesses τ axis-symmetry when it is invariant under τ dyadic translations ; it remains the same when its entries are translated about all x_i axis that corresponding to τ_i's = 1 of the binary expression of τ.

Since we have shown in the previous section the axes of dyadic translations coincide with the axes of an H diagram; the τ axis-symmetry can be easily obtained from the H diagram representing the function. The following examples show some discrete functions and their respective axis-symmetries.

Example 6.1 $\underline{f} = (0, 1, 2, 3, 4, 5, 6, 7)$.

The H diagram of $\underline{f}$:

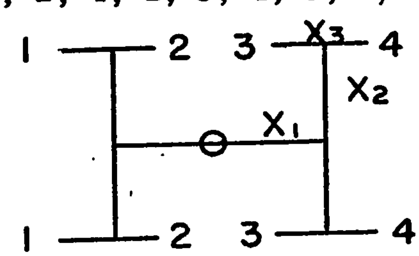


Axes of symmetry : none.

Hence $\tau_1 = \tau_2 = \tau_3 = 0$, $\tau_{decimal} = 0$

Example 6.2 $\underline{f} = (1, 2, 1, 2, 3, 4, 3, 4)$

The H diagram of $\underline{f}$:

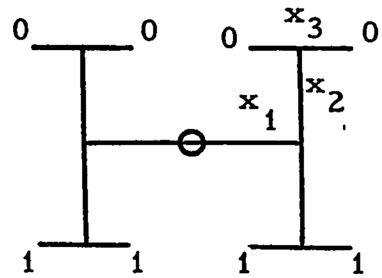


Axis of symmetry: x_2

Hence, $\tau_1 = \tau_3 = 0$, $\tau_2 = 1$; and $\tau_{decimal} = 2$.

Example 6.3 $\underline{f} = (1, 1, 0, 0, 1, 1, 0, 0)$.

The H diagram of $\underline{f}$



Axes of symmetry : x_1 and x_3

Hence, $\tau_1 = \tau_3 = 1$, $\tau_2 = 0$; and $\tau_{decimal} = 5$.

Therefore, axis-symmetry is another characteristic property of the Walsh functions in addition to the sequency and period. The relationship with them will be shown in the next section. Table 6.2 shows the τ axis-symmetries of Walsh functions for the case N = 8.

Table 6.2

Axis-Symmetries Of Walsh Functions For N = 8.

Wal (s, m)	Walsh functions	Axis-Symmetry $\tau_{decimal}$	$= \tau_1 \tau_2 \tau_3$
Wal (0, m)	+ + + + + + + +	7	1 1 1
Wal (1, m)	+ + + + - - - -	3	0 1 1
Wal (2, m)	+ + - - - - + +	1	0 0 1
Wal (3, m)	+ + - - + + - -	5	1 0 1
Wal (4, m)	+ - - + + - - +	4	1 0 0
Wal (5, m)	+ - - + - + + -	0	0 0 0
Wal (6, m)	+ - + - - + - +	2	0 1 0
Wal (7, m)	+ - + - + - + -	6	1 1 0

6.3 Relations of Sequency, Axis-Symmetry, Row-Index And Period of a Walsh Function.

Let the domain m of a Walsh function wal (s, m) be divided into $N = 2^n$ subintervals. Then the period p of a Walsh function is the minimum number for which

$$wal_N(s, m+p) = wal_N(s, m) \qquad (6.1)$$

is true for all m, when the addition is modulo N[48].

The period is the smallest number of subintervals after which the Walsh function repeats itself. By examining the Walsh functions listed in Table 6.2 we can find that their periods are 1, 2, 4 or 8. All of them are powers of 2 and if the period is expressed in binary notation, then only 1 bit will be equal to 1 and the rest will be zero.

To find the period of a Walsh function or to relate it with sequency, the best way is to use the row position (which will be called the row-index and counting from zero) of that particular Walsh function in an Hadamard matrix. It is well known that there is a one-to-one correspondence and simple relationship between the row-index and the sequency[50]. Actually, if we express the sequency by its Gray code of $n = \log_2 N$ bits, say $g_{n-1} g_{n-2} \cdots g_0$, and reverse the bits, then the decimal representation of this reversed bit Gray code using general binary transformation will give the row-index i of the corresponding Walsh function. In mathematical expression,

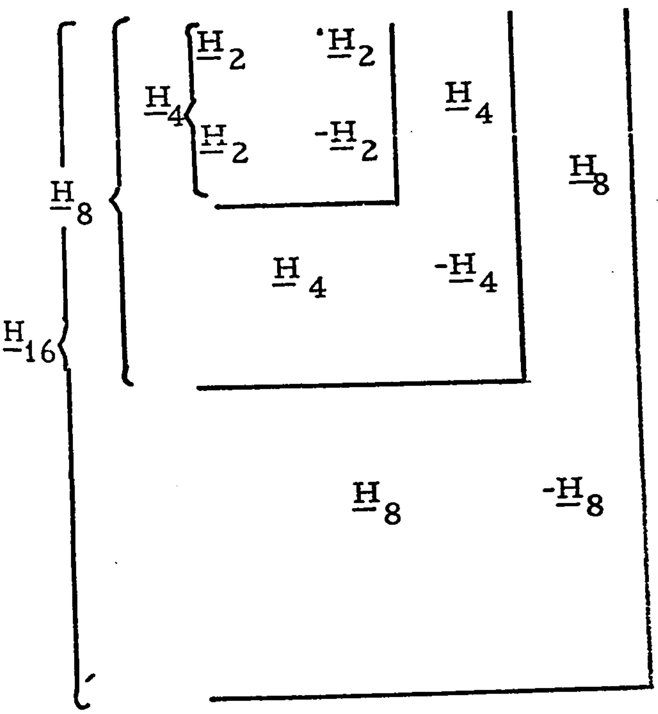
$$i = \sum_{j=0}^{N-1} g_j \, 2^{n-j-1} \qquad (6.2)$$

For example, let N = 8 and S = 4, then the Gray code for 4 will be 110 and the reversed bit Gray code becomes 011. The decimal representation for 011 is 3. Therefore, the Walsh function with sequency 3 will be in the third row of the Hadamard matrix counting from zero.

As mentioned in Chapters 2 and 5, an Hadamard matrix of order $N = 2^n$ can be generated simply by the Kronecker-product:

$$\underline{H}_N = \underline{H}_2^{[n]} \tag{6.3}$$

where the bracketed exponent means the Kronecker-product is performed n times upon $\underline{H}_2$. Any Hadamard matrix will have the nested structure shown in Fig. 6.4* for $\underline{H}_{16}$.



$i_{dec.}$	i_3	i_2	i_1	i_0	p_4	p_3	p_2	p_1	p_0	$p_{dec.}$ p
0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	0	1	0	2
2	0	0	1	0	0	0	1	0	0	4
3	0	0	1	1	0	0	1	0	0	4
4	0	1	0	0	0	1	0	0	0	8
5	0	1	0	1	0	1	0	0	0	8
6	0	1	1	0	0	1	0	0	0	8
7	0	1	1	1	0	1	0	0	0	8
8	1	0	0	0	1	0	0	0	0	16
9	1	0	0	1						
10	1	0	1	0						
11	1	0	1	1						
12	1	1	0	0						
13	1	1	0	1						
14	1	1	1	0						
15	1	1	1	1						

Figure 4 Nesting In The Hadamard Matrix $\underline{H}_{16}$

* Fig. 1, pp. 230 in Parkyn Jr.'s paper(49).

For those Walsh functions having row-indices $i = p/2$ to $p - 1$, the period is always equal to p, since each of them is merely N/p consecutive repetitions of the row in $\underline{H}_p$ having the same index i. The periods along with the row-indices are listed side by side in Fig. 6.4 for $\underline{H}_{16}$. It has to be pointed out that if we want to express the period in binary notation an extra bit is required, i.e. for $N = \log_2 n$ we need n+1 bits instead of n bits. Let us assume that the binary notations for i and p are $i_{n-1} i_{n-2} \cdots i_o$ and $p_n p_{n-1} \cdots p_o$ respectively, then they are related as follows:

$$
\begin{aligned}
p_n &= i_{n-1} \\
p_{n-1} &= \overline{i_{n-1}}\, i_{n-2} \\
p_{n-2} &= \overline{i_{n-1}}\, \overline{i_{n-2}}\, i_{n-3} \\
&\vdots \\
p_2 &= \overline{i_{n-1}}\, \overline{i_{n-2}} \cdots \overline{i_2}\, i_1 \\
p_1 &= \overline{i_{n-1}}\, \overline{i_{n-2}} \cdots \overline{i_1}\, i_0 \\
p_0 &= \overline{i_{n-1}}\, \overline{i_{n-2}} \cdots \overline{i_1}\, \overline{i_0}
\end{aligned}
\qquad (6.4)
$$

Table 6.3

Walsh Functions, Sequency, Axis-Symmetry, Row-Index And Period For N=8.

Walsh Functions	Sequency $\sum_{j=o}^{2} s_j 2^j = S$	$s_2 s_1 s_0$	Axis-Symmetry $\sum_{j=1}^{3} \tau_j 2^{3-j} = \tau$	$\tau_1 \tau_2 \tau_3$	Row-Index $\sum_{j=o}^{2} i_j 2^j = i$	$i_2 i_1 i_0$	Period $\sum_{j=o}^{3} p_j 2^j = p$	$p_3 p_2 p_1 p_0$
+ + + + + + + +	0	0 0 0	7	1 1 1	0	0 0 0	1	0 0 0 1
+ + + + - - - -	1	0 0 1	3	0 1 1	4	1 0 0	8	1 0 0 0
+ + - - - - + +	2	0 1 0	1	0 0 1	6	1 1 0	8	1 0 0 0
+ + - - + + - -	3	0 1 1	5	1 0 1	2	0 1 0	4	0 1 0 0
+ - - + + - - +	4	1 0 0	4	1 0 0	3	0 1 1	4	0 1 0 0
+ - - + - + + -	5	1 0 1	0	0 0 0	7	1 1 1	8	1 0 0 0
+ - + - - + - +	6	1 1 0	2	0 1 0	5	1 0 1	8	1 0 0 0
+ - + - + - + -	7	1 1 1	6	1 1 0	1	0 0 1	2	0 0 1 0

Table 6.3 lists the 8-bit Walsh functions along with the sequency, the axis-symmetry, the row-index and the period. The relationship between the i_j's and τ_k's is extremely simple as shown in the following equation :

$$i_j = \overline{\tau_{3-j}} \qquad \text{for } j = 0, 1, 2$$

In summary, the row-index relates with sequency by reversed bit Gray to binary conversion, with axis-symmetry by complementation and with the period by the equations shown in (6.4). Through these links all other relations can also be easily derived.

6.4 Axis-Symmetry Spectrum.

In the previous section we have shown that there exists simple relation among sequencies, axis-symmetries, row-indices and periods of Walsh functions. Moreover, the one-to-one correspondences between sequency and row-index, and axis symmetry and row-index of Walsh function are shown. The coefficients of Walsh-Hadamard transform defined as in Chapter 5 is ordered according to the ascending order of the row-index. If sequency spectrum is required, the coefficients have to be rearranged according to the relationship as listed in Table 6.3. Since an extremely simple relation exists between row-index and axis-symmetry, we can easily obtain an axis-symmetry spectrum directly from the H diagram which is used for obtaining the fast Walsh Hadamard transform in Chapter 5.

Analogous to the sequency spectrum, an axis-symmetry spectrum of a function is an orthogonal transformation using Walsh functions as bases, hence it is also a measure of the correlation of this function with the Walsh functions. It differs from the sequency spectrum in that the ordering of the coefficients is according to the ascending order of the axis-symmetry instead of the sequency.

Let the Walsh-Hadamard transform of a discrete function $\underline{f}$ be $\underline{F} = (F_0, F_1, \ldots\ldots, F_{N-1})$, and its axis-symmetry spectrum be $\underline{A} = (A_0, A_1, \ldots\ldots, A_{N-1})$. Then,

$$A_i = F_{N-1-i} \quad \text{for all } i \tag{6.5}$$

This is due to the fact that if both the row-index and axis-symmetry are expressed in binary numbers, one can be obtained from another simply by complementations. Since both of these two numbers have the same number of bits, namely $n = \log_2 N$, and range from 0 to N-1, their relation in decimal representation is shown above in (6.5) as the subscripts for A and F .

Assuming that the transform is obtained by using fast algorithms on an H diagram, then the axis-symmetry spectrum can be read out directly from this H diagram. The procedure is that instead of starting from the left - most bottom corner, going from left to right and upward along the axes, it starts from the right-most top cell of the H diagram, going from right to left and downward. This is shown by the following examples.

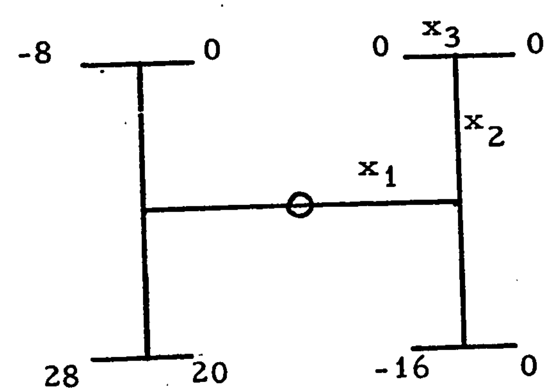
<u>Example 6.4</u> $\underline{f}$ = (0, 1, 2, 3, 4, 5, 6, 7)

The Walsh-Hadamard transform of $\underline{f}$

$\underline{F}$ = (28, 20, -8, 0, -16, 0, 0, 0)

(The H diagram for $\underline{F}$ is shown on the right)

$\underline{A}$ = (0, 0, 0, -16, 0, -8, 20, 28) .

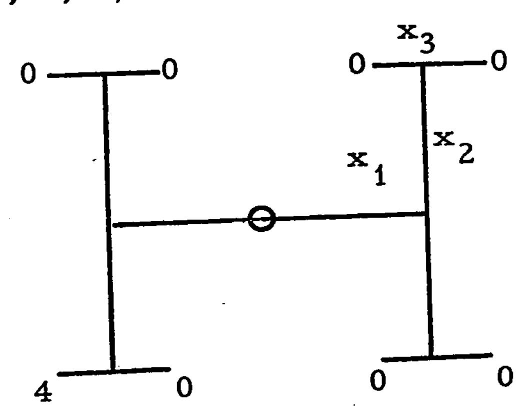


<u>Example 6.5</u> $\underline{f}$ = (1, 1, 0, 0, 1, 1, 0, 0).

The Walsh-Hadamard transform of $\underline{f}$

$\underline{F}$ = (4, 0, 4, 0, 0, 0, 0, 0).

$\underline{A}$ = (0, 0, 0, 0, 0, 4, 0, 4) .



6.5 Axis-Symmetry Spectral Analyzis

Let us examine the spectrum coefficients of the discrete function possessing axis-symmetries. First, we assume that a discrete function of 8 components is symmetry about axis x_3 . Referring to Fig. 5.4(a), this is equivalent to say that a = b, c = d, e = f and g = h. Under this condition and by Fig. 5.4(d) we will find that

$$A_0 = A_2 = A_4 = A_6 = 0$$

where A_i's are the axis-symmetry spectrum coefficients of the function. If we denote the subscripts of these zero coefficients by binary number, then all its least significant bit will be equal to zero. Table 6.4 lists all the zero coefficients for functions that possess axis-symmetries

Table 6.4

Zero Coefficients For Functions That Possess Axis-Symmetries

Symmetries about	Zero coefficients	Subscripts in binary
x_1	A_0, A_1, A_2, A_3	000, 001, 010, 011
x_2	A_0, A_1, A_4, A_5	000, 001, 100, 101
x_3	A_0, A_2, A_4, A_6	000, 010, 100, 110
x_1, x_2	$A_0, A_1, A_2, A_3, A_4, A_5$	000, 001, 010, 011, 100, 101
x_1, x_3	$A_0, A_1, A_2, A_3, A_4, A_6$	000, 001, 010, 011, 100, 110
x_2, x_3	$A_0, A_1, A_2, A_4, A_5, A_6$	000, 001, 010, 100, 101, 110
x_1, x_2, x_3	$A_0, A_1, \ldots A_6$	000, 001, ..., 110

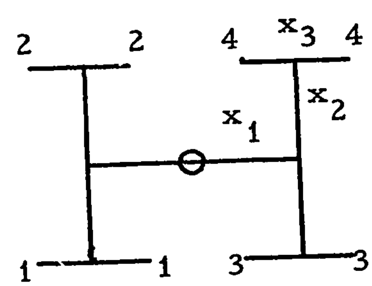
Although Table 6.4 listed only N = 8 case, it can be generalized to any length $N = 2^n$. Let the subscripts in binary be denoted by $(b_1, b_2, \ldots, b_n)$. Then, for a function having axis-symmetry about x_i, all its spectrum coefficients with subscripts $b_i = 0$ will be equal to zero. For a function having axis-symmetries about x_i and x_j, all its spectrum coefficients with subscripts $b_i = 0$ or $b_j = 0$ will be equal to zero. And for a function having axis-symmetries about all the axes, all its spectrum coefficients will be equal to zero except the one with the subscript 11 .... 1 or $(2^n - 1)$ in decimal. It is worth pointing out that the set of zero coefficients for a function possessing axis-symmetries about x_i and x_j is the union of the two sets of zero coefficients for the functions possessing axis-symmetry about x_i and x_j separately.

Taking into account this symmetry property, for a function possessing axis-symmetries the procedure for obtaining its fast Walsh-Hadamard transform can be simplified. This is illustrated by the following examples.

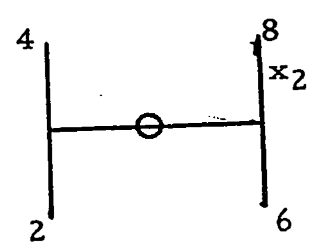
<u>Example 6.6</u> $\underline{f} = (1, 1, 2, 2, 3, 3, 4, 4)$.

Axis-symmetry about x_3

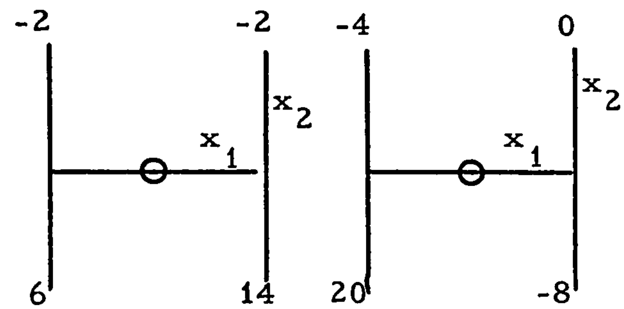
(1) H diagram of $\underline{f}$



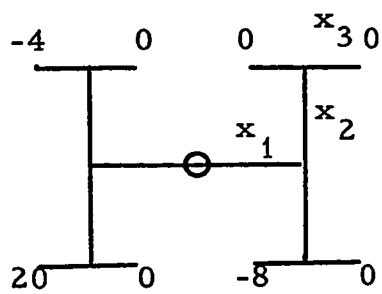
(2) Reduce the axis corresponding to x_3 and double the values



(3) Perform the fast Walsh-Hadamard transform on this reduced H diagram



(4) Expand the H diagram by adding the x_3 axis. Retain the values on the left of this axis and insert zeroes on the right.

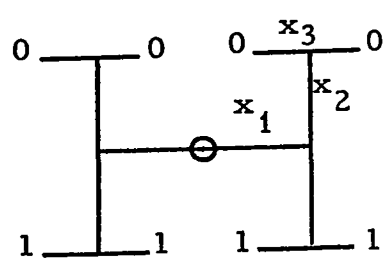


Thus, the Walsh-Hadamard transform $\underline{F} = (20, 0, -4, 0, -8, 0, 0, 0)$ or, the axis-symmetry spectrum $\underline{A} = (0, 0, 0, -8, 0, -4, 0, 20)$.

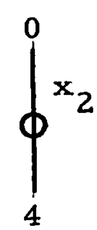
Example 6.7 $\underline{f} = (1, 1, 0, 0, 1, 1, 0, 0)$

Axis-symmetries about x_1 and x_3.

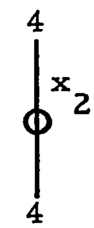
(1) H diagram of $\underline{f}$



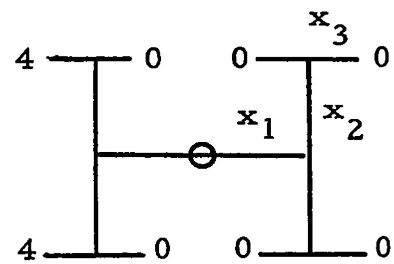
(2) Reduce the axes corresponding to x_2 and x_3 and times the values by four



(3) Perform the fast Walsh-Hadamard transform on this reduced H diagram



(4) Expand the H diagram by adding the x_1 and x_3 axes, Retain the values on the left-most corner and insert zero on the rest



The Walsh-Hadamard transform $\underline{F}$ = (4, 0, 4, 0, 0, 0, 0, 0)
or, the axis-symmetry spectrum $\underline{A}$ = (0, 0, 0, 0, 0, 4, 0, 4)

Certain combinations of the axis-symmetry spectral values are invariant to cyclic shifts that preserves axis-symmetries[42]. Table 6.6 shows these combinations for the case N = 8.

Table 6.6

Invariant Quantities Under Cyclic Shifts

Symmetries about	Invariant Quantities
x_1	$D_1^2 = A_4^2 + A_5^2 + A_6^2 + A_7^2$
x_2	$D_2^2 = A_2^2 + A_3^2 + A_6^2 + A_7^2$
x_3	$D_3^2 = A_1^2 + A_3^2 + A_5^2 + A_7^2$
x_1, x_2	$D_{12}^2 = A_6^2 + A_7^2$
x_1, x_3	$D_{13}^2 = A_5^2 + A_7^2$
x_2, x_3	$D_{23}^2 = A_3^2 + A_7^2$
x_1, x_2, x_3	$D_{123}^2 = A_7^2$

Comparing Table 6.6 with Table 6.5, the invariant quantities are nothing but the sum of the squares of the non-zero coefficients.

6.6 Summary

The "axis-symmetry" was first introduced and defined in terms of dyadic translation. In addition to the sequency, every Walsh function can be distinguished by a unique axis-symmetry number. A row-index representing the row position of each Walsh function in a Hadamard matrix was also introduced. It was shown that simple relations exist among the sequency, axis-symmetry, row-index and period of a Walsh function. Among the first three there also exists a one-to-one correspondence.

The axes of an H diagram coincide the axes used for defining the axis-symmetry. And it is also useful for setting up the row-index. Hence, with great ease one can do spectral analysis on the H diagram. The analysis of spectra for those functions that possess axis-symmetries was shown in this chapter. A simpler procedure for obtaining the Walsh-Hadamard transform of those functions was also achieved on an H diagram.

CHAPTER 7

CONCLUSIONS AND COMMENTS FOR FURTHER STUDY

In conclusion, the properties of Walsh-Hadamard transforms have been studied from different points of view. Initially, a brief survey on the philosophy and the applications of the Walsh-Hadamard transform was presented. The logical Walsh-Hadamard transform was generalized for certain longer lengths. A combinatorial network for obtaining the logical Walsh-Hadamard transform was worked out for N = 2, 4 and 8. The two-dimensional version was also provided. The piecewise logical Walsh-Hadamard transform was introduced by using modular design and permutations. The H diagram was not only used for implementation of the logical Walsh-Hadamard transform, but also found useful in the conventional fast Walsh-Hadamard transform and in the spectral analysis of those functions that possess axis-symmetries.

The following problems may be worth further study.

1. The performance and efficiency of logical Walsh-Hadamard comparing with the conventional Walsh-Hadamard transform and other orthogonal transforms in terms of speed, rate-distortion, quantized levels, etc.

2. It is well known that the Hadamard matrix is not unique. The classification of the Hadamard matrices, especially for those that have an order of multiple of four but not a power of two, will be interesting.

3. To perform dyadic differentiation, dyadic integration and the like by using H diagrams.

4. Spectral analysis for two-dimensional patterns on an H diagram.

APPENDIX I

To show that (a) $\text{LWHT}\,(1,x_1,x_2,x_3,1,x_1,x_2,x_3)=(1,z_1,z_2,z_3,0,0,0,0)$

(b) $\text{LWHT}\,(1,x_1,x_2,x_3,0,x_1,x_2,x_3)=(1,0,0,0,z_1,z_2,z_3,z_4)$

Proof: (a) Let $\underline{X} = [1,x_1,x_2,x_3,1,x_1,x_2,x_3]$ and $\underline{Z}$ its LWHT in vector form. Assuming that $\underline{Y} = [y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8]$ is the conventional Walsh-Hadamard transform of $\underline{X}$.

By definition,

$$Y = \underline{X}\,\underline{H}_8$$

$$= [1,x_1x_2,x_3,1,x_1,x_2,x_3] \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Or,

$$y_1 = 2(1 + x_1 + x_2 + x_3)$$
$$y_2 = 2(1 - x_1 + x_2 - x_3)$$
$$y_3 = 2(1 + x_1 - x_2 - x_3)$$
$$y_4 = 2(1 - x_1 - x_2 + x_3)$$
$$y_5 = y_6 = y_7 = y_8 = 0$$

Since x_i's take on values of 0 and 1, y_1 is always greater than 1. By applying threshold, $z_1 = 1$ and $z_5 = z_6 = z_7 = z_8 = 0$. Let $z_i = f_M(y_i)$ for i = 2, 3, and 4, then $\underline{Z} = \text{LWHT}\,(\underline{X})$.

(b) This time let $\underline{X} = 1, x_1, x_2, x_3, 0, \overline{x}_1, \overline{x}_2, \overline{x}_3$. The same assumption as in (a) is for $\underline{Y}$ and $\underline{Z}$. Then we have

$$y_1 = 1 + (x_1+\overline{x}_1) + (x_2+\overline{x}_2) + (x_3+\overline{x}_3)$$

$$y_2 = 1 - (x_1+\overline{x}_1) + (x_2+\overline{x}_2) - (x_3+\overline{x}_3)$$

$$y_3 = 1 + (x_1+\overline{x}_1) - (x_2+\overline{x}_2) - (x_3+\overline{x}_3)$$

$$y_4 = 1 - (x_1+\overline{x}_1) - (x_2+\overline{x}_2) + (x_3+\overline{x}_3)$$

For i = 5, 6, 7 and 8, y_i's are of no concern ; we simply let $z_i = f_M(y_i)$.

Since x_i's take on values of 0 and 1, we have,

$$x_i + \overline{x}_i = 1 \quad \text{for } i = 1, 2 \text{ and } 3$$

Therefore,

$$y_1 = 4$$

$$y_2 = y_3 = y_4 = 0$$

Hence, $z_1 = 1$ and $z_2 = z_3 = z_4 = 0$. Thus $\underline{Z} = \text{LWHT}(\underline{X})$

APPENDIX II

A Program for the Frequency Count of the Walsh-Hadamard Transform Coefficients

```
                                                               WATFIV 2ND JOB CARD
$JOB   $W681180    Y. L. MAR
       DIMENSION MF(16), MI(16), MJ(16), MG(16), MH(16), NC(16,16)
        COMMON N,NH, MF, NVAR
    20 FORMAT (I1)
    30 FORMAT ('X'//' FREQUENCY COUNT OF W.H.T. COEFFS. FOR N = ', I3,/)
    40 FORMAT (I9, 8I6)
    25 READ (1, 20) NVAR
        N = 2**NVAR
        NH = N/2
        N3 = N + NH
        N2 = N3 + 1
       DO 28 I=1,N2
       DO 28 J=1,N
    28 NC(I,J) = 0
C       GENERATION OF DATA
       NN = 2**N
       NN1 = NN/2
    24 DO 11 II=1,NN1
       K = NN-II
       CALL DBNHX (K, 2)
       DO 17 J = 1, N
    17 MI(J) = MF(J)
    16  CALL FINDV
       DO 26 J = 1,N
        IX = 1 + N - MF(J)
        NC1 = NC(IX,J) + 1
    26 NC(IX,J) = NC1
    11 CONTINUE
       WRITE (3,30) N
       DO 27 I = 1, N3
        IX1 = N - I + 1
    27 WRITE (3,40) IX1, (NC(I,J), J=1,N)
        GO TO 25
       END .

        SUBROUTINE FINDV
       DIMENSION MB(16)
        COMMON NS, NHS, MFS(16), NVS.
        DO 120  KK=1, NVS
        DO 100 I = 1, NHS
   100  MB(I) = MFS(2*I-1)+MFS(2*I)
        N1=NHS+1
        DO 110 I= N1, NS
         J=2*I-NS-1
```

```
  110  MB(I) = MFS(J) - MFS(J+1)
       DO 120 I=1, NS
  120  MFS(I) = MB(I)
       RETURN
       END

       SUBROUTINE DBNHX (ID, NMD)
       COMMON ND, NHD, MFD(16), NVAD
       DO 105 I=1, ND
       KD = ND-I+1
       MFD(KD) = MOD(ID, NMD)
  105  ID = (ID-MFD(KD)) / NMD
       RETURN
       END

$ENTRY
```

REFERENCES

(1) S.G.S. Shiva et al, "Review of digital image processing techniques for bandwidth reduction," Univ. of Ottawa Report, December, 1972.

(2) P.M. Will and K.S. Pennington, "Grid Coding: A Novel Technique for Image Processing," Proc. IEEE, Vol. 60, No. 6, pp. 669-680, June 1972.

(3) W.T. Cochran et al, "What is the fast Fourier transform?" Proc. IEEE, Vol. 55, No. 10, pp. 1664-1674, October, 1967.

(4) J.W. Cooley and J.W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. of Comput., Vol. 19, pp. 297-301, April, 1965.

(5) A. Rosenfeld, Picture Processing by Computer. Academic Press, New York, London, 1969.

(6) C.E. Shannon, "A mathematical theory of communications," Bell Systems Tech. J., Vol. 27, pp. 379-423, 623-656, 1948.

(7) G.B. Anderson and T.S. Huang, "Piecewise Fourier transformation for picture bandwidth compression," IEEE Trans. Comm. Tech., Vol. COM-19, No. 2, pp. 133-140, April,1971.

(8) H.C. Andrews, "Multidimensional Rotations in Feature Selection," IEEE Trans. Computers, Vol. C-20, pp. 1045-1051, September 1971.

(9) A. Grasselli, Automatic Interpretation and Classification of Images. Academic Press, New York, London, 1969, Ch. 5.

(10) K.S. Fu, Sequential Methods in Pattern Recognition and Machine Learning. Academic Press, New York, London, 1968, Ch. 2.

(11) W.B. Davenport Jr. and W.L. Root, An Introduction to the Theory of Random Signals and Noise. McGraw-Hill, New York, 1958.

(12) H.C. Andrews and J. Kane, "Kronecker matrices, computer implementation, and generalized spectra," J. Ass. Comput. Mach., Vol. 17, pp. 260-268, April, 1970.

(13) H.C. Andrews and K.L. Caspari, "A generalized technique for spectral analysis," IEEE Trans. Computers, Vol. C-19, pp. 16-25, Jan. 1970.

(14) J.W. Carl, "Generalized harmonic analysis for pattern recognition: A biologically derived model," thesis, Air Force Institute of Technology, School of Engg., August 1968.

(15) P.I.P. Goulton, "Smearing techniques in pattern identification," Ph.D. dissertation, University of Toronto, Toronto, Ontario, Canada, 1966.

(16) H.C. Andrews, "Walsh functions in image processing, feature selection and pattern recognition,"Proc. of the 1971 Symp. on Walsh Functions, pp. 26-32, Washington D.C., April,1971.

(17) H.F. Harmuth, Transmission of Information by Orthogonal Functions. Springer-Verlag, New York / Heidelberg / Berlin, 1970, and 2nd Edition, 1972.

(18) N.H. Searle, "A logical Walsh transform," Proc. of the 1970 Symp. on Walsh Functions, pp. 95-98, April, 1970.

(19) G.E. Marihugh and R.E. Anderson, "The H diagram: A graphical approach to logic design, "IEEE Trans. Computers, Vol. C-20, pp. 1192-1196, October 1971.

(20) T.S. Huang, "Run-length coding and its extensions," in Picture Bandwidth Compression, ed. by T.S. Huang and O.J. Tretiak, Gordon and Breach, N.Y., 1972.

(21) J. Capon, "A probabilistic model for run-length coding of pictures," IEEE Trans. Inform. Theory, Vol. IT-5, pp. 157-163, Dec. 1959.

(22) R.B. Lackay, "The wonderful world of Walsh functions," Proc. of the 1972 Applications of Walsh Functions, pp. 2-7, Washington D.C., March 1972.

(23) N. Ahmed and K.R. Rae, "Walsh functions and Hadamard transform," Proc. of the 1972 Applications of Walsh Functions, pp. 8-13, Washington D.C., March 1972.

(24) W.A. Parkyn, Jr., and G.E. Cash, "Analysis of the logical Walsh transform for N = 16," Proc. of the 1972 Applications of Walsh Functions, pp. 281-288, March 1972.

(25) M.A. Harrison, Introduction to Switching and Automata Theory, McGraw-Hill, 1965.

(26) C.L. Sheng, Introduction to Switching Logic, Intext, 1972.

(27) D.J. Sakrison and V.R. Algazi, "Comparison of line-by-line and two-dimensional encoding of random images," IEEE Trans. Inform. Theory, Vol. IT-17, No. 4, pp. 386-398, July 1971.

(28) M. Tasto and P.A. Wintz, "A bound on the rate-distortion function and application to images," IEEE Trans. Inform. Theory, Vol. It-18, No. 1, pp. 150-159, Jan. 1972.

(29) J.D. Kennedy, "Walsh function imagery analysis," Proc. of the 1971 Symp. on Walsh Functions, pp. 7-10, Washington D.C., April 1971.

(30) E.J. Claire, S.M. Farber and R.R. Green, "Practical techniques for transform data compression/image coding," Proc. of the 1971 Symp. on Walsh Functions, pp. 2-6, Washington D.C., April 1971.

(31) G.G. Murray, "Modified transforms in imagery analysis," Proc. of the 1972 Applications of Walsh Functions," pp. 235-239, Washington, D.C., March 1972.

(32) R.B. Blackman and J.W. Tukey, The Measurement of Power Spectra, From the Point of View of Communications Engineering, Dover, New York, 1959.

(33) H.C. Andrews, Computer Techniques in Image Processing, Academic Press, New York and London, 1970.

(34) W.K. Pratt, J. Kane, and H.C. Andrews, "Hadamard transform image coding." Proc. IEEE, Vol. 57, pp. 58-68, January 1969.

(35) I.J. Good, "The Interaction Algorithm and Practical Fourier Analysis," J. Royal Statistical Society, B20, No. 2, pp. 361-372, 1958.

(36) W.M. Gentlemen, "Matrix multiplication and fast Fourier transformations," Bell Sys. Tech. J., Vol. 47, pp. 1099-1103, July-August 1968.

(37) S.W. Golomb et al, Digital Communications - With Space Application. Prentice - Hall Inc., Englewood Cliffs, N.J., 1964.

(38) J.E. Whelchel and D.F. Guinn, "The fast Fourier-Hadamard transform and its use in signal representation and classification," Eascon 1968 Record, pp. 561-573.

(39) J.L. Shanks, "Computation of the Fast Walsh-Fourier transform," IEEE Trans. Computers, Vol. C-18, pp. 457-459, May 1969.

(40) Y. Y. Shum and A.R. Elliott, "Computation of the fast Hadamard transform," Proc. Symp Appl. of Walsh Functions, pp. 177-180 , Washington, 1972.

(41) H.S. Stone, "Parallel processing with the perfect shuffle," IEEE Trans. Computers, Vol C-20, pp. 153-161, February, 1971.

(42) N.A. Alexandridis, "Walsh-Hadamard transformations in image processing," Engineering Report No. 71-8, March 1971, University of California, Los Angeles, California.

(42) R.E.A.C. Paley, "A remarkable series of orthogonal functions," Proc. London Math. Soc., Vol. 34, pp. 241-279, 1932.

(43) H.L. Peterson, "Generation of Walsh functions," Proc. of the 1970 Symp. on Walsh Functions, pp. 55-57, Washington D.C., April 1970.

(44) F. Pichler, "Walsh functions and optimal linear systems," Proc. of the 1970 Symp. on Walsh Functions, pp. 17-22, Washington D.C., April 1970.

(45) E. Gibbs and M.J. Millard, "Walsh functions as solutions of a logical differential equation," DES Report 1 (1969), National Physical Laboratory, Teddington, Middlesex, England.

(46) P.L. Butzer and H.J. Wagner, "A calculus for Walsh functions defined on R_+," (to appear) Proc. of the 1973 Symp. on Walsh Functions, Washington D.C., April 1973.

(47) N.A. Alexandridis, "Relations among sequency, axis-symmetry, and period of Walsh functions," IEEE Trans. Inform. Theory, Vol. IT - 17, pp. 495-497, July 1971.

(48) W.A. Parkyn, Jr., "Digital image processing aspects of the Walsh transform," Proc. of the 1970 Symp. on Walsh Functions pp. 152-156, Washington D.C., April 1970.

(49) W.A. Parkyn, Jr., "Sequency and periodicity concepts in imagery analysis,"Proc. of the 4th Hawaii International Conference on Sys. Sc., pp. 229-232, North Hollywood, Calif., 1971.

(50) N.A. Alexandridis and A. Klinger, "Walsh orthogonal functions in geometrical feature extraction," Proc. of the 1971 Symp. on Walsh Functions, pp. 18-25, Washington D.C., April, 1971.

VITA

NAME: Yung-Leung Henry MAR

BORN: Feb. 12, 1940, Shanghai, China

EDUCATED:

Primary : Hong Kong

Secondary: Normal University's Affiliated Middle School, Taipei, China

University: (a) National Taiwan University, Taipei, China. 1961 B. Sc. in E. E.

(b) University of Ottawa, Ottawa, Ontario, Canada, 1970. M.Sc. in E. E.

UNIVERSITE D'OTTAWA / UNIVERSITY OF OTTAWA
École des études supérieures/School of Graduate Studies

Title of thesis THE PIECEWISE LOGICAL WALSH-HADAMARD TRANSFORM AND APPLICATIONS OF H DIAGRAMS

Name of candidate MAR, Henry Y.L.

Degree Ph.D. Department ELECTRICAL ENGINEERING

Date of defence July 6, 1973.

We, the undersigned, certify that we have approved this thesis and that the candidate has defended it successfully.

K.S. Fu ______ (Signature of Examiners)

S.G.S. Shiva ______

W. Steenaart ______

E.L. Cohen ______

C.L. Sheng (not present at the defence)
(Thesis Supervisor)

(Dean of Graduate Studies)

UNIVERSITÉ D'OTTAWA / UNIVERSITY OF OTTAWA
École des études supérieures/School of Graduate Studies

NAME OF AUTHOR MAR, Henry Y.L.

TITLE OF THESIS THE PIECEWISE LOGICAL WALSH-HADAMARD TRANSFORM AND APPLICATIONS OF H DIAGRAMS

DEGREE Ph.D. (Electrical Engineering) YEAR GRANTED 1973

Henry Y. L. Mar
(Signed) (Author)

Date: July 6, 1973.

Permanent Address:
~~32 Gott~~
Bell Northern Research
Dept 3E10, Ottawa, Ont.