FAST TRANSFORMATIONS WITH WALSH-HADAMARD FUNCTIONS

FAST TRANSFORMATIONS
WITH
WALSH-HADAMARD FUNCTIONS

by

FRANK Y. Y. SHUM, B.Eng. (McMASTER UNIV.)

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Engineering

McMaster University
September 1972

MASTER OF ENGINEERING (1972) McMASTER UNIVERSITY
(Electrical Engineering) Hamilton, Ontario

TITLE: Fast Transformations with Walsh-Hadamard Functions

AUTHOR: Frank Y. Y. Shum, B.Eng., McMaster University

SUPERVISOR: Professor A. R. Elliott

NUMBER OF PAGES: 120, vii

SCOPE AND CONTENTS:

In this thesis, various techniques to generate Walsh-Hadamard functions are discussed. Efficient algorithms to compute the discrete Walsh-Hadamard transform have been derived and implemented. The design of a simple, but very fast, digital circuit that can perform the transform or its inverse is presented. These algorithms have been applied to the processing of speech for the investigation of bit rate reduction. Intelligible speech has been reconstructed from 8 or 4 dominant Walsh-Hadamard coefficients out of a field of 64, with a constant update time of 8 milliseconds, on a CDC-1700 computer.

ACKNOWLEDGMENTS

I would like to thank Dr. A. R. Elliott for his constant stimulation and guidance throughout the course of this work. The financial assistance provided by the Department of Electrical Engineering is gratefully acknowledged. Gratitude also is due to Dr. S. S. Haykin, Director of the Communications Research Laboratory, for providing the computer facility. I would also like to thank my colleague Mr. C. M. Thorsteinson for his advice and assistance in programming.

A special note of thanks goes to Mrs. Vivian Tong for typing the manuscript. Last, but certainly not the least, my deepest gratitude is to my wife Lai for her patience, love, and constant encouragement throughout the past two years.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER I

INTRODUCTION

The system of sine and cosine functions plays a distinguished role in many engineering applications, notably in network theory and communications. The concept of frequency was originally defined as the parameter f in $\sin(2\pi ft)$ and $\cos(2\pi ft)$. Whenever the term frequency is used, reference is made implicitly to these functions. Sinusoidal functions have indisputable advantages for the analysis of linear time-invariant circuits, because they are only attenuated and delayed by such network elements as resistors, capacitors and coils, their frequency and shape remaining unchanged. Fourier series and Fourier transforms permit the representation of a large class of functions by a superposition of sine and cosine functions. It is also possible to represent various types of functions by other orthogonal series. This thesis deals with a particular set of orthogonal functions called Hadamard functions and another set called Walsh functions.

Prior to the development of digital computers, few systems took real advantage of the power of orthogonal transformations except for the traditional Fourier spectral analysers. The advent of digital computers have brought a radical change to scientific research and applications. Digital

computers are more versatile than their analog counterparts and they are capable of handling a vast amount of data. Discrete Fourier decompositions were attempted, but the implementation proved to be complicated and time consuming. An algorithm that vastly reduces the computational effort for evaluating the discrete Fourier coefficients of a time series was reported by Cooley and Tukey in 1965[1]. This algorithm, now widely known as the "fast Fourier transform" or FFT, has opened new avenues of scientific investigation never considered practical because of exorbitant computing times. General areas in which the FFT is finding successful application include digital signal enhancement, real-time speech analysis, image enhancement in pattern recognition, spatial filtering, power spectra estimation, decomposition of convolved functions, and computation of auto-correlation, covariance, and other related functions. In essence, the scientific analyst now has available to him all those frequency-domain analysis techniques once considered inefficient.

The theory of orthogonal transformations is becoming an active area in which computer users are searching for powerful signal processing tools. At the same time, technological advancement in semi-conductor devices and integrated circuits has made it possible to generate almost any waveform economically. Methods other than the Fourier technique have begun to become prevalent in certain aspects of digital signal processing. The purpose of this thesis is to

investigate two sets of orthogonal functions that are of increasing importance in many scientific applications. These functions were defined by J. Hadamard[2] in 1893 and J. L. Walsh[3] in 1923, but have remained relatively unnoticed until recently. These Hadamard and Walsh functions are binary in nature; they assume a value of +1 or -1 on the interval of definition. By assigning +1 to logical 1 and -1 to logical 0, these waveforms may be generated economically from integrated circuits that are readily available. Furthermore, computer algorithms similar to the FFT for evaluating the discrete Hadamard and Walsh coefficients of a time series have been developed, and they are commonly known as the "fast Hadamard transform" or FHT[4] and the "fast Walsh transform" or FWT[5] . Implementation of these algorithms involves mostly additions and subtractions. Consequently, the FHT or FWT has an advantage in speed and cost over the FFT which requires multiplication of complex numbers. While the importance of the FFT cannot be discounted, the FHT and FWT are emerging as a research tool in spectral analysis, pattern recognition, and various other applications.

Previous work by Pratt[4], Campanella[6], Boesswetter[7] and others has demonstrated the feasibility of using the discrete Hadamard or Walsh transform in image or voice processing with promising results. Part of this thesis is devoted to the implementation of a technique for speech analysis and synthesis on the CDC-1700 computer installation

at McMaster University. Tentative results from this research have shown that highly intelligible speech may be reproduced from a small proportion of coefficients in either the Walsh or the Hadamard domain. One possible application of this approach is in areas where data compression rather than quality is the governing criterion. Such may be the case with word recognition or speech synthesis schemes with a relatively large vocabulary (500 - 5000 words).

In order to understand the Walsh-Hadamard transform as a research tool, Chapter II covers the basic theoretical material on orthogonal transformations in general. Chapter III deals specifically with the Walsh and Hadamard functions, while Chapter IV discusses several methods of implementing the FWT or FHT in terms of computational efficiency and memory savings. Based on one of these algorithms, a high-speed hardware circuit for obtaining the first eight Walsh or Hadamard coefficients is proposed. Chapter V discusses the application of Walsh-Hadamard transformation to speech analysis and reconstruction. Conclusions and recommendation for future research are given in Chapter VI.

CHAPTER II

A THEORETICAL BACKGROUND FOR ORTHOGONAL TRANSFORMATIONS

2.1 INTRODUCTION

This chapter is devoted to a brief discussion of orthogonal functions and includes some specific mathematical definitions that are used in subsequent chapters. For simplicity, examples used in the discussion of orthogonal transformations in this chapter are confined to the best known example which is the Fourier technique. The details of the Hadamard and Walsh transforms are given in Chapter IV. However, those details are based on the concepts presented in this chapter. Finally, some of the reasons which justify the use of orthogonal transformations in signal processing are given.

2.2 ORTHOGONAL FUNCTIONS

A system of real and almost everywhere non-vanishing functions $\{g(n,x)\}$ is defined to be orthogonal in the interval $a \leq x \leq b$ if the following criterion is satisfied :

$$\int_a^b g(n,x)\, g(m,x)\, dx = K_n \delta_{m,n} \qquad (2\text{-}1)$$

where K_n = a constant

$\delta_{m,n}$ = Kronecker delta

$$\delta_{m,n} = 1 \quad \text{if } m = n$$
$$= 0 \quad \text{otherwise.}$$

When the constant K_n in equation (2-1) is equal to unity for all values of n, the functions g(n,x) are said to be normalized, and $\{g(n,x)\}$ is called an orthonormal set. The Walsh and the Hadamard systems to be discussed are orthonormal.

There are virtually an infinite number of orthogonal systems. However, most of these systems are not complete, i.e., there exist non-trivial functions orthogonal to all of them. Consequently, they do not permit a convergent series expansion of all quadratically integrable functions[1]. Nevertheless, some of these are of great practical interest.

2.3 SERIES EXPANSION BY ORTHOGONAL FUNCTIONS

Orthogonal functions are of great interest and utility from both theoretical and applied viewpoints. One of the properties of an orthogonal system $\{g(n,x)\}$ is that the functions are linearly independent. Hence they may be used in the series representation of an arbitrary function s(x) which is defined in the interval $a \leq x \leq b$, provided that :

s(x) is periodic, i.e., s(x) = s(x+b-a) ;

s(x) has a finite number of discontinuities ;

---

[1] Any function s(x) is defined to be quadratically integrable if

$$\int_{x_1}^{x_2} s^2(x)\, dx < \infty$$

$$\int_a^b s(x)\,dx \qquad \text{exists.}$$

For instance, the function $s(x)$ may be expanded in a series of an orthonormal system $\{g(n,x)\}$:

$$s(x) = \sum_{n=0}^{\infty} c(n)\, g(n,x) \tag{2-2}$$

Any coefficient $c(m)$ may be evaluated by multiplying equation (2-2) with $g(m,x)$ and integrating the products in the interval of orthogonality. Due to the orthogonal nature of $\{g(n,x)\}$, all terms on the right-hand side of the resulting equation will vanish except for $n = m$:

$$\int_a^b s(x)g(m,x)dx = c(m)\int_a^b g(m,x)g(m,x)dx \tag{2-3}$$

Since the functions on the right are orthonormal, the equation reduces to the simple form :

$$\int_a^b s(x)g(m,x)dx = c(m)\int_a^b dx \tag{2-4}$$

Therefore,

$$c(m) = \frac{1}{b-a}\int_a^b s(x)g(m,x)dx \tag{2-5}$$

The other coefficients can be evaluated in the same way. Once all the coefficients have been obtained, the original function $s(x)$ can be reconstructed using equation (2-2).

To measure how well a function s(x) is approximated by equation (2-2) when n is finite, the mean square deviation Q of s(x) from its series representation may be used :

$$Q = \int_a^b \left[s(x) - \sum_{n=0}^{k} c(n)g(n,x) \right]^2 dx \qquad (2\text{-}6)$$

Substituting equation (2-4) into the cross-product term after the expansion and using the orthonormal property of $\{g(n,x)\}$, the value of Q is :

$$Q = \int_a^b s^2(x)dx - (b-a) \sum_{n=0}^{k} c^2(n) \qquad (2\text{-}7)$$

The so-called Bessel's inequality follows from equation (2-7):

$$\int_a^b s^2(x)dx \geq (b-a) \sum_{n=0}^{\infty} c^2(n) \geq (b-a) \sum_{n=0}^{k} c^2(n) \qquad (2\text{-}8)$$

If Q converges to zero with increasing k for any function s(x) which is quadratically integrable in the interval $a \leq x \leq b$, the orthonormal system $\{g(n,x)\}$ is said to be complete. This is known as the completeness theorem or Parseval's theorem. In this case, the equality condition in equation (2-8) holds true :

$$\int_a^b s^2(x)dx = (b-a) \sum_{n=0}^{\infty} c^2(n) \qquad (2\text{-}9)$$

A complete orthonormal system is always closed, i.e.,

the equality

$$\int_a^b s(x)g(n,x)dx = 0 \qquad (2\text{-}10)$$

is not satisfied for all values of n when the function s(x) is quadratically integrable. A thorough discourse on orthogonal functions and their properties may be found in a book by Harmuth[8].

2.4 THE FOURIER SERIES AND THE FOURIER TRANSFORM

The Fourier series is probably the most well-known series expansion of a time function by orthogonal functions. Any time function s(t) which is defined in the interval 0 to T and satisfies the conditions as outlined previously, may be represented in the frequency domain by an infinite number of sinusoidal components :

$$s(t) = a_0 + 2\sum_{n=1}^{\infty}(a_n \cos n\omega_0 t + b_n \sin n\omega_0 t) \qquad (2\text{-}11)$$

where

$$a_n = \frac{1}{T}\int_{-T/2}^{T/2} s(t)\cos(n\omega_0 t)\,dt \qquad (2\text{-}12)$$

$$b_n = \frac{1}{T}\int_{-T/2}^{T/2} s(t)\sin(n\omega_0 t)\,dt \qquad (2\text{-}13)$$

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} s(t)\,dt \qquad (2\text{-}14)$$

and ω_0 is considered to be the fundamental frequency, i.e., $\omega_0 = 2\pi/T$.

Equations (2-12) and (2-13) are known as the Euler-Fourier formulas, and the series of (2-11), when a_n and b_n are determined by the Euler-Fourier formulas, is called the Fourier series of s(t). More specifically, a Fourier series is a trigonometric series in which the coefficients a_n and b_n are given, for some absolutely integrable function s(t), by equations (2-12) and (2-13).

Often it is advantageous to use the exponential form of the Fourier series when the magnitudes of the frequency components are of interest. Equation (2-11) can then be simplified to :

$$s(t) = \sum_{n=-\infty}^{\infty} c(n)\exp(jn\omega_0 t) \qquad (2\text{-}15)$$

where $$c(n) = \frac{1}{T}\int_{-T/2}^{T/2} s(t)\exp(-jn\omega_0 t)dt \qquad (2\text{-}16)$$

and $$j = \sqrt{-1}\ .$$

From inspection of equation (2-15), it should be apparent that a periodic time function contains all frequencies that are harmonically related to the fundamental. The magnitude of each of these components is given by $|c(n)|$.

The frequency spectrum of a non-periodic time function can be obtained by the use of the Fourier integral transform which is effectively the Fourier series transform plus a limiting operation. By letting the period T approach infinity, the harmonic component spacing, i.e., $\Delta f = 1/T$,

decreases and approaches a continuous rather than a discrete variable. The Fourier integral transform is generally defined as :

$$F(f) = \int_{-\infty}^{\infty} s(t)\exp(-2\pi jft)dt \qquad (2\text{-}17)$$

where f is the continuous variable. It follows that :

$$s(t) = \int_{-\infty}^{\infty} F(f)\exp(2\pi jft)df \qquad (2\text{-}18)$$

In general, the quantity F(f) will be complex and may be expressed in the form :

$$F(f) = A(f)\exp(jp(f)) \qquad (2\text{-}19)$$

A(f) is considered to be the relative amplitude of the frequency spectrum, and p(f) the initial phase.

2.5 THE DISCRETE FOURIER TRANSFORM

If digital analysis techniques are to be used for analysing a continuous waveform, then it is necessary that the data be sampled (usually at equally spaced intervals of time) in order to produce a time series of discrete samples which can be fed into a digital computer. Such a time series will completely represent the continuous waveform, provided this waveform is frequency band-limited and the samples are taken at a rate which is at least twice the highest frequency present in the waveform. When these

samples are equally spaced, they are known as Nyquist samples. The discrete Fourier transform or DFT, as the name implies, has mathematical properties that are entirely analogous to the Fourier integral transform. In particular, the DFT defines a spectrum of a time series. Multiplication of the transform of two series corresponds to convolving the time series.

The DFT of a time series which consists of N samples is defined by[1] :

$$A(r) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) \exp(-2\pi jrk/N) \qquad (2\text{-}20)$$

$$r = 0, 1, 2, \ldots\ldots, N-1$$

where $A(r)$ = the r^{th} coefficient of the DFT,

$S(k)$ = the k^{th} sample of the time series.

The S(k)'s can be complex numbers and the A(r)'s are almost always complex. For notational convenience, equation (2-20) is often written as :

$$A(r) = \frac{1}{N} \sum_{k=0}^{N-1} S(k) W^{rk} \qquad (2\text{-}21)$$

where $$W = \exp(-2\pi j/N) \qquad (2\text{-}22)$$

---

[1] The definition of the DFT is not uniform. Some authors prefer not to use the factor 1/N; others use $A(r)/\sqrt{N}$ as the DFT coefficients. The definition of equation (2-20) will be followed in this thesis.

The inverse discrete Fourier transform or IDFT is defined by :

$$S(k) = \sum_{r=0}^{N-1} A(r)W^{-rk} \qquad (2\text{-}23)$$

$$k = 0, 1, 2, \ldots\ldots, N-1$$

This form is very similar to that of the DFT. Thus the procedure to compute the DFT is almost identical to that for the DFT. Further details on the DFT and its inverse can be found in reference (9).

2.6 SOME USEFUL PROPERTIES OF ORTHOGONAL TRANSFORMATIONS

Decomposition of an input waveform into a set of coefficients of orthogonal waveforms for generalized spectral analysis is equivalent to a vector-matrix multiplication if implemented on a digital computer. An orthogonal transformation provides a set of coefficients which indicate the degree of correlation of data vectors, with each vector defining the transformation matrix. The coefficients are independent and consequently, higher order approximations can be obtained without recalculating lower-order coefficients. The inverse transformation is particularly simple to implement, being the matrix transpose rather than the inverse, thus eliminating the need for matrix inversion.

In digital communication and coding theory, orthogonal vectors themselves have become important candidates

for signal design and error correcting codes. Orthogonal transformations are entropy preserving in an information theoretic sense, and when applied to certain pattern recognition systems, the transformations provide rotations of the pattern space in which feature selection and maximum variance techniques can be applied. However, possibly one of the most appealing justifications for the use of orthogonal transformations lies in the resulting analysis of the eigenvalues and eigenvectors which define optimum solutions to various systems in the theory of stochastic approximations as well as conventional systems theory. Examples of eigenvector solutions to specific systems are numerous. In the area of stochastic processes, given a sample variance matrix, the eigenvector solution provides a set of orthogonal vectors with associated eigenvalues. The matrix comprised of the eigenvector solution to the covariance matrix is orthogonal and becomes the optimal decomposition of signals from the class of signals described by the covariance matrix. The decomposition is optimum in a mean square error sense, meaning that the truncation of the coefficients of the lowest eigenvalues, for a dimensionality or bandwidth reduction, reconstructs the best mean square approximation to the original signal.

While the eigenvector solution to specific systems or processes provides optimum orthogonal decomposition matrices, calculation of eigenvalues and their associated vectors is a very difficult, if not numerically impossible task. If an

arbitrary N by N matrix is presented, it has a maximum of N^2 degrees of freedom and thus will require N^2 computer operations for vector-matrix multiplication. Consequently, often suboptimum, in a mean square error sense, decompositions are desirable, if they can be efficiently and conveniently generated by digital processes. This implies that the N by N matrix has some additional constraint limiting its maximum number of degrees of freedom. The class of orthogonal transformations which are implementable in $pN \cdot \log_p N$ ($p \ll N$) or less operations, compared to the normal N^2 operations required for arbitrary linear transformations, include the Fourier, Hadamard, Walsh, generalized Walsh, Haar, generalized Haar, and a variety of other unnamed classes of orthogonal transformations. Indeed, the object of this thesis is investigate further the Walsh-Hadamard matrices of dimension N by N, where N is an integer power of two, and to apply these Walsh-Hadamard transformations to the area of speech communications, mainly because of their unique adaptability to digital hardware. The next chapter discusses these specific transforms in more detail.

CHAPTER III

HADAMARD AND WALSH FUNCTIONS

3.1 INTRODUCTION

In recent years, Walsh and Hadamard functions have been increasingly applied to communication engineering and other areas. The Walsh-Hadamard analysis resembles the Fourier technique in both geometric and analytic characteristics. While the Fourier bases are sinusoids with harmonic frequencies, the Walsh-Hadamard bases are square waves with zero-crossings that need not be evenly spaced. It is known that the trigonometric functions sine and cosine are not naturally fitted to the essentially binary operations of digital computers. In contrast, the Walsh-Hadamard bases, being binary in nature, are directly compatible with digital computers and digital circuitry. Since the invention of integrated circuits, digital networks are more acceptable to practical applications. It may be fruitful at this stage to present some background on these functions and, within a digital environment, matrix theory offers a tool for such purposes.

3.2 HADAMARD MATRICES

The Hadamard matrix is a square array of plus and

minus ones, whose rows and columns are orthogonal to each other. Hence the product of the matrix and its transpose is the identity matrix times a constant N. N is the order of the matrix and the lowest value it may assume is two, thus giving the lowest-order Hadamard matrix as :

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3-1}$$

Hadamard matrices of any order are known to exist for N up to 200. However, current interest and the research conducted in this thesis are with those where N is an integer power of two[1]. These matrices may be obtained by a Kronecker (direct) product expansion[33] such that :

$$H_{2N} = \begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix} \tag{3-2}$$

For example,

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & - & 1 & - \\ 1 & 1 & - & - \\ 1 & - & - & 1 \end{pmatrix} \tag{3-3}$$

$$H_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \end{pmatrix} \tag{3-4}$$

[1]These Hadamard matrices are symmetrical and can be related to Walsh matrices of the same order.

3.3 HADAMARD FUNCTIONS[1]

Each row of a Hadamard matrix corresponds to a Hadamard function Had(h,k) for h = 0, 1, 2, ... , N-1, where the rows are numbered sequentially from the top. Each function, except Had(0,k), has the same number of plus ones and minus ones. The number of zero-crossings created by +1 to -1 or -1 to +1 within the interval of orthogonality is known as the sequency of the function[2].

The product of two Hadamard functions yields a third:

$$\text{Had}(i,k)\ \text{Had}(j,k) = \text{Had}(h,k) \qquad (3\text{-}5)$$

The value of h is determined by :

$$h = i \oplus j$$

where h, i, j are expressed in binary numbers and the sign $\oplus$ stands for an exclusive-or or sum-without-carry operation. As an example, considered the multiplication of Had(3,k) by Had(5,k). Using binary numbers 011 for 3 and 101 for 5, one obtains 110 for 6 :

$$\begin{array}{rll} 0\ 1\ 1 & \ldots\ldots & 3 \\ \oplus\ 1\ 0\ 1 & \ldots\ldots & 5 \\ \hline 1\ 1\ 0 & \ldots\ldots & 6 \end{array} \qquad (3\text{-}6)$$

---

[1] The discussions in this paragraph apply to the Walsh functions as well.

[2] The original definition of sequency by H.F.Harmuth[8] was one half the number of zero-crossings. The definition used here eliminates the ambiguity arising from the earlier definition which can have two functions with the same sequency.

The actual multiplication is done bit by bit, if required. Equation (3-5) may be verified for the particular example from equation (3-4).

Since an exclusive-or operation is associative, the multiplication of Hadamard functions must also be associative :

$$\Big[Had(h,k)Had(i,k)\Big]Had(j,k) = Had(h,k)\Big[Had(i,k)Had(j,k)\Big] \quad (3\text{-}7)$$

The product of Had(h,k) and Had(0,k) leaves Had(h,k) unchanged since :

$$h \oplus 0 = h \quad (3\text{-}8)$$

The product of Had(h,k) with itself yields Had(0,k) since :

$$h \oplus h = 0$$

Hence the inverse element is the element itself.

3.4 GENERATION OF HADAMARD FUNCTIONS

Hadamard functions of length $N = 2^m$ are usually generated through the product of Rademacher functions[10] which are an incomplete system of orthogonal square waves with evenly spaced zero-crossings. For instance, with N = 8, the values of the Rademacher functions, namely r_0, r_1 and r_2, for k = 0, 1, 2, ... , 7, are as illustrated in Figure 3-1. The number of zero-crossings for any Rademaher function r_j is $2^{(j+1)}$. A comparision of Figure 3-1 with equation (3-4)

k	0	1	2	3	4	5	6	7
r_0	1	1	1	1	-	-	-	-
r_1	1	1	-	-	1	1	-	-
r_2	1	-	1	-	1	-	1	-

Figure 3-1: 8-length Rademacher Functions

will indicate that r_0, r_1 and r_2 correspond to Had(4,k), Had(2,k) and Had(1,k) respectively. The relationship is straightforward and may be easily applied to higher values of N.

To produce any specfic Had(h,k), h must be expressed in a binary notation. For this example,

$$h = 4h_4 + 2h_2 + h_1$$

where $h_i = 0$ or 1

$i = 1, 2$ or 4.

The correct combination of r_0, r_1 and r_2 is given by the binary 1's of h_4, h_2 and h_1 respectively. For $h_i = 0$, the corresponding r_j is not required as a multiplying factor and, in its place, a value of 1 is assumed for all values of k.

The above technique can easily be adapted to a hardware realization. Figure 3-2 shows a typical circuit. The Rademacher functions are generated by an N-bit binary counter and the desired Hadamard function is obtained by setting the

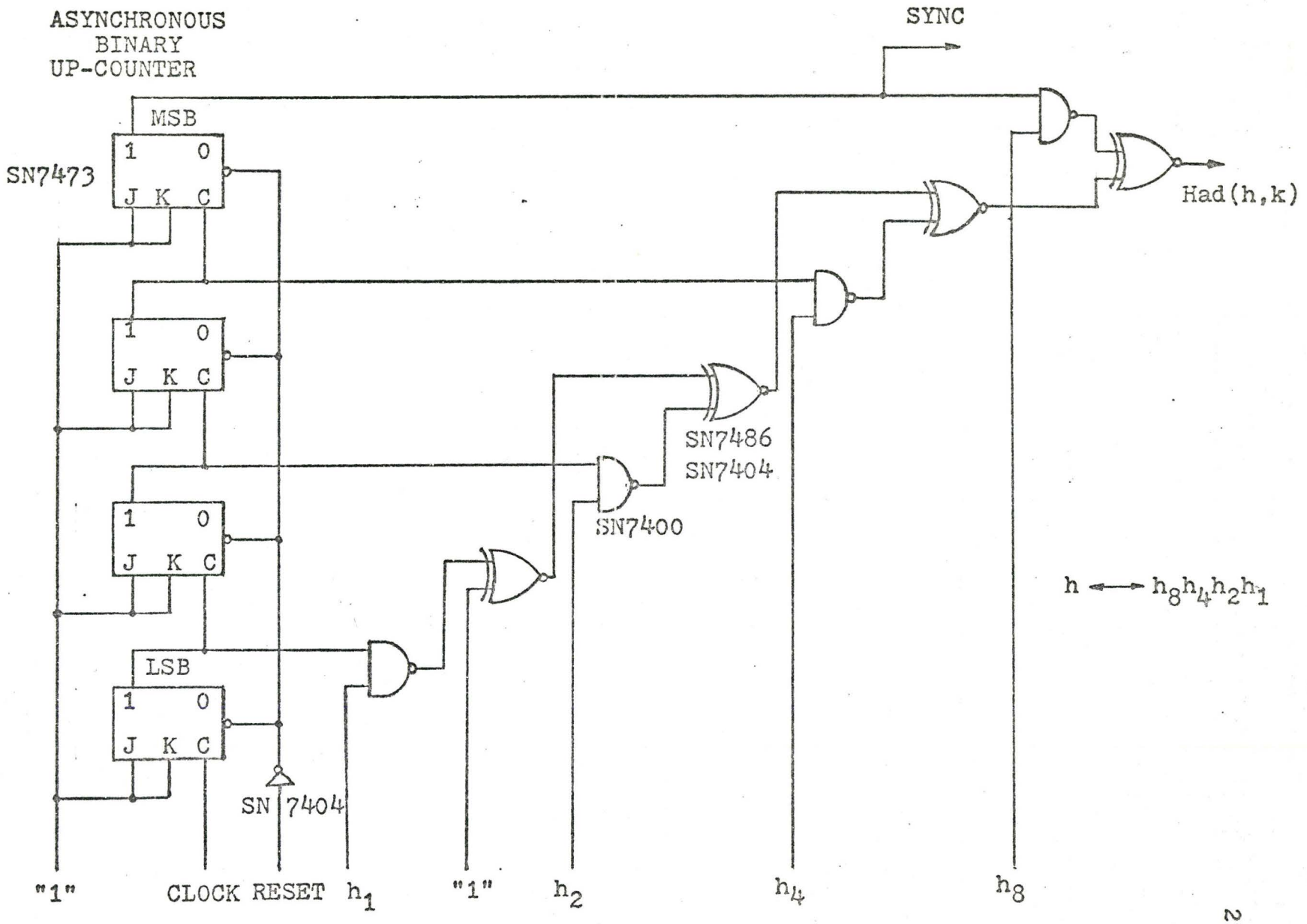


Figure 3-2: Typical 16-length Hadamard Function Generator

appropriate select lines to logical 1 according to the natural binary code. The multiplication is carried out by successive exclusive-or operations. Alternatively, the exclusive-or gates may be replaced by an integrated-circuit package such that if the parity of the outputs from the NAND gates is even, the value of the Hadamard function is +1 or logical 1, and -1 or logical 0 if the parity is odd.

There is a simple technique by which the entire sequence of +1's and -1's for any individual Hadamard function may be deduced without referring to the matrix or Rademacher functions. The initial value of the sequence is +1 since Had(0,k) is always +1. The length of the sequence is determined by the binary representation of h such that for every bit h_i, the sequence is to be extended to the right by i bits. The binary nature of Hadamard functions means that this extension is achieved by doubling its length each time. Accordingly, the bits in this expansion that must be examined are h_1, h_2, h_4, ... , $h_{N/2}$.

As mentioned earlier, if $h_i = 0$, a multiplying factor of +1 is asserted. The previous string of +1's and -1's is to be copied once more to the right. If $h_i = 1$, the corresponding Rademacher function is required in the product; its value is +1 for k = 0, 1, 2, ... , i-1, but -1 for k = i, i+1, i+2, ... , 2i-1. In this case, the sequence is to be expanded by adding on the complement of the existing string of +1's and -1's. This technique is summarized in

Figure 3-3 along with an illustrative example.

Another approach to define Hadamard functions is by means of a multiplicative iteration equation. This method is discussed in the next chapter.

3.5 WALSH FUNCTIONS AND THEIR RELATIONSHIP WITH HADAMARD FUNCTIONS

Walsh functions have the same characteristics as Hadamard functions. In addition, they are ordered in terms of sequency, i.e., a specific Walsh function Wal(w,k) has exactly w zero-crossings. Therefore, given a Hadamard matrix, the Walsh matrix can be obtained by rearranging the rows such that the number of sign changes is in increasing order. For instance,

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & - & - \\ 1 & - & - & 1 \\ 1 & - & 1 & - \end{bmatrix} \qquad (3\text{-}10)$$

$$W_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \end{bmatrix} \qquad (3\text{-}11)$$

The relationship between Hadamard and Walsh functions is illustrated in Figure 3-4 and Figure 3-5.

The conversion from Hadamard to Walsh numbering requires the following steps :

Hadamard function Had(h,k) of length $N = 2^m$.

Let $h = h_1 + 2h_2 + ... + ih_i + ... + (N/2)h_{N/2}$

where $h_i = 0$ or 1,

$i = 1, 2, 4, ... , N/2$

For every h_i, extend the sequence of +1's and -1's to twice its length by adding to its right either of the following :

1) the previous sequence if $h_i = 0$;
2) the complement of the previous sequence if $h_i = 1$.

For example, with N = 8, determine Had(3,k) :

Since $h = h_1 + 2h_2 + 4h_4$,

$\therefore$ Had(3,k) = Had(011,k),

Sequence Generation	Binary Index Value
1	initial value
1 -	$h_1 = 1$
1 - - 1	$h_2 = 1$
1 - - 1 1 - - 1	$h_4 = 0$

Figure 3-3: Generation of Hadamard Functions through Binary Indexing.

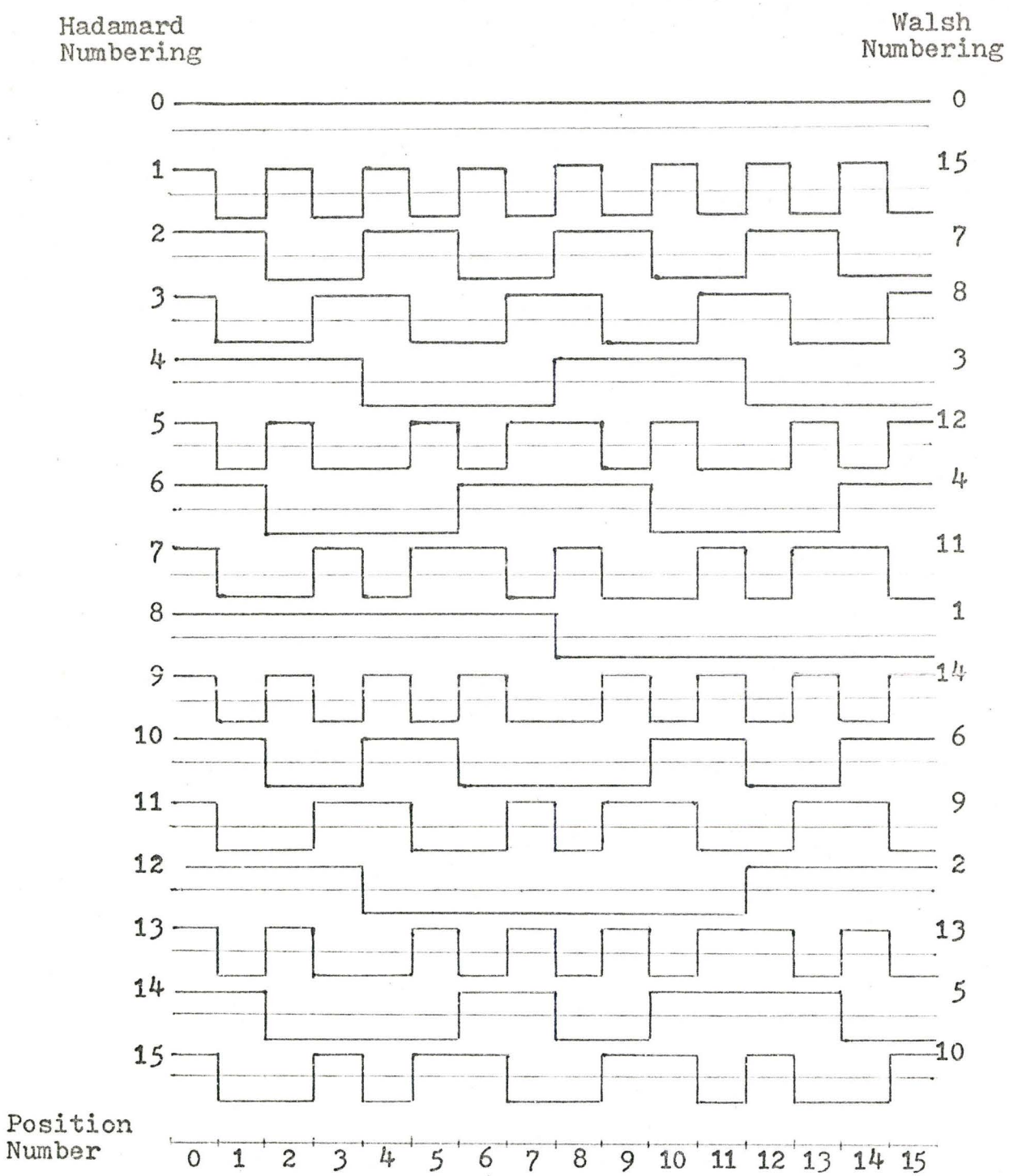


Figure 3-4: Hadamard-Walsh Relation & Ordering

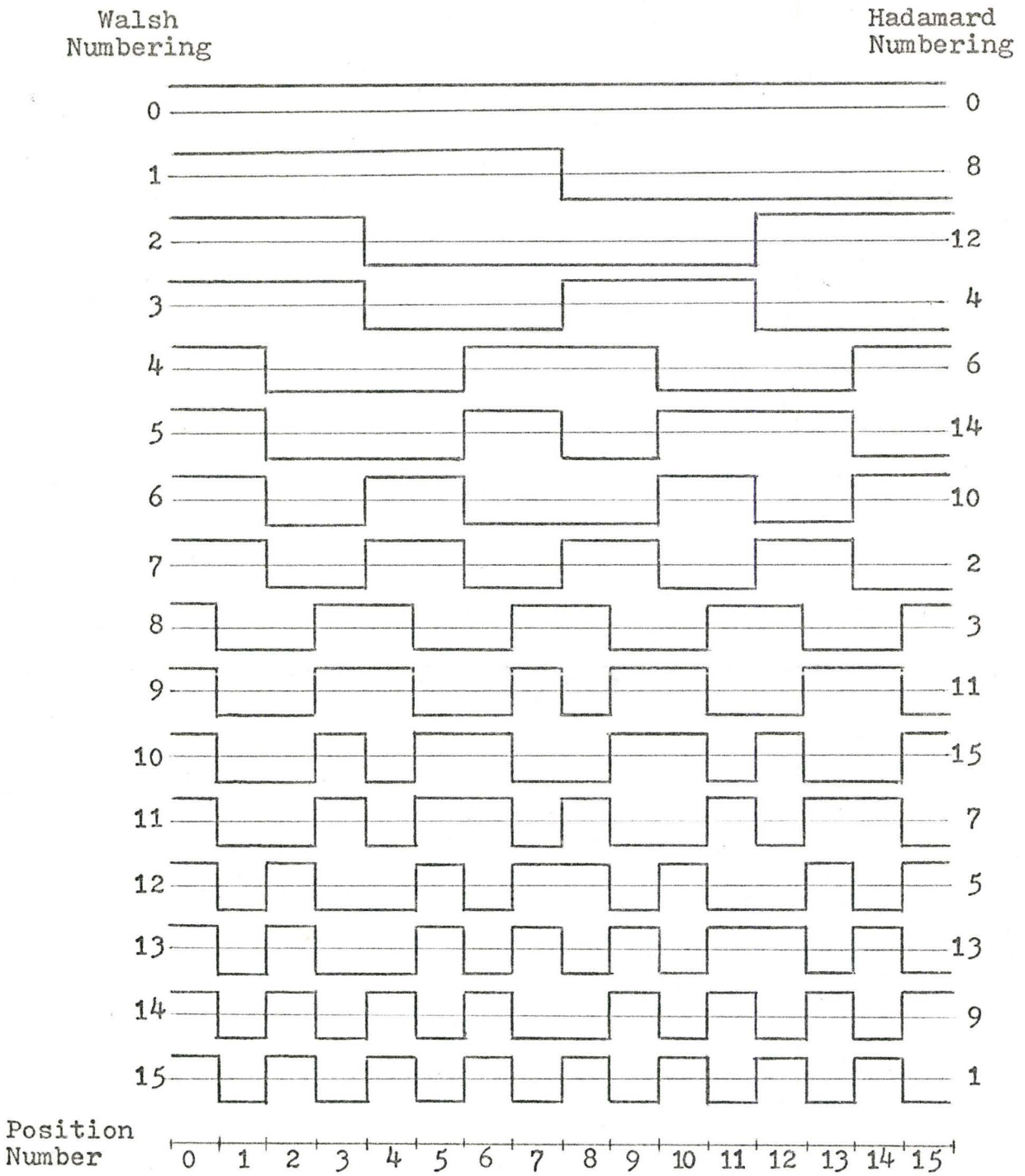


Figure 3-5: Walsh-Hadamard Relation & Ordering

1. Express the Hadamard number h in binary form.
2. Reverse the bits of this binary number.
3. Perform a Gray-to-binary code conversion by carrying out successive exclusive-or operations.

Figure 3-6 shows a typical hardware implementation of such a converter.

The procedure to convert Walsh to Hadamard numbering is as follows :

1. Express the Walsh number w in binary form.
2. Perform a binary-to-Gray code conversion by carrying out modulo-2 addition of the adjacent bits of the binary coded number.
3. Reverse the bits of the result.

A typical Walsh-to-Hadamard code converter based on this procedure is illustrated in Figure 3-7.

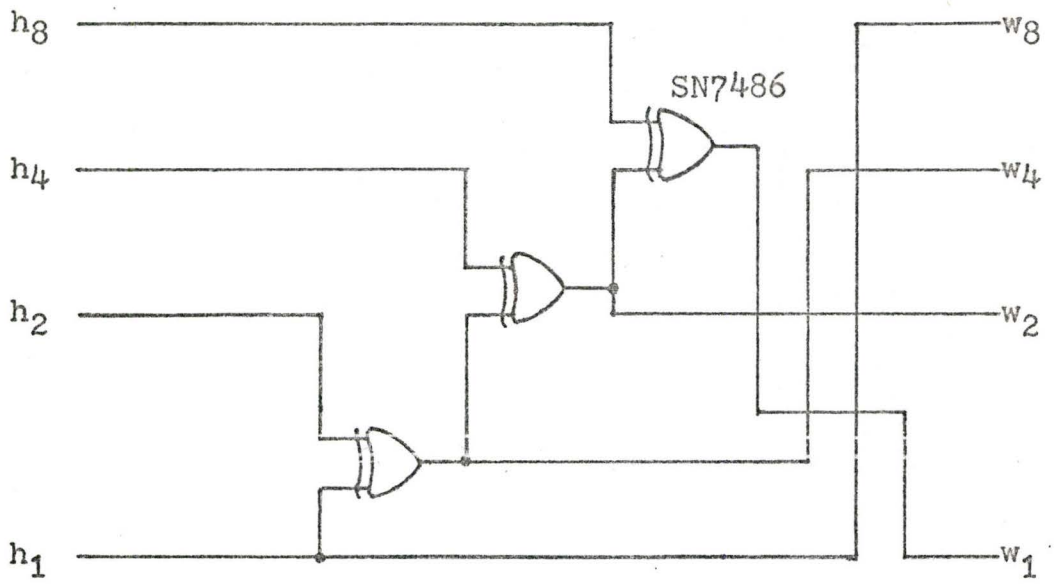
3.6 GENERATION OF WALSH FUNCTIONS

Walsh functions of length $N = 2^m$ may be generated through the product of Rademacher functions. To produce any Wal(w,k), w must be expressed in a binary form. For instance, with $N = 2^3$,

$$w = 4w_4 + 2w_2 + w_1$$

where $w_i = 0$ or 1

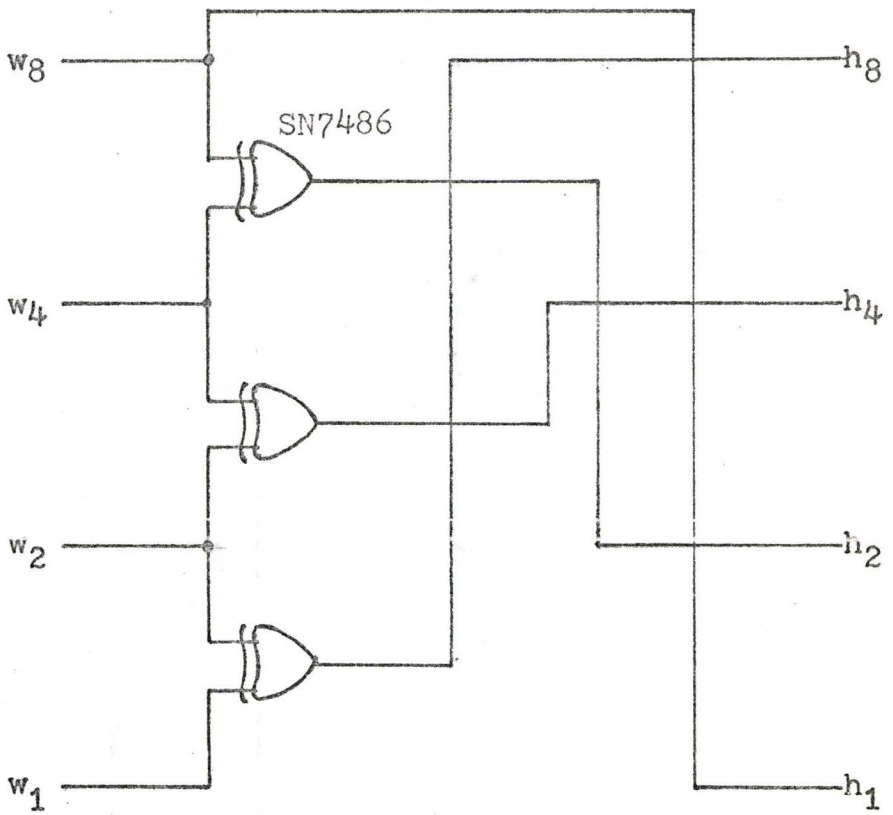
$i = 1, 2$ or 4.



Hadamard no. $h \longleftrightarrow h_8h_4h_2h_1$

Walsh no. $w \longleftrightarrow w_8w_4w_2w_1$

Figure 3-6: Hadamard to Walsh Numbering Converter



Walsh no. $w \longleftrightarrow w_8 w_4 w_2 w_1$

Hadamard no. $h \longleftrightarrow h_8 h_4 h_2 h_1$

Figure 3-7: Walsh to Hadamard Numbering Converter

The binary number is then converted to Gray code by carrying out modulo-2 addition of the adjacent bits. For this example,

$$g_4 = w_4$$
$$g_2 = w_4 \oplus w_2$$
$$g_1 = w_2 \oplus w_1$$

where the sign $\oplus$ denotes modulo-2 addition. The results for this typical example are illustrated in Figure 3-8. The correct combination of Rademacher functions r_0, r_1 and r_2 is determined by the binary 1's of g_1, g_2 and g_4 respectively as explained in the case of Hadamard function generation.

A simple technique to generate Walsh functions is by the multiplication of a basic set of Walsh functions Wal(i,k) where i = 1, 2, 4, ... , N/2. Again, with N = 8, the values of the basic Walsh functions for k = 0, 1, 2, ... , 7 are as illustrated in Figure 3-9. These basic Walsh functions may be produced from an N-bit Gray code counter by assigning the value of +1 to a logical 1 and -1 to a logical 0. Wal(1,k) will then correspond to the output of the most significant bit of the counter whereas Wal(N/2,k) corresponds to the least significant bit of the counter.

To produce any specific Wal(w,k), the Walsh number w is converted to binary form. If $w_i = 1$, Wal(i,k) is required as a multiplying factor; if $w_i = 0$, Wal(i,k) is not involved. The actual multiplication may be performed by

w	w_4	w_2	w_1	g_4	g_2	g_1	g
0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
2	0	1	0	0	1	1	3
3	0	1	1	0	1	0	2
4	1	0	0	1	1	0	6
5	1	0	1	1	1	1	7
6	1	1	0	1	0	1	5
7	1	1	1	1	0	0	4

$g_4 = w_4$ $\qquad$ $0 \oplus 0 = 0$

$g_2 = w_4 \oplus w_2$ $\qquad$ $0 \oplus 1 = 1$

$g_1 = w_2 \oplus w_1$ $\qquad$ $1 \oplus 1 = 0$

$\qquad\qquad\qquad\quad$ $1 \oplus 0 = 1$

Figure 3-8: Example of Gray Code Conversion.

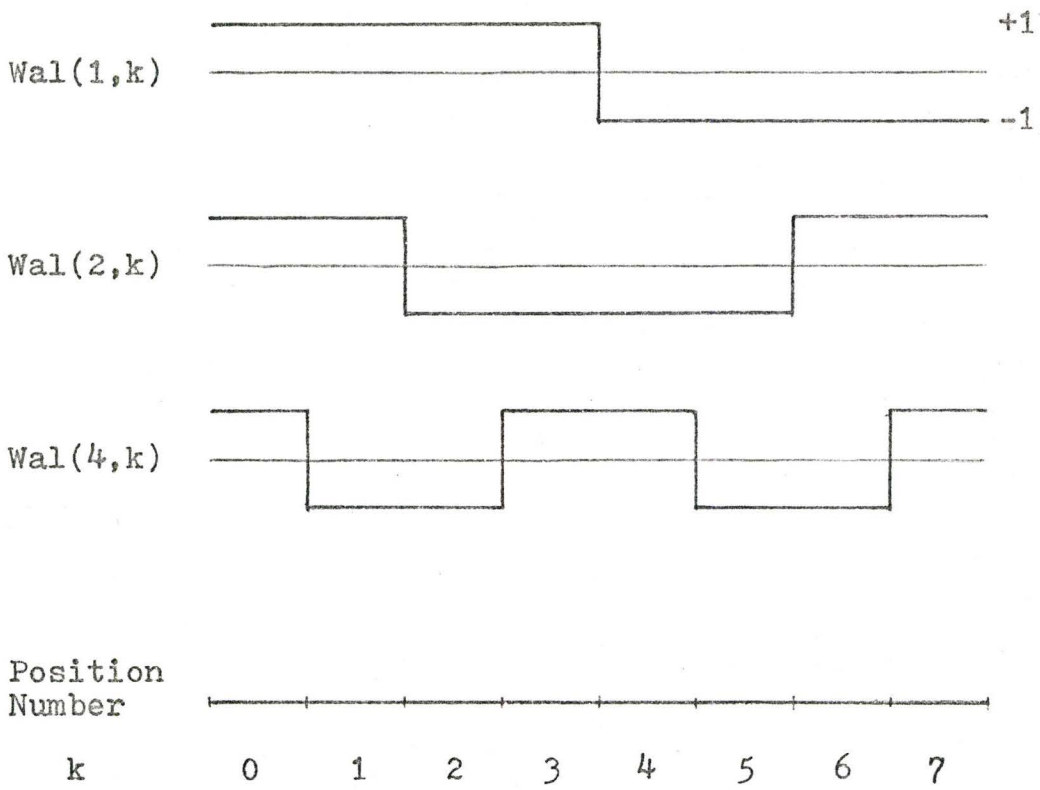


Figure 3-9: Basic Set of 8-length Walsh Functions.

exclusive-or gates or by a parity-check integrated-circuit package. The diagram of a typical Walsh function generator based on this technique is shown in Figure 3-10.

The entire sequence of +1's and -1's for any individual Walsh function may be deduced. This technique was discovered by Swick[11] and was subsequently utilized by Peterson[12] in the design of a Walsh function generator. This method will be explained here with a modified explanation.

It has been shown previously that any Walsh function can be obtained from the product of a basic set of Walsh functions. Of this set, the one that has the most number of sign changes is Wal(N/2,k). Consequently, after the Walsh number w has been expressed in a binary form, the bits to be examined successively are $w_{N/2}$, $w_{N/4}$, ... , w_2 and w_1. The initial value of Wal(w,k) is always +1. The length of Wal(w,k) is determined by the number of bits in w such that for every bit w_i, the sequence of +1's and -1's is to be extended to the right by doubling its existing length.

As mentioned earlier, if $w_i = 0$, the corresponding Wal(i,k) is not required as a multiplying factor. The even symmetry of the basic Walsh functions implies that the sequence is to be expanded such that the end result possesses even symmetry at the centre point. This can be achieved simply by reading the original sequence from left to right but copying from right to left. If $w_i = 1$, Wal(i,k) is involved in the multiplication. Its value is +1 for the

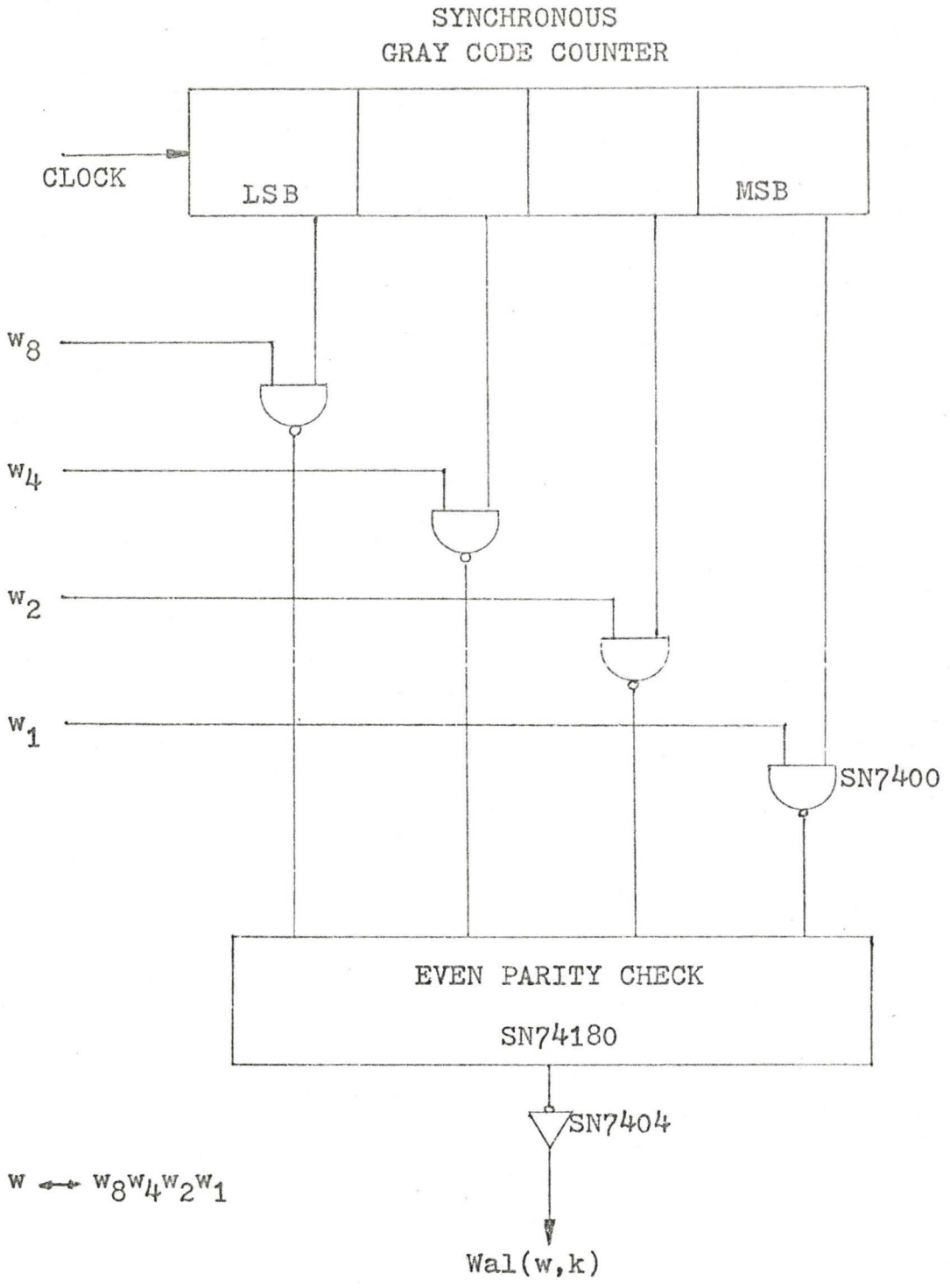


Figure 3-10: Typical 16-length Walsh Function Generator

entire length of the existing sequence but -1 for the length of the sequence to be added on. This, coupled with the symmetry property of Walsh functions, will produce, after the expansion, a string of +1's and -1's that has odd symmetry about its centre point. Two examples to illustrate this technique are shown in Figure 3-11.

3.7 SOME DISCUSSIONS ON HADAMARD AND WALSH FUNCTIONS

Several algorithms to generate Hadamard and Walsh functions have been given. It may be observed that the basic Hadamard functions correspond to the outputs of a natural binary counter, whereas the basic Walsh functions correspond to those of a Gray code counter. Thus these orthogonal systems are naturally important candidates for signal design and coding theory. Walsh functions, in particular, have long been utilized as the Reed-Muller code[13, 33] and recently in signal multiplexing[14].

Walsh functions are ordered in terms of their sequency. Although this is a useful property from a mathematical viewpoint, a fair amount of code conversion is required if these functions are to be generated by a digital computer. Hence Hadamard functions seem to be a better choice in this aspect. In hardware implementation, however, a parity check or exclusive-or operation at the output of a natural binary counter is vulnerable to undesirable voltage levels when operating at a high clock rate. These hazards are not present with a Gray code counter which inherently

Walsh function Wal(w,k) of length $N = 2^m$.

Let $w = w_{N/2}N/2 + ... + iw_i + ... + 2w_2 + w_1$

where $w_i = 0$ or 1,

$i = N/2, N/4, ..., 4, 2, 1.$

For every w_i, extend the sequence of +1's and -1's to twice its length by adding to its right a sequence that has

1) even symmetry about the end point if $w_i = 0$;
2) odd symmetry if $w_i = 1$.

For instance, with $N = 8$, $w = 4w_4 + 2w_2 + w_1$.

Sequence Generation	Binary Index Value	
1	initial value	
1 -	$w_4 = 1$	
1 - 1 -	$w_2 = 1$	
1 - 1 - - 1 - 1	$w_1 = 0$	Wal(6,k)
1 1	$w_4 = 0$	
1 1 - -	$w_2 = 1$	
1 1 - - 1 1 - -	$w_1 = 1$	Wal(3,k)

Figure 3-11: Generation of Walsh Functions through Binary Indexing.

prohibits more than one level change at a time. This property has been discussed by Kitai and Siemens[15]. Another hazard-free design[16] is to obtain the desired sequency by clocking a J-K flip-flop with a rate-multiplier[17].

As digital computers are becoming increasingly indispensible in signal processing techniques, Hadamard and Walsh functions are steadily establishing their status as an important research tool. These binary-valued functions are presently being applied to areas that previously have been tackled predominantly by the Fourier technique. One major potential advantage of Walsh or Hadamard functions over sinusoidal waveforms is their relatively simple computability. This is a major factor of consideration when the process is to be performed numerous times. Another reason favouring the use of pulse-type Walsh-Hadamard waveforms is their suitability to digital circuitry.

CHAPTER IV

HADAMARD AND WALSH TRANSFORMATIONS

4.1 INTRODUCTION

In the area of signal processing, it is often desirable to perform some type of transformation on a function in order to learn more about that function. Certain computational processes of particular interest are discrete orthogonal transformations on input functions in the search for characteristic properties of the transform domain otherwise obscured in the original data. Because of the restriction of digital computation to discrete operations, it is natural to think of digital transformation processes on digitized functions in the concept of matrix algebra. In such a situation, let the input function s(x) be considered as a vector in the x dimension with N samples and let the transformation matrix be H_N. Then the transformation operation can be expressed in matrix notation as :

$$\begin{bmatrix} s(x) \end{bmatrix}_{x \downarrow} \quad \overset{u \longrightarrow}{\begin{bmatrix} H_N(x,u) \end{bmatrix}} = \begin{bmatrix} S(u) \end{bmatrix} \qquad (4\text{-}1)$$

where the vector S(u) is the transformation result of the

vector-matrix multiplication. The vector S(u) can now be expressed in arithmetic form as :

$$S(u) = \sum_{x} s(x)\, H_N(x,u) \qquad (4\text{-}2)$$

If the matrix H_N is constrained to be orthogonal, then the linear transformation can be interpreted as a decomposition of the input data into generalized spectra where each spectral component in the transform domain corresponds to the amount of energy of that spectral orthogonal function within the input data. Using such a concept, the idea of frequency can now be generalized to include transformations of orthogonal functions other than sine and cosine waves. This type of generalized spectral analysis will allow the investigation of specific orthogonal decompositions which could be "better" matched (in possibly an eigenvector sense) to specific purposes and input data classifications[18]. It should be noted that the above discussion can be generalized to multidimensional transforms. An example of the application of such a multidimensional concept is one in which the orthogonal basis vectors in a particular dimension of the data space are matched to the natural eigenvectors of that dimension. In other words, a Fourier transform might be applicable to one dimension of data whereas a Hadamard transform might be applied to data transformations in a second dimension.

The use of high speed digital computers has greatly

enhanced the techniques of signal processing by allowing complex mathematical relations to be implemented in a relatively short span of time. However, certain computational tasks, such as matrix multiplication, still require an inordinate amount of computational complexity as well as storage demands. Because vector-matrix operations are so prevalent in signal analysis problems, it becomes quite useful to attempt a streamlining, where possible, of the computer matrix multiplication process. Probably the most famous of these efforts has resulted in the Fast Fourier Transform (FFT) in which the data-vector-Fourier-matrix multiplication is implemented much more efficiently than normal matrix multiplication requirements[1]. Of somewhat less fame but of great importance is the Fast Hadamard or Walsh Transform (FHT, FWT) which also has resulted in considerable savings in computational and storage requirements for computer implementation[4, 5].

4.2 HADAMARD AND WALSH TRANSFORMS

Being orthonormal, the Hadamard and Walsh functions can be used for a series expansion of a signal. These functions are orthonormal over an interval of time θ and have the value of +1 or -1. To simplify the explanation, only the Hadamard series will be presented in fair detail.

The Hadamard series expansion of a function s(t) is defined as :

$$s(t) = \sum_{h=0}^{\infty} C(h)\ \mathrm{Had}(h,\theta) \qquad (4\text{-}3)$$

where C(h) is the coefficient of the h^{th} Hadamard function. Assuming an interval of orthogonality equal to unity, any specific coefficient C(h) may be determined from the following relationship :

$$C(h) = \int_0^1 s(\theta)\ \mathrm{Had}(h,\theta)\ d\theta \qquad (4\text{-}4)$$

Thus the coefficients can be evaluated by equation (4-4), and the original waveform s(t) can be reconstructed by applying equation (4-3).

If digital analysis techniques are to be used, then it is necessary to sample the signal a finite number of times in order to produce a time series of discrete data which can be fed into a computer. Such a time series will completely represent the continuous waveform, provided this waveform is frequency band-limited and the sampling rate is at least twice the highest frequency present in the waveform.

The discrete Hadamard transform of a time series which consists of N samples can be defined by the following equation :

$$C(h) = \frac{1}{N} \sum_{k=0}^{N-1} S(k)\ \mathrm{Had}(h,k) \qquad (4\text{-}5)$$

$$h = 0,\ 1,\ 2,\ \ldots\ ,\ N\text{-}1$$

where $C(h)$ = h^{th} normalized Hadamard coefficient,

$S(k)$ = discrete samples of the signal,

$Had(h,k)$ = h^{th} Hadamard function.

The inverse Hadamard transform is given by :

$$S(k) = \sum_{h=0}^{N-1} C(h)\, Had(k,h) \qquad (4\text{-}6)$$

$$k = 0, 1, 2, \ldots\ldots, N-1$$

Equations (4-5) and (4-6) are known as the "Hadamard transform pair".

The discrete Walsh transform and its inverse are defined similarly by :

$$C(w) = \frac{1}{N} \sum_{k=0}^{N-1} S(k)\, Wal(w,k) \qquad (4\text{-}7)$$

$$w = 0, 1, 2, \ldots\ldots, N-1$$

$$S(k) = \sum_{w=0}^{N-1} C(w)\, Wal(k,w) \qquad (4\text{-}8)$$

$$k = 0, 1, 2, \ldots\ldots, N-1$$

If an arbitrary N by N matrix is presented, it has a maximum N^2 degrees of freedom. It will be necessary to store a representation of the transformation matrix and N^2 computer operations will be required for vector-matrix multiplication. However, if this matrix has some additional constraint limiting its maximum number of degrees of freedom, then it seems plausible that the computation should require

less than N^2 operations. Indeed, the main interest of this thesis is in Hadamard and Walsh matrices whose N is an integral power of two. With this constraint, the number of operations becomes $2N \times \log_2 N$ which is the reference index for the Radix-2 FFT and FHT. By storing only the parameters necessary for algorithmic generation of the matrix, it is possible to save considerable storage requirement in the computer. Moreover, such a system, being binary in nature, will adapt readily to hardware implementation with digital circuitry.

4.3 COMPUTATION OF THE HADAMARD OR WALSH TRANSFORM BY A MULTIPLICATIVE ITERATION EQUATION

Several algorithms of varying complexity and efficiency have been derived for computing the discrete Walsh or Hadamard transform[4, 5, 18]. Some of these are concerned with time domain or sequency, different from the conditions as formulated in previous chapters, while others may require a lot of code conversion or indices that inherently reduce their usefulness. The algorithms to be presented here are relatively simple and, in addition, the sequency will be ordered according to the conditions as outlined in Chapter III.

The Hadamard functions can be generated from the first two of the set by a multiplicative iteration equation. Using this iteration equation, an efficient Hadamard transform computation algorithm can be derived.

The first two Hadamard functions are defined as :

$$\text{Had}(0,k) = 1 \quad \text{for } k = 0, 1, 2, \ldots\ldots, N-1 \qquad (4\text{-}9)$$

$$\text{Had}(1,k) = \begin{cases} 1 & \text{if } k \text{ is even,} \\ -1 & \text{if } k \text{ is odd.} \end{cases} \qquad (4\text{-}10)$$

The remainder of the set can be generated by a multiplicative iteration equation :

$$\text{Had}(h,k) = \text{Had}(\lfloor h/2 \rfloor, k/2) \cdot \text{Had}(h-2\lfloor h/2 \rfloor, k) \qquad (4\text{-}11)$$

where $\lfloor h/2 \rfloor$ indicates the integer part of $h/2$. It may be observed that

$$h-2\lfloor h/2 \rfloor = \begin{cases} 0 & \text{if } h \text{ is even,} \\ 1 & \text{if } h \text{ is odd.} \end{cases}$$

The iteration procedure is continued until equation (4-11) is reduced to factors whose Hadamard number is either 0 or 1. Hence, for an 8-length Hadamard set, equation (4-11) can be rewritten as :

$$\text{Had}(h,k) = \text{Had}(\lfloor h/4 \rfloor, k/4) \cdot \text{Had}(\lfloor h/2 \rfloor - 2\lfloor h/4 \rfloor, k/2) \cdot \text{Had}(h-2\lfloor h/2 \rfloor, k) \qquad (4\text{-}12)$$

It may be observed that if the number of subdivisions within the interval of orthogonality is halved, then the variable k becomes k/2, and the sequency in Had(h,k/2) is but one half that of Had(h,k). Accordingly,

$$\begin{aligned} \text{Had}(1,k/2) &= 1 \quad \text{if } k = 0,\ 1,\ 4,\ 5 \\ &= -1 \qquad\ \ = 2,\ 3,\ 6,\ 7 \end{aligned} \tag{4-13}$$

Following this reasoning,

$$\begin{aligned} \text{Had}(1,k/2) &= \text{Had}(2,k) \\ \text{Had}(1,k/4) &= \text{Had}(4,k) \end{aligned} \tag{4-14}$$

Let the indices h and k in equation (4-12) be replaced with a set which can have only values 0 and 1, i.e., let

$$\begin{aligned} h &= 4h_4 + 2h_2 + h_1 \\ k &= 4k_4 + 2k_2 + k_1 \end{aligned}$$

With these new indices, dividing h by two and keeping the integer part is equivalent to shifting the binary string to the right by one bit which is then dropped, i.e., if

$$h \longleftrightarrow h_4h_2h_1$$

then

$$h/2 \longleftrightarrow 0h_2h_1.$$

Equation (4-12) can now be expressed as :

$$\begin{aligned} \text{Had}(h_4h_2h_1,\ k_4k_2k_1) = \text{Had}(h_4,k_4)&\cdot\text{Had}(h_2,k_2k_1) \\ &\cdot\text{Had}(h_1,\ k_4k_2k_1) \end{aligned} \tag{4-14}$$

Since

$$\begin{aligned} \text{Had}(h_4,k_4) &= (-1)^{h_4k_4} \\ \text{Had}(h_2,k_2k_1) &= (-1)^{h_2k_2} \\ \text{Had}(h_1,k_4k_2k_1) &= (-1)^{h_1k_1} \end{aligned} \tag{4-15}$$

equation (4-14) can be restated as :

$$\text{Had}(h_4h_2h_1,\ k_4k_2k_1) = (-1)^{h_4k_4}(-1)^{h_2k_2}(-1)^{h_1k_1} \qquad (4\text{-}16)$$

Using binary notation, the discrete Hadamard transform for 8 samples, as defined by equation (4-5), is given by :

$$C(h_4h_2h_1) = \frac{1}{8} \sum_{k_4=0}^{1} \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} S(k_4k_2k_1)\cdot \text{Had}(h_4h_2h_1,\ k_4k_2k_1)$$

$$= \frac{1}{8} \sum_{k_4=0}^{1} (-1)^{h_4k_4} \sum_{k_2=0}^{1} (-1)^{h_2k_2} \qquad (4\text{-}17)$$

$$\sum_{k_1=0}^{1} (-1)^{h_1k_1}\ S(k_4k_2k_1)$$

The array obtained from $\sum_{k_1=0}^{1} (-1)^{h_1k_1}\ S(k_4k_2k_1)$ becomes the data array for the next stage of computation each of which requires four additions and four subtractions. Eventually, the Hadamard coefficients C(h) will be evaluated in their correct sequency order without shuffling the original data S(k) or the end result. The signal flow graph corresponding to equation (4-17) is shown in Figure 4-1.

Since the Hadamard functions are symmetrical with respect to the argument (h,k), i.e.,

$$\text{Had}(h,k) = \text{Had}(k,h) \qquad (4\text{-}18)$$

equations (4-5) and (4-6) are analogous, except for the factor 1/N. Consequently, the procedure to compute the

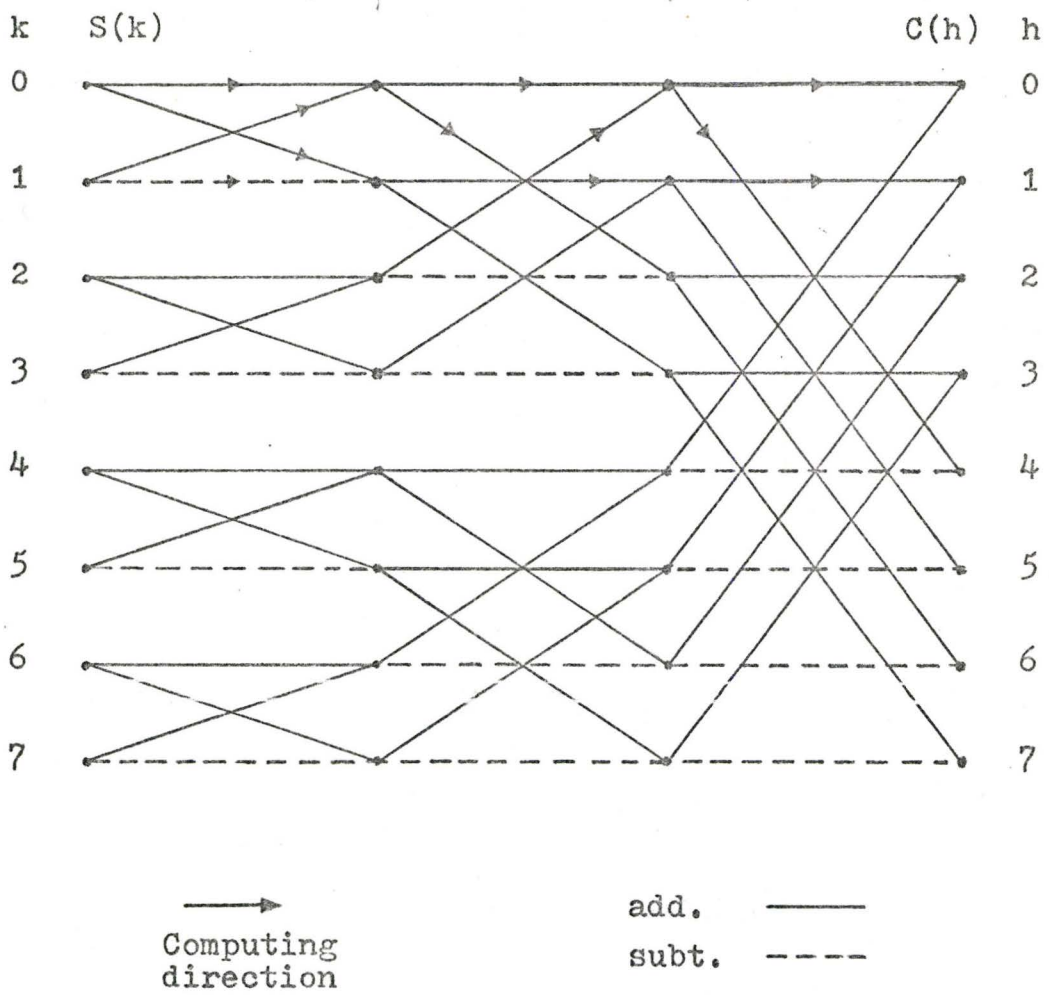


Figure 4-1: Signal Flow Graph for Fast Hadamard Transform.

inverse Hadamard transform can be implemented using an iterative approach similar to equation (4-17).

The Walsh functions can also be generated by a multiplicative iteration equation. The first two Walsh functions are defined as :

$$Wal(0,k) = 1 \quad \text{for all values of } k \tag{4-19}$$

$$\begin{aligned} Wal(1,k) &= 1 \quad \text{for } k = 0, 1, 2, \ldots, N/2-1 \\ &= -1 \qquad\quad = N/2, N/2+1, \ldots, N-1 \end{aligned} \tag{4-20}$$

A set of orthogonal functions whose sequency is in bit-reversed order can be generated by the following equation :

$$\begin{aligned} Wal'(w,k) = {} & Wal'(\lfloor w/4 \rfloor, 4k) \cdot Wal'(\lfloor w/2 \rfloor - 2\lfloor w/4 \rfloor, 2k) \\ & \cdot Wal'(w - 2\lfloor w/2 \rfloor, k) \end{aligned} \tag{4-22}$$

Using binary notation, multiplying k by two is equivalent to shifting the bit string to the left one position. Thus, if

$$k \longleftrightarrow k_4 k_2 k_1$$

then $$2k \longleftrightarrow k_4 k_2 k_1 0 \ .$$

For these 8-length functions, any index such as 2k can be evaluated modulo 8. This implies that any bits above the third bit are deleted. Accordingly,

$$2k(\text{modulo } 8) \longleftrightarrow k_2 k_1 0$$

$$4k(\text{modulo } 8) \longleftrightarrow k_1 00$$

Following this reasoning, equation (4-22) can be expressed

as :

$$Wal'(w_4w_2w_1,\ k_4k_2k_1) = Wal'(w_4, k_100)\cdot Wal'(w_2, k_2k_10) \cdot Wal'(w_1,\ k_4k_2k_1) \qquad (4\text{-}23)$$

The sequency-ordered Walsh functions can be obtained by introducing a multiplying factor to the above set of functions such that the correct symmetry property will be produced. For the specific example where N = 8,

$$\begin{aligned} Wal(0w_20,\ k_4k_2k_1) &= Wal'(w_2,\ k_2k_10)\ (-1)^{w_2k_4} \\ &= (-1)^{w_2k_2}\ (-1)^{w_2k_4} \\ &= (-1)^{w_2(k_4 \oplus k_2)} \end{aligned} \qquad (4\text{-}24)$$

where the sign $\oplus$ denotes modulo-2 addition. Similarly,

$$Wal(w_400,\ k_4k_2k_1) = (-1)^{w_4(k_2 \oplus k_1)} \qquad (4\text{-}25)$$

It has been shown in previous chapters that :

$$Wal(w_4w_2w_1, k) = Wal(w_1, k)\ Wal(w_20, k)\ Wal(w_400, k) \qquad (4\text{-}26)$$

Since

$$Wal(w_1,\ k_4k_2k_1) = (-1)^{w_1k_4} \qquad (4\text{-}27)$$

substituting equations (4-24), (4-25) and (4-27) into (4-26) will produce a completely factorized expression for the 8-length Walsh function set :

$$Wal(w_4w_2w_1,\ k_4k_2k_1) = (-1)^{w_1k_4}\ (-1)^{w_2(k_4 \oplus k_2)} \cdot (-1)^{w_4(k_2 \oplus k_1)} \qquad (4\text{-}28)$$

The fast Walsh transform can be derived from this iteration equation. However, the sampled data $S(k_4k_2k_1)$ must be shuffled initially by reversing the bits of the index k to produce $S(k_1k_2k_4)$, and the Walsh coefficients as defined by equation (4-7) can be evaluated by :

$$C(w_4w_2w_1) = \frac{1}{8} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \sum_{k_4=0}^{1} S(k_1k_2k_4)\,Wal(w_4w_2w_1,\ k_4k_2k_1)$$

$$= \frac{1}{8} \sum_{k_1=0}^{1} (-1)^{w_4(k_2 \oplus k_1)} \sum_{k_2=0}^{1} (-1)^{w_2(k_4 \oplus k_2)} \sum_{k_4=0}^{1} (-1)^{w_1k_4}\ S(k_1k_2k_4) \qquad (4\text{-}29)$$

The array obtained from $\sum_{k_4=0}^{1} (-1)^{w_1k_4} S(k_1k_2k_4)$ becomes the data array for the next stage of computation each of which requires four additions and four subtractions. Eventually the Walsh coefficients C(w) will be produced in their correct order. The signal flow graph corresponding to equation (4-29) is shown in Figure 4-2. The procedure to compute the inverse Walsh transform can be implemented using an iterative approach similar to equation (4-29), with the condition that the data array be shuffled initially.

From Figure 4-2, it may be apparent that the summation of two elements may be stored in the original locations if desired. This "in place" computation property is desirable in terms of memory requirements, and is often regarded as an

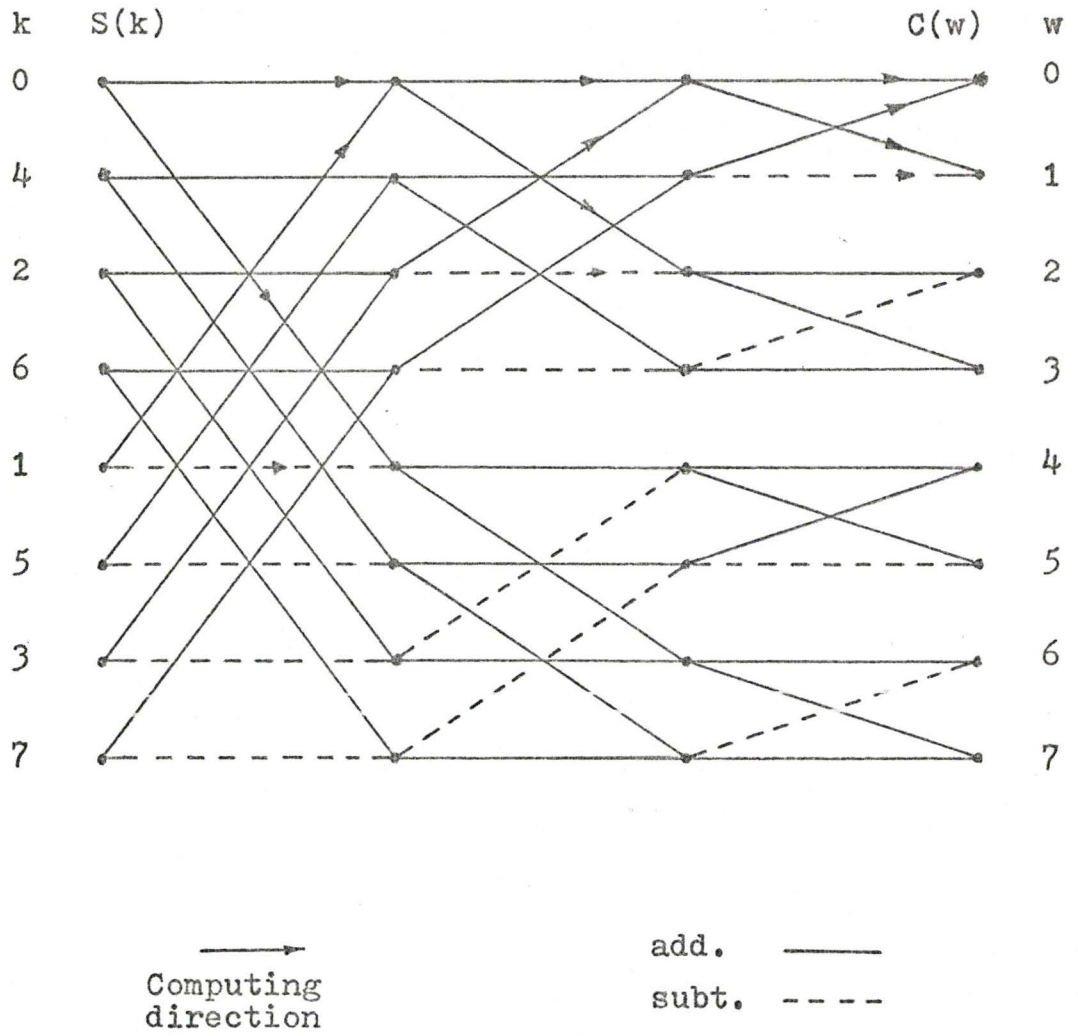


Figure 4-2: Signal Flow Graph for Fast Walsh Transform.

important criterion[9] in determining the efficiency of implementing fast orthogonal transformations.

Walsh functions are ordered in terms of their sequency. This property is not naturally fitted to the essentially binary operations of the digital computer. Consequently, the FWT algorithm is more complicated than the FHT since the Hadamard elements are binary in nature.

4.4 FORMULATION OF THE FAST HADAMARD OR WALSH TRANSFORM BY BINARY INDEXING

The fast Hadamard transform may also be derived from the binary notation of h, where h is the index of the Hadamard coefficient. With N = 8 as an example, h can be expressed in a bi ry form as :

$$h = 4h_4 + 2h_2 + h_1$$

where $h_i = 0$ or 1,

$i = 1, 2$ or 4.

Figure 4-3 shows all the possible values of h. As will be shown, the locations (h, h_i) under columns h_1, h_2 and h_4 are related to the corresponding nodes on the signal flow graph of Figure 4-1.

From the graph, it may be apparent that the terms (any term appearing in the flow graph will be given the general symbol d) concurring at any node differ in their indexing by i. The same result can be obtained by noting where asymmetry occurs under column h_i, as illustrated by

h	h_1	h_2	h_4
0	<u>0</u>	0	0
1	1	<u>0</u>	0
2	<u>0</u>	1	0
3	1	1	<u>0</u>
4	<u>0</u>	0	1
5	1	<u>0</u>	1
6	<u>0</u>	1	1
7	1	1	1

$h \longleftrightarrow h_4 h_2 h_1$

Figure 4-3: Binary Notation of Hadamard Number h.

the horizontal lines in Figure 4-3. The mathematical operator associated with these two terms is determined by $(-1)^{(h,h_i)}$ or $1-2(h,h_i)$. Hence, if $(h,h_i) = 0$, the two terms d_h and d_{h+i} are summed. On the other hand, if (h,h_i) is 1, a subtraction should take place. From Figure 4-1, the correct formulation is $(d_{h-i} - d_h)$.

Any two terms that are summed are also involved in a subsequent subtraction, and the results can be stored in the same memory locations if desired such that :

$$d_h = d_h + d_m \qquad (4\text{-}30)$$

$$d_m = d_h - d_m \qquad (4\text{-}31)$$

where d_h, d_m = sampled data, intermediate results or Hadamard coefficients,

$m = h + i$

$h = 0, 1, 2, \ldots, N-1$, excluding those of m.

For a general-purpose computer, computations are done in a serial fashion. Consequently, "in place" arithmetic requires the modification of equation (4-31) to :

$$d_m = d_h - d_m - d_m \qquad (4\text{-}32)$$

Computation of the inverse Hadamard transform can be formulated from equation (4-30) and (4-32).

In computing the Hadamard transform, the factor of 1/N may be incorporated into each stage of computation

as a factor of 1/2, thereby preventing the possibility of an "overflow" in integer arithmetic. Accordingly, the basic equations for the Hadamard transform are :

$$d_h = (d_h + d_m)/2 \qquad (4\text{-}33)$$

$$d_m = d_h - d_m \qquad (4\text{-}34)$$

For example, with i = 1 :

$$d_0 = (d_0 + d_1)/2 \qquad d_1 = d_0 - d_1$$

$$d_2 = (d_2 + d_3)/2 \qquad d_3 = d_2 - d_3$$

$$d_4 = (d_4 + d_5)/2 \qquad d_5 = d_4 - d_5$$

$$d_6 = (d_6 + d_7)/2 \qquad d_7 = d_6 - d_7$$

The next two stages of computation, for i = 2, 4, may be similarly obtained. The general steps to implement the fast Hadamard transform based on binary notation correspond to the signal flow graph of Figure 4-1, and consequently to the multiplicative iteration equation (4-17), except for the modifications required to accomodate "in place" arithmetic.

The fast Walsh transform can also be derived from the binary notation of the Walsh number w. However, some steps must be undertaken to produce the coefficients that are ordered according to sequency. The first step is to shuffle the data such that their indices will be in bit-reversed order. The binary notation of w is then converted to the Gray code by an exclusive-or operation on w and w/2, an operation that is available to the user for most computers.

For an 8-length Walsh set, the Walsh number w can be expressed as :

$$w = 4w_4 + 2w_2 + w_1$$

for $w_i = 0$ or 1

$i = 1, 2, 4.$

The Gray code of w is given by :

$$g = 4g_4 + 2g_2 + g_1$$

where $g_4 = w_4$

$g_2 = w_4 \oplus w_2$

$g_1 = w_2 \oplus w_1$

and the sign $\oplus$ denotes an exclusive-or or sum-modulo-2 operation. Figure 4-4 shows all the possible values of w and their equivalent Gray code. The locations (w, g_i) under columns g_4, g_2 and g_1 are related to the corresponding nodes on the signal flow graph of Figure 4-2.

Following the same reasoning in explaining the fast Hadamard transform, if $(w, g_i) = 0$, the two terms d_w and d_{w+i} are summed, whereas a subtraction should take place if $(w, g_i) = 1$. The basic equations for computing the Walsh transform for a real signal would be :

$$d_w = (d_w + d_m)/2 \qquad (4\text{-}35)$$

$$d_m = d_w - d_m \qquad (4\text{-}36)$$

where d_w, d_m = sampled data, intermediate results or Walsh coefficients,

w	g_4	g_2	g_1
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Figure 4-4: Walsh Number w and Their Equivalent Gray Code in Binary Notation.

$$m = w + i,$$

$$w = 0, 1, 2, \ldots\ldots, N-1, \text{ excluding } m.$$

These two equations will be adequate for the first stage of computation, i.e., when i = N/2, provided they are repeated N/2 times with w = 0, 1, 2, ..., N/2. For the subsequent stages, however, it seems desirable to introduce two additional equations :

$$d_v = (d_v + d_u)/2 \qquad (4\text{-}37)$$

$$d_u = d_v - d_u \qquad (4\text{-}38)$$

where $u = m + i$

$v = u + i$

This modification will make up for the inherently even symmetry of the Gray code and thereby improve computational efficiency. A FORTRAN subroutine based on this concept is shown in Figure 4-5, with the condition that the sampled data have been shuffled with their indices in bit-reversed order. This assumption is necessary at this stage because bit reversal by itself is a complicated process. It will be shown later that other algorithms can provide greater efficiency if the constraint of "in place" arithmetic is removed.

The inverse Walsh transform can be obtained from a set of equations similar to (4-35) - (4-38). The required changes are shown in Figure 4-5.

```
      SUBROUTINE FWT24I (M,N,N2)
      INTEGER H,Q
      DIMENSION M(N)
C     WALSH TRANSFORM OF A REAL SIGNAL
C     M = SAMPLED DATA, PRESHUFFLED WITH INDICES
C     IN BIT-REVERSED ORDER
C     N = 2**N2 = NO. OF COEFF. PER UPDATE INTERVAL
      H = N/2
      DO 1 J = 1,H
      K = J + H
    5 M(J) = (M(J) + M(K))/2
    1 M(K) = M(J) - M(K)
      Q = 1
      DO 4 NM = 2,N2
      I = 0
      H = H/2
      DO 3 NQ = 1,Q
      DO 2 NH = 1,H
      I = I + 1
      J = I + H
      K = J + H
      L = K + H
    6 M(I) = (M(I) + M(J))/2
    7 M(J) = M(I) - M(J)
    8 M(L) = (M(L) + M(K))/2
    2 M(K) = M(L) - M(K)
    3 I = L
    4 Q = Q*2
      RETURN
      END
```

For inverse Walsh transform, the lines with the same statement numbers are to be replaced by :

```
    5 M(J) = M(J) + M(K)
    1 M(K) = M(J) - M(K) - M(K)
    6 M(I) = M(I) + M(J)
    7 M(J) = M(I) - M(J) - M(J)
    8 M(L) = M(L) + M(K)
    2 M(K) = M(L) - M(K) - M(K)
```

Figure 4-5: FORTRAN IV Subroutine for "in place" Fast Walsh Transform.

4.5 MATRIX FORMULATION OF THE FAST HADAMARD TRANSFORM

It is well known that an efficient way to implement the Hadamard transform of an N-length ($N=2^m$) real signal is to decompose the transform matrix into m factors that have many zero elements, thereby reducing the number of arithmetic operations. For example,

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \end{bmatrix}$$

$$= G_0 \cdot G_1 \cdot G_2 \qquad (4\text{-}39)$$

$$= P \cdot P \cdot P \qquad (4\text{-}40)$$

$$= Q \cdot Q \cdot Q \qquad (4\text{-}41)$$

where

$$G_0 = \begin{bmatrix} 1 & 1 & . & . & . & . & . & . \\ 1 & - & . & . & . & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & 1 & - & . & . & . & . \\ . & . & . & . & 1 & 1 & . & . \\ . & . & . & . & 1 & - & . & . \\ . & . & . & . & . & . & 1 & 1 \\ . & . & . & . & . & . & 1 & - \end{bmatrix} \qquad (4\text{-}42)$$

$$G_1 = \begin{bmatrix} 1 & . & 1 & . & . & . & . & . \\ . & 1 & . & 1 & . & . & . & . \\ 1 & . & - & . & . & . & . & . \\ . & 1 & . & - & . & . & . & . \\ . & . & . & . & 1 & . & 1 & . \\ . & . & . & . & . & 1 & . & 1 \\ . & . & . & . & 1 & . & - & . \\ . & . & . & . & . & 1 & . & - \end{bmatrix} \qquad (4\text{-}43)$$

$$G_2 = \begin{pmatrix} 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & . & 1 & . & . & . & 1 \\ 1 & . & . & . & - & . & . & . \\ . & 1 & . & . & . & - & . & . \\ . & . & 1 & . & . & . & - & . \\ . & . & . & 1 & . & . & . & - \end{pmatrix} \qquad (4\text{-}44)$$

$$P = \begin{pmatrix} 1 & 1 & . & . & . & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & . & . & 1 & 1 & . & . \\ . & . & . & . & . & . & 1 & 1 \\ 1 & - & . & . & . & . & . & . \\ . & . & 1 & - & . & . & . & . \\ . & . & . & . & 1 & - & . & . \\ . & . & . & . & . & . & 1 & - \end{pmatrix} \qquad (4\text{-}45)$$

$$Q = \begin{pmatrix} 1 & . & . & . & 1 & . & . & . \\ 1 & . & . & . & - & . & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & 1 & . & . & . & - & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & 1 & . & . & . & - & . \\ . & . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . & - \end{pmatrix} \qquad (4\text{-}46)$$

Since the non-zero elements of 1 and -1 in each factor can be formulated into equations as arithmetic operators of + and -, only additions and subtractions will be required. The indices of the terms to be summed are determined by the positioning of the 2 non-zero elements on each row. The number of factors is given by $\log_2 N$ and the number of arithmetic operations per factor is $2 \times N$. Hence the total number of operations required to evaluate the Hadamard coefficients, barring the normalizing factor of $1/N$, becomes $2N \times \log_2 N$ rather than N^2 in the normal case.

Matrix Q is the transpose of P, or vice versa. Consequently the characteristics of both are analogous. It may be sufficient to describe the properties of P, and deduce those of Q on similar guidelines.

It may be observed from equation (4-45) that the product of matrix P and a one-dimensional array involves merely additions and subtractions of data that are adjacent, and the storing of the results in memory registers that are N/2 locations apart. In the typical computer, mathematical calculations are done in a serial fashion. Hence, in software, at least one other array is required as a buffer, as illustrated by the array LR in the FORTRAN subroutine PHTR2 shown in Figure 4-6. Each factor P corresponds to one stage of computation. After m stages, the Hadamard coefficients are located in the original data array.

Equation (4-39) may be interpreted as the building up of the final matrix from the lowest-order Hadamard matrix by successive multiplication. Since the factors are orthogonal, the results generated through additions and subtractions may be stored in the original data array. Using this scheme, the computations are done "in place", thereby saving memory, though at a slight increase of computer time, as an additional iteration step is required to accommodate the change in the pattern of the non-zero elements. The FORTRAN subroutine FHTB2 of Figure 4-7 is a software implementation of the FHT algorithm based on equation (4-39).

```
      SUBROUTINE PHTR2 (JR,LR,N,M)
      DIMENSION JR(N), LR(N)
C     JR = DATA ARRAY
C     LR = BUFFER ARRAY
C     N  = 2**M ; NO, OF DATA
      NH = N/2
      LF = 1
      M1 = AND(M,1)
      IF (M1.NE.0) GO TO 20
      MM = M
      K  = NH
      GO TO 22
   20 MM = M-1
      K  = NH/2
   22 LL = K
   30 JF = LF+1
      IN = 1
      DO 4 I = 1,MM
      J = JF
      IF (IN.LT.0) GO TO 2
      DO 1 L = LF,LL
    5 LR(L) = (JR(J-1) + JR(J))/2
    6 LR(L+K) = LR(L) - JR(J)
    1 J = J+2
      GO TO 4
    2 DO 3 L = LF,LL
    7 JR(L) = (LR(J-1)+LR(J))/2
    8 JR(L+K) = JR(L)-LR(J)
    3 J  = J+2
    4 IN = -IN
      IF (M1.EQ.0) RETURN
      IF (M1.LT.0) GO TO 40
      M1 = -1
      LF = LF + NH
      LL = LL + NH
      GO TO 30
   40 DO 9 I = 1,NH
      J = I + NH
   10 JR(I) = (JR(I)+JR(J))/2
    9 JR(J) = JR(I)-JR(J)
      RETURN
      END
```

Figure 4-6: FORTRAN IV Subroutine for Fast Hadamard Transform Using Identical Factorized Matrices.

```
      SUBROUTINE FHTB2 (MR,N,M)
      DIMENSION MR(N)
C     HADAMARD TRANSFORM OF A REAL SIGNAL
C     MR = SAMPLED DATA
C     N = NO. OF DATA
C     2**M = NO. OF COEFF. PER TIME INTERVAL
      L = N
      K = 1
      DO 3 NM = 1,M
      I = 0
      L = L/2
      DO 2 NL = 1,L
      DO 1 NK = 1,K
      I = I + 1
      J = I + K
    4 MR(I) = (MR(I) + MR(J))/2
    1 MR(J) =  MR(I) - MR(J)
    2 I = J
    3 K = K*2
      RETURN
      END
```

For inverse Hadamard transform, the lines with the same statement numbers are to be replaced by :

```
    4 MR(I) = MR(I) + MR(J)
    1 MR(J) = MR(I) - MR(J) - MR(J)
```

Figure 4-7: FORTRAN IV Subroutine for Fast Hadamard Transform Using "in place" Computation.

4.6 MATRIX FORMULATION OF THE FAST WALSH TRANSFORM

Reordering of the Hadamard coefficients to generate their Walsh counterpart by computer operation is very time consuming. Instead, the fast Walsh transform of a real signal may be derived from a matrix factorization similar to the FHT as shown below :

$$W_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & - & - & - & - \\ 1 & 1 & - & - & - & - & 1 & 1 \\ 1 & 1 & - & - & 1 & 1 & - & - \\ 1 & - & - & 1 & 1 & - & - & 1 \\ 1 & - & - & 1 & - & 1 & 1 & - \\ 1 & - & 1 & - & - & 1 & - & 1 \\ 1 & - & 1 & - & 1 & - & 1 & - \end{pmatrix} \tag{4-47}$$

$$= Y_0 \cdot Y_1 \cdot Y_2$$

$$= Z_0 \cdot Z_1 \cdot Z_2 \tag{4-48}$$

where

$$Y_0 = \begin{pmatrix} 1 & 1 & . & . & . & . & . & . \\ 1 & - & . & . & . & . & . & . \\ . & . & 1 & 1 & . & . & . & . \\ . & . & 1 & - & . & . & . & . \\ . & . & . & . & 1 & 1 & . & . \\ . & . & . & . & 1 & - & . & . \\ . & . & . & . & . & . & 1 & 1 \\ . & . & . & . & . & . & 1 & - \end{pmatrix} \tag{4-49}$$

$$Y_1 = \begin{pmatrix} 1 & . & 1 & . & . & . & . & . \\ 1 & . & - & . & . & . & . & . \\ . & 1 & . & - & . & . & . & . \\ . & 1 & . & 1 & . & . & . & . \\ . & . & . & . & 1 & . & 1 & . \\ . & . & . & . & 1 & . & - & . \\ . & . & . & . & . & 1 & . & - \\ . & . & . & . & . & 1 & . & 1 \end{pmatrix} \tag{4-50}$$

$$Y_2 = \begin{bmatrix} 1 & . & . & . & 1 & . & . & . \\ 1 & . & . & . & - & . & . & . \\ . & 1 & . & . & . & - & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & 1 & . & . & . & - & . \\ . & . & . & 1 & . & . & . & - \\ . & . & . & 1 & . & . & . & 1 \end{bmatrix} \qquad (4\text{-}51)$$

$$Z_0 = \begin{bmatrix} 1 & . & . & . & 1 & . & . & . \\ 1 & . & . & . & - & . & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & 1 & . & . & . & - & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & 1 & . & . & . & - & . \\ . & . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . & - \end{bmatrix} \qquad (4\text{-}52)$$

$$Z_1 = \begin{bmatrix} 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & 1 & . & . & . & - & . & . \\ 1 & . & . & . & - & . & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . & - \\ . & . & 1 & . & . & . & - & . \end{bmatrix} \qquad (4\text{-}53)$$

$$Z_2 = \begin{bmatrix} 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & 1 & . & . \\ . & . & 1 & . & . & . & 1 & . \\ . & . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . & - \\ . & . & 1 & . & . & . & - & . \\ . & 1 & . & . & . & - & . & . \\ 1 & . & . & . & - & . & . & . \end{bmatrix} \qquad (4\text{-}54)$$

Equation (4-47) may be interpreted as the building up of the final matrix from the lowest-order Walsh matrix by successive multiplication. A close look at matrix Y_0 in equation (4-49) reveals that it is identical to matrix G_0 in (4-42). The other matrices Y_1 and Y_2 correspond to G_1

and G_2 with the rows rearranged to produce Walsh coefficients in their correct order. Since these matrices are not orthogonal, a buffer array is required in addition to the data array for computing the FWT. The coefficients would eventually be located in the buffer if $m(N=2^m)$ is odd. However, for m odd, the "in place" summation technique for the first computation stage which corresponds to vector-matrix multiplication with factor Y_0, effectively makes m look even, guaranteeing that the coefficients end up in the data array. For matrices Y_1 and Y_2, an obvious gain in computer time is to utilize four equations for the summation process. The resultant FORTRAN subroutine based on equation (4-47) is shown in Figure 4-8. The subroutine FWTR2 of Figure 4-9 is based on the factorized form in equation (4-48) by Ulman[5].

4.7 HARDWARE IMPLEMENTATION OF THE FAST HADAMARD AND WALSH TRANSFORMS

In a parallel-operating system, the same memory elements may be utilized to hold the original sampled data, intermediate results, and eventually, after m operations, the required Hadamard coefficients in their correct order. A preliminary design shows that it is possible to use complement arithmetic along with M.S.I. adder packages available, and D-type flip-flops arranged as registers for the sampled data as well as the coefficient values.

A general rule for recycling of the results from the

```
      SUBROUTINE FWTB2 (MR,NR,N,M)
      DIMENSION  MR(N), NR(N)
      IN = AND(M,1)
      IF (IN.EQ.0) GO TO 2
      DO 1 K = 1, N, 2
      L = K + 1
      MR(K) = (MR(K) + MR(L))/2
    1 MR(L) = MR(K) - MR(L)
      GO TO 4
    2 IN = -1
      DO 3 K = 1, N, 2
      L = K + 1
      NR(K) = (MR(K) + MR(L))/2
    3 NR(L) = NR(K) - MR(L)
    4 NH = N/2
      K = 1
      DO 8 NM = 2,M
      KH = K
      K = K*2
      KP = K + 1
      L = 1
      I = 1
      IN = -IN
      NH = NH/2
      DO 8 NL = 1,NH
      DO 7 NK = 1,KH
      IF (IN.GT.0) GO TO 5
      MR(L)    = (NR(I) + NR(I+K))/2
      MR(L+1)  =  MR(L) - NR(I+K)
      MR(L+2)  = (NR(I+1) - NR(I+KP))/2
      MR(L+3)  =  MR(L+2) + NR(I+KP)
      GO TO 6
    5 NR(L)    = (MR(I) + MR(I+K))/2
      NR(L+1)  =  NR(L) - MR(I+K)
      NR(L+2)  = (MR(I+1) - MR(I+KP))/2
      NR(L+3)  =  NR(L+2) + MR(I+KP)
    6 I = I + 2
    7 L = L + 4
    8 I = I + K
      RETURN
      END
```

Figure 4-8: FORTRAN IV Subroutine for Fast Walsh Transform.

```
      SUBROUTINE FWTR2(MR,NR,N,M)
      DIMENSION MR(N),NR(N)
      KH = N
      K = KH/2
      LH = 1
      IN = 1
      DO 4 NM = 1,M
      IN = -IN
      I = 1
      J = 0
      LD = LH*2
      LH = KH/2
      DO 3 NK = 1,KH
      L = LD
      DO 2 NL = 1,LH
      L = L-1
      J = J+1
      IF (IN.LT.0) GO TO 1
    6 MR(I) = (MR(J) + NR(J+K))/2
    7 MR(I+L) = MR(I) - NR(J+K)
      GO TO 2
    1 NR(I) = (MR(J) + MR(J+K))/2
    8 NR(I+L) = NR(I) - MR(J+K)
    2 I = I+1
    3 I = I+LH
    4 LH = LD
      IF (IN.GT.0) RETURN
      DO 5 I = 1,K
      MR(I) = NR(I)
    5 MR(I+K) = NR(I+K)
      RETURN
      END
```

For the inverse Walsh transform, the lines with the same statement numbers are to be replaced by :

```
    6 MR(I) = NR(J) + NR(J+K)
    7 MR(I+L) = NR(J) - NR(J+K)
    1 NR(I) = MR(J) + MR(J+K)
    8 NR(I+L) = MR(J) - MR(J+K)
```

Figure 4-9: FORTRAN IV Subroutine for Walsh Transform Based on Ulman's Algorithm (reference 5).

output of the adders, based on matrix P in equation (4-45) is thus given by :

$$d_j = (d_k + d_{k+1})/2 \qquad (4\text{-}55)$$

$$d_{j+N/2} = (d_k - d_{k+1})/2 \qquad (4\text{-}56)$$

where $j = 0, 1, 2, \ldots\ldots, N/2-1$

for $k = 0, 2, 4, \ldots\ldots, N-2$

Figure 4-10 demonstrates a proposed digital circuit that will implement the fast Hadamard transform for the determination of 8 coefficients. The expansion to a circuit that will obtain the first 64 coefficients is straightforward and will have the same general structure as the circuit of Figure 4-10. Under the assumption that a 7-bit coefficient value is sufficient, with the eighth bit handling transient arithmetic overflow, the cost per coefficient, with off-the-shelf components, would be under $12.00 at today's prices.

For $N = 2^6$, (i.e., 64 coefficients) the circuit need only be clocked 6 times, with adequate time between clock pulses to allow the output of the adders to settle. With modern integrated-circuit packages, this should be accomplished in less than 200 nanoseconds. A circuit for obtaining 128 coefficients would take about 250 nanoseconds.

The Hadamard coefficients may be rearranged to obtain the Walsh ordering. In a hardware implementation to obtain the Walsh coefficients, a straightforward approach would be to add a new set of registers at the output of the adders to hold the Walsh coefficients. This is illustrated in

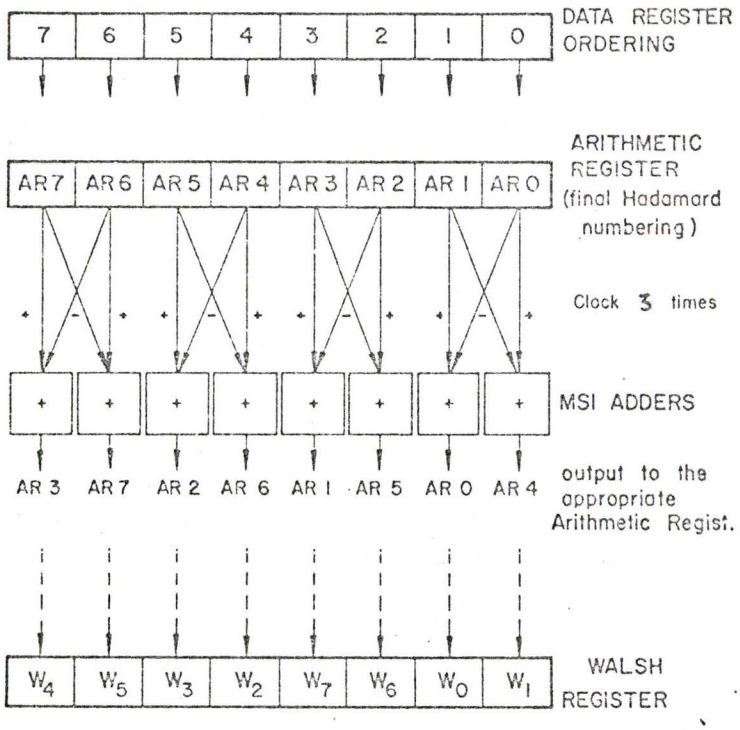


Figure 4-10: Typical Parallel Array for 8 Hadamard or Walsh Coefficients.

Figure 4-10.

Note that the forward and the inverse transforms differ only by a simple factor of two. Accordingly, a shifting of the bit lines returning to the storage registers by one bit will allow the inverse transform to be performed by the same hardware. This circuit has not been implemented at present. However, its versatility and simplicity have initiated the design of a special-purpose processor for analysing or synthesizing an audio signal in the Walsh or Hadamard domain. This processor will be discussed in the next chapter.

CHAPTER V

SPEECH PROCESSING WITH WALSH-HADAMARD TRANSFORM

5.1 INTRODUCTION

Many communication systems convert analog signals into a digital form for processing and transmission. A main topic in such a system is to determine an encoding scheme which will minimize the number of coding symbols required to describe the source signal. This coding method must not degrade the quality of the signal beyond certain fidelity limits. Furthermore, the encoded data must not be overly sensitive to channel errors. A great amount of investigation has been performed in the search for source encoding systems [19,20,21]. Unfortunately, most of the systems developed either do not exhibit satisfactory performance, or are too difficult to implement.

An encoding scheme widely used nowadays for the transmission of digital signals is pulse-code modulation or PCM. The time function is sampled at a uniform rate which is at least twice the highest frequency present in the waveform. The samples are then encoded in a binary code which provides an adequate number of quantization levels, selected in accordance with some appropriate rate-distortion criteria, such as signal-to-noise ratio or mean square error. However, conventional PCM techniques require high bit rates in order to transmit the signals with a given level of distortion.

It is well known that the energy of a time function is conserved in the transform domain if the time function is expanded into a complete orthonormal set[22]. This implies that if a few of the coefficients are of large magnitude in terms of absolute values, then many of the other coefficients must be of very low magnitude and conceivably may be discarded to obtain a bit rate reduction. The amount of success in data compression through proper coding of the "dominant" coefficients depends on the suitability of the transformation for representing the signal. The complexity of the implementation with respect to other available techniques is also an important factor to be considered.

Orthogonal transformations of signals as a means to reduce the bit rate necessary for data transmission have been modeled and exploited to various degrees by a number of investigators. Discrete forms of the Harhunen-Loève, Slant, Fourier, Walsh-Hadamard, and Haar transforms have been examined for their effectiveness in data compression, in a mean square error sense or signal-to-noise ratio, and reported by independent researchers[23-26]. Their results tend to indicate that the K-L transform[27,28] yields an orthogonal function set that optimally expresses the information content of the signal, whereas the other transforms are ranked in the order that appears above. Because the orthogonal function set in the K-L method is determined from the covariance matrix of the signal to be represented, this

method should and does yield the lowest mean square error for a given information rate. However, when analysing speech signals, the K-L transform matrix has to be re-formulated whenever the covariance matrix changes during the process. Also, this transform does not possess a fast computational algorithm and is not feasible to be implemented on a computer of limited capability. The Haar transform, on the other hand, has the attribute of an extremely efficient computational algorithm[29], but results in a relatively large coding error.

The Slant transform was introduced by Shibata and Enomoto[30] in 1970 and has been augmented by Pratt[31] who has shown that the variance function for the Slant transform is reasonably close to that of the K-L transform. Transformation with this orthonormal function set requires additions and multiplications with irrational numbers. To date, signal processing with the Slant transform has been limited to only a few investigators.

The most widely used technique in bit rate reduction at present is the Fourier transform. High speed FFT processors are available commercially to aid in real-time analysis of speech or images. The Walsh-Hadamard transform has also received considerable attention because of its compatibility to digital processing and ease of implementation, especially when small computers with limited facilities are used. The results of speech synthesis presented by Boesswetter[7], and Campanella and Robinson[6] had demonstrated that dominant

term synthesis from the Walsh domain was possible, and with good quality.

5.2 COMPUTER FACILITY AND IMPLEMENTATION

A block diagram of the CDC-1700 computer system and the apparatus used in the speech research is illustrated in Figure 5-1. The present configuration of the system is such that the input/output bus can only handle A/D or D/A conversion one at a time, in a buffered mode. Consequently, real-time operation is not possible, although this can be contemplated with better computer facilities soon to be implemented, or with the aid of special-purpose hardware.

The original sentences that had been processed were taken from a master tape supplied by the 1972 Conference on Speech Communication and Processing. Since speech is considered to be band limited to 3.6 KHz, the audio signal was sampled at an 8 KHz, and the data was formatted into 14-bit words by the A/D converter. These digital samples were stored on a disk capable of holding 1.5 million words. Only 12 cylinders, each with a capacity of 15,360 words, out of a total 99 were used, which corresponds to approximately 23 seconds of speech being processed on one pass.

At the end of the input process, the speech data on the disk was transferred to the computer memory in blocks of 3,072 samples. Every group of 64 words was transformed into the Hadamard (or Walsh) domain. As the time between successive

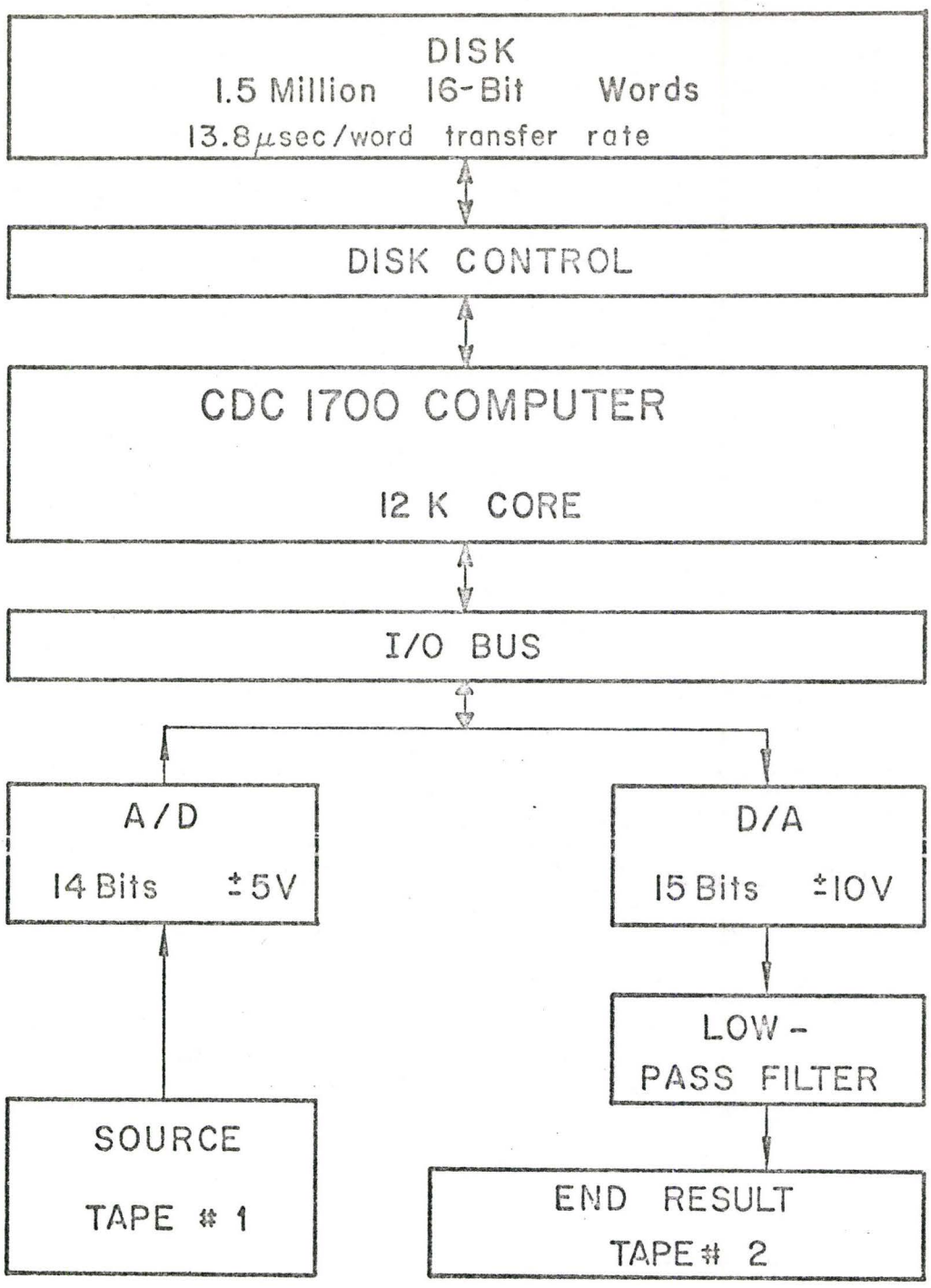


Figure 5-1: Block Diagram of CDC-1700 Computer Facility for Speech Processing.

samples was 125 microseconds, each group of data words corresponded to a time duration or window size of 8 milliseconds. The choice of window size was not related to any specific features of speech, such as pitch period or formant structure, but was based on the effective spectral resolution required to represent the characteristics of the spectrum over an ensemble of talkers and words. The frequency of resolution in this case is 125 Hz and is adequate for the analysis of long-term spectral distribution of speech. Also, the assignment of 64 which is an integral power of two, enables efficient algorithms to compute the orthogonal coefficients to be applied directly.

The dominant (in terms of the largest absolute value) 8 (or 4) coefficients were selected through another algorithm. This algorithm is a straightforward search, comparison and retention of the larger coefficient magnitudes, and their corresponding coefficient numbers are generated. The computer memory designated for the coefficients was blanked and the dominant coefficients were located in their proper order. Speech synthesis was performed by using the inverse orthogonal transformation. The results were stored on the disk, destroying the original data file. A final D/A conversion through a low-pass filter produced the resultant analog signal that was recorded on tape. This tape has been filed with the Electrical Engineering Department of McMaster University. A flow-chart of the above procedure is shown in

Figure 5-2. The computer programs and subroutines are given in the appendix.

5.3 FORMAT AND BIT RATE OF TEST RESULTS

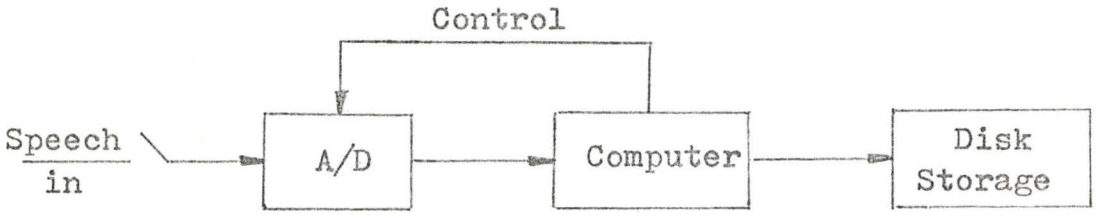
To demonstrate the degradation of the speech waveform due to the operating system, the original samples were sent directly through the D/A converter. The bit rate of data transfer was equivalent to 112,000 bits/second. Scaling the 14-bit words to 7 bits showed little change.

A second recording was made with the speech reconstructed from the dominant eight of a group of analysed 64 coefficients, updated every 8 milliseconds. To specify the coefficient number, 6 bits are required. The coefficient amplitude values are of 7 bits (including sign). Hence, eight 13-bit words describe the speech waveform every 8 milliseconds. In effect, this corresponds to an average of bit rate of reconstruction of 13,000 bits/second.

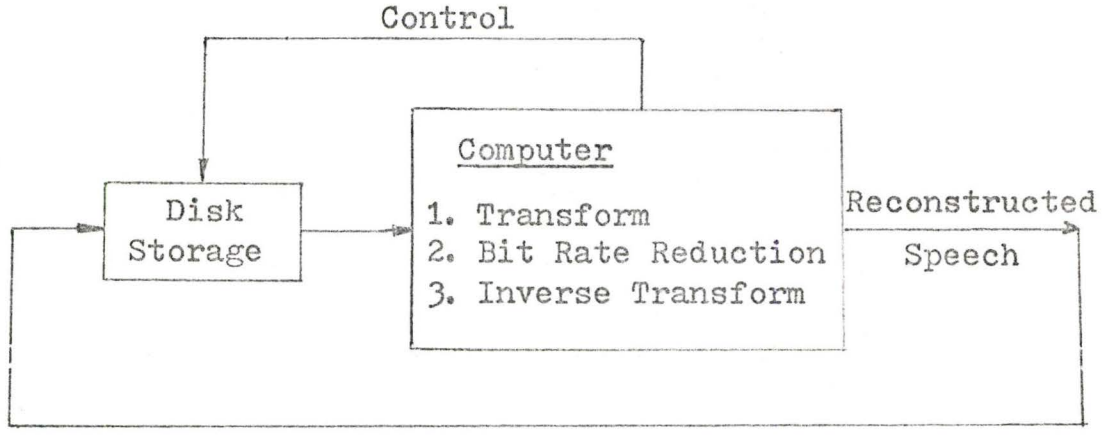
A final recording was made using only 4 dominant coefficients in the same 8-millisecond time interval. The reconstructed signal corresponded to a bit rate of 6,500 bits/second.

The memory cycle time of the CDC-1700 computer is 1.1 microseconds. The time required for the forward transform, dominant term selection, the inverse transform, and the transfer of data to the disk, is about 2 to 3 times real-time, depending on the number of dominant coefficients.

Input Stage



Bit Rate Reduction



Output Stage

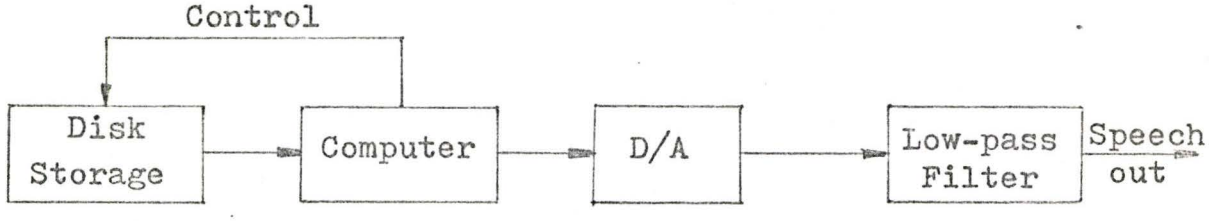


Figure 5-2: Symbolic Scheme for Bit Rate Reduction of Speech.

5.4 RESULTS AND OBSERVATIONS

The tentative results presented here are the beginning of a project to determine an effective digital scheme for minimizing the bit rate of data transfer for intelligible speech communication. For this purpose, the algorithms used in the research are made sufficiently flexible such that changes may be made on the sampling rate, the update time interval, and the number of dominant coefficients for synthesizing speech. Other combinations of dominant terms and number of Hadamard or Walsh coefficients per update interval have been tested. The tentative results show that a fair to good reproduction of speech can be obtained from 4 to 8 coefficients out of 64 in either domain. Comparison with equivalent bit rate PCM systems have not been done at this time.

In using the Hadamard or Walsh transforms, the number of phonemes subjected to severe degradation increases as the number of coefficients retained decreases. Initially, signal level thresholds were used with the results that phonemes such as /l/, /w/, and /r/ were difficult to reconstruct with 4 coefficients. Lowering the threshold level changed the time positioning of the 8 millisecond interval, as well as creating more significant coefficients. The phonemes /l/, /w/, and /r/ then were reconstructed reasonably well, but now the reconstruction of sounds such as /th/, /g/ and /p/ depended on the time location of the sampling interval relative to the speech waveform. Specific research into modifying the algorithms for varying time intervals, and

threshold levels to compensate for this inherent degradation of some phonemes has been initiated. Nevertheless, the Hadamard or Walsh transform shows promise in reducing the memory requirement for speech synthesis or word recognition.

It is also anticipated that by using a radix-4 algorithm, i.e., 4 non-zero elements in each row of a factorized matrix and consequently 4 generalized equations, the total computing time may be reduced. Furthermore, a completely digital instrument performing the inverse Hadamard or Walsh transform has been designed and constructed[32]. This device may be coupled to the CDC-1700 computer, or any general-purpose computer with a minimum 13-bit parallel transfer register available as an output. It can receive up to 8 coefficient words (formatted as described previously) in its own internal buffer registers, while it is independently reconstucting speech from the previous set of coefficients. Assuming an 8-millisecond update rate before new coefficients are received, and allowing about 100 microseconds for the computer to send 8 words to the device, this unit will then allow the computer 7.9 milliseconds to analyse speech waveforms and to select the dominant coefficients in each time period. Consequently, real-time operation may be anticipated in the near future.

5.5 PROPOSAL OF A MOBILE COMMUNICATION SYSTEM

The effectiveness of Walsh or Hadamard transform in bit rate reduction has initiated the design of a combination transmitter-receiver whose block diagram is illustrated in Figure 5-3. A key component in the circuit is the high speed parallel array hardware of Figure 4-10. This circuit can determine and transmit an arbitrary number of dominant coefficients out of a field of 64 in either the Hadamard or Walsh domains. Similarly, it can receive dominant coefficients and reconstruct the necessary waveform. The relevant data are formatted as the amplitude of the coefficient and the corresponding coefficient number. These data words are transmitted over standard data links to similar transmitter-receivers. It is estimated that a total package of this type would cost about $3,000 - $5,000. This apparatus can be used to extract the information-bearing elements of speech with an attempt to create an inexpensive digital vocabulary for voice synthesis.

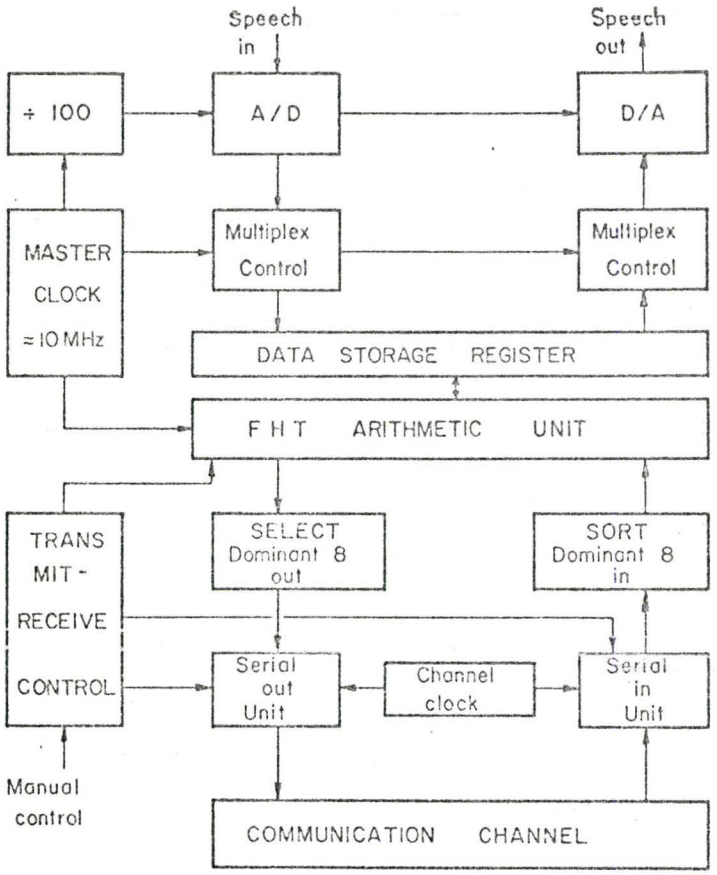


Figure 5-3: Digital Transmitter-receiver System for Speech Communications.

CHAPTER VI

CONCLUSIONS

The purpose of this project is to derive and implement efficient algorithms to compute discrete Walsh and Hadamard coefficients of a time function. Several schemes to define or generate these orthogonal functions have been examined and presented[11,15,16]. The results indicate that the two-level Walsh or Hadamard function set is readily adaptable to digital implementation; it is a natural candidate for signal multiplexing[14] and error-correcting codes[8]. Each function set has its merits or disadvantages with respect to the other.

Orthogonal transformation of a one-dimensional time function has been formulated. Extension to two-dimension is straightforward[4] but has not been attempted in the work presented here. Computational algorithms for the fast Hadamard transform (FHT) and the fast Walsh transform (FWT) have been obtained through multiplicative iteration equations or by means of the binary notation of the Walsh or Hadamard index. The FORTRAN subroutines based on these results have been streamlined to guarantee efficiency as well as minimum memory requirement. Other FHT and FWT algorithms have been derived through factorization of the transformation matrix. These factorized forms have appeared in recent literature[4, 5,18] but have not been utilized to the best advantage in

terms of simplicity[4] or memory requirement[4,5]. A parallel-operating processor for the FHT and FWT, or their inverse, have been proposed. This circuit is very simple, but very fast.

A technique to process speech in the Walsh or Hadamard domain has been modeled and implemented on a CDC-1700 computer system. Intelligible speech has been obtained with a bit rate reduction from 8:1 to 4:1. Degradation of the information content due to "dominant term" synthesis is briefly discussed. Other data compression schemes using orthogonal transformations include "threshold" filtering which retains only those coefficients above a threshold level, and "zone" filtering which disregards any terms outside a portion of the interval of orthogonality. These schemes have not been widely accepted.

The immediate objective of this project is to evaluate the effectiveness of bandwidth compression in the Hadamard or Walsh domain. The tentative results tend to indicate that the Hadamard domain should be preferred since the FHT is more compatible than the FWT in terms of computer analysis. The long range goal is to reduce the bit rate necessary for speech communication over standard data links for those applications requiring intelligibility rather than high quality, and simultaneously to reduce the hardware requirement in the implementation. A possible outcome of this approach is the production of an inexpensive audio voice

response unit for a computer that will require a minimum of memory storage. To this end, future research should be aimed at extracting information bearing elements of speech with an attempt to create a digital vocabulary for voice synthesis.

APPENDIX

APPENDIX

COMPUTER PROGRAM LISTINGS

This appendix contains the listings of computer programs written mainly for the speech processing research conducted in this project. The languages used, namely FORTRAN IV for program HTISR4 and ASSEMBLER for the others, are specifically for a CDC-1700 computer system. Comment statements have been included in the listings where necessary. For better apprehension, the appropriate reference manuals published by the Control Data Corporation should be used.

It is asserted that the number of digitized samples per update time interval is an integral power of two. To prevent integer arithmetic overflow, the Hadamard or Walsh coefficients are normalized. This is achieved by incorporating a factor of 1/2 after each addition or subtraction. This factor is not necessary for the inverse transform. The programs for the inverse transform have not been included.

HTISR4

This is a main program which utilizes most of the subsequent subroutines to reconstruct an analog signal from only a portion of its Hadamard coefficients, in a non-real-time mode. The quality of the resynthesized signal depends on the number of dominant coefficients retained per update time interval. Other variable parameters include the sampling rate, the starting address on the disk and the

number of cylinders used. (pp. 91-92)

FHTA2

This subroutine generates the Hadamard coefficients of a discrete time series, using "in place" arithmetic operations. (pp. 93-94)

PHTA2

This subroutine computes the discrete Hadamard coefficients of a digitized signal, using identical factored matrices. A buffer array is required in the computations. However, the coefficients will always be located in the data array. (pp. 95-97)

FWTA2

This subroutine computes the Walsh coefficients of a discrete time series. The coefficients will be produced in the data array. However, a buffer array is required. (pp. 98-99)

LISTEN

This subroutine uses ADBIN, DESK, NARRAY, Q8PSEN to sample an analog signal through A/D channel 0 at a given rate. The data will be stored immediately on a specific part of a disk. (pp. 100-101)

ADBIN

This subroutine samples an analog signal through A/D channel 0 at a given rate, in a buffered mode. (pp. 102-103)

NARRAY

This subroutine halts computer operations until buffered I/O reaches a specific memory location. (p. 103)

PLABAK

This subroutine uses DABOUT, DESK, NARRAY, Q8PSEN to transfer a specific portion of the disk storage through D/A channel 0 at a given rate, in a buffered mode. (pp. 104-105)

DABOUT

This subroutine outputs a signal through D/A channel 0 at a given rate, in a buffered mode. (p. 106)

EXTRAC

This subroutine determines a number of the most dominant data from an array. These data are retained in a buffer, and the corresponding location numbers are generated in a second buffer. (pp. 107-108)

RELOCA

This subroutine relocates a given number of data from an array, using indices supplied by a second array, to a third array which has been blanked initially. (p. 109)

SYNHAD

This subroutine synthesizes a discrete time series from a given number of Hadamard coefficients using indices supplied by a third array. (pp. 110-111)

SYNWAL

This subroutine synthesizes a discrete time-series from a given number of Walsh coefficients using indices supplied by a third array. (pp. 112-113)

```
      PROGRAM HTISR4
C     HADAMARD TRANSFORM IN SIGNAL RESYNTHESIS
C     USING THE DOMINANT COEFFICIENTS ONLY.
C     TO TERMINATE THE ENDLESS OPERATION OF THIS
C     PROGRAM, PRESS THE MANUAL INTERRUPT BUTTON
C     ON THE TTY & THEN KEY CR. AFTER A "MI" HAS
C     BEEN PRINTED, TYPE "*Z" & HIT CR.
      COMMON IC(4),MR(3072),NR(256),LR(520)
      WRITE (4,11)
   11 FORMAT(45HSAMPLING RATE = 200000 WORDS/SECOND. IR = )
      READ  (4,21) IR
   21 FORMAT(I2)
      WRITE (4,12)
   12 FORMAT(28HSTARTING ADDRESS ON DISK = $)
      READ  (4,22) INIT
   22 FORMAT($4)
      WRITE (4,13)
   13 FORMAT(19HNO. OF CYLINDERS = )
      READ  (4,21) LYC
      WRITE (4,14)
   14 FORMAT(40HNO. OF SAMPLES AS A TIME INTERVAL, LH = )
      READ  (4,23) LH
      WRITE (4,15)
   15 FORMAT(11HLOG2(LH) = )
      READ  (4,21) NCS
      WRITE (4,16)
   16 FORMAT(31HNO. OF DOMINANT COEFFICIENTS = )
      READ  (4,21) NDC
C     NO. OF LH WORDS ON 1/15 CYLINDER
      IL = 1024/LH
C     A "PAUSE 00001" STATEMENT APPEARS ON THE TTY TO
C     ALLOW TIME TO PREPARE FOR A/D I/P PROCESS
    6 CALL LISTEN (MR,3840,IR,INIT,LYC)
C     A/D DATA NOW STORED ON DISK
C     RECOVER INFORMATION BY THE CYLINDER
      KSID = INIT
      DO 5 L = 1,LYC
C     HANDLE EACH CYLINDER BY THE FIFTHS
      DO 4 K = 1,5
C     TRANSFER DATA FROM DISK TO CORE
      CALL DESK (MR,3072,1,KSID)
      JN = 1
C     LH MUST NOT EXCEED 1024
      DO 1 J = 1,3
C     FOR WALSH TRANSFORM, USE SUBROUTINE FWTA2
      CALL FHTA2 (MR(JN),1024,NCS)
      IN = JN
      DO 3 I = 1,IL
C     DOMINANT COEFF. ARE RETAINED IN BUFFER LR WITH
C     CORRESPONDING COEFF. NO. GENERATED AT NR
```

```
51              CALL EXTRAC (MR(IN),LR,NR,LH,NDC)
52        C     DATA ARRAY BLANKED; DOMINANT COEFF. RELOCATED
53        C     FROM LR TO MR USING INDICES FROM NR
54              CALL RELOCA (MR(IN),LR,NR,LH,NDC)
55          3   IN = IN + LH
56        C     FOR INVERSE WALSH TRANSFORM, USE IWTA2
57              CALL IHTA2 (MR(JN),1024,NCS)
58          1   JN = JN + 1024
59        C     RECONSTRUCTED SIGNAL TRANSFERRED TO DISK
60              CALL DESK (MR,3072,0,KSID)
61        C     DISK ADDRESS UPDATED
62          4   KSID = KSID + 32
63        C     A "PAUSE 00002" APPEARS ON THE TTY TO ALLOW
64        C     TIME TO PREPARE FOR D/A O/P PROCESS
65              CALL PLABAK (MR,3840,IR,INIT,LYC)
66              GO TO 6
67         23   FORMAT (I4)
68              STOP
69              END
```

```
          NAM    FHTA2
          ENT    FHTA2
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL FHTA2 (MR,LH,NC)
*   TRANSFORMS THE LH DATA SAMPLES IN ARRAY MR
*   TO THE HADAMARD DOMAIN WITH UPDATE INTERVAL
*   2**NC, USING "IN PLACE" ARITHMETIC
*   EG    CALL FHTA2 (MR,1024,6)
FHTA2     0      0
          LDQ*   FHTA2      PICK UP PARAMETERS
          LDA-   (2),Q      DATA ARRAY

          STA*   MR
          LDA-   1,Q        NO. OF SAMPLES
          STA*   LH
          LDA-   2,Q        NO. OF FACTORED MATRICES
          STA*   NC
          LDA*   (NC)
          STA*   NC
          INQ    3
          STQ*   FHTA2      RETURN ADDRESS
          ENQ    1
          STQ*   NB
          LDA*   (LH)
L1        ARS    1
          STA*   LH
          LDQ*   MR
L2        STA*   N2
          LDA*   NB
L3        STA*   N3
          LDA-   (2),Q
          ADD*   (NB),Q
          ARS    1
          STA-   (2),Q
          SUB*   (NB),Q
          STA*   (NB),Q
          INQ    1
          LDA*   N3
          INA    -1
          SAZ    1
          JMP*   L3
          LDA*   N2
          INA    -1
          SAZ    2
          ADQ*   NB
          JMP*   L2
          LDA*   NC
          INA    -1
          SAZ    6
          STA*   NC
```

```
        LDA*    NB
        ALS     1
        STA*    NB
        LDA*    LH
        JMP*    L1
        JMP*    (FHTA2)
        BSS     MR,LH,NC,NB,N2,N3
        END
MON
```

```
          NAM    PFHA2
          ENT    PFHA2
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL PFHA2 (MR,NR,LH,NC)
*   TRANSFORM THE LH DATA SAMPLES TO THE HADAMARD
*   DOMAIN USING IDENTICAL FACTORED MATRICES,
*   WITH ARRAY NR AS A BUFFER. LH = 2**NC.
*   TO ENSURE THAT THE COEFFS. WILL APPEAR IN THE
*   DATA ARRAY MR, A SINGLE STAGE RADIX-4 TRANSFORM,
*   IE. 4 NON-ZERO ENTRIES PER ROW IN A FACTORED
*   MATRIX, IS APPLIED IF NC IS ODD.
*   EG    CALL PFHA2 (MR,NR,64,6)
PFHA2     0      0
          LDA-   I          SAVE REGISTER I
          STA*   SI
          LDQ*   PFHA2      PICK UP PARAMETERS
          LDA-   (2),Q      DATA ARRAY
          INA    -1
          STA*   MR
          LDA-   1,Q        BUFFER ARRAY
          INA    -1
          STA*   NR
          LDA-   2,Q        NO. OF DATA
          STA*   LH
          LDA-   3,Q
          STA*   NC         LH = 2**NC
          INQ    4
          STQ*   PFHA2      RETURN ADDRESS
          LDQ*   (LH)
          QRS    1
          STQ*   LH
          LDA*   (NC)       CHECK IF NC ODD OR EVEN
          TRA    Q
          AND-   3
          SAN    1
          JMP*   OK         NC EVEN; PROCEED
          INQ    -2         NC ODD; APPLY ONE STAGE
          STQ*   NC         RADIX-4 HADAMARD TRANSFORM
          LDA*   LH
          TRA    Q
          ARS    1
          STA*   QR
          AAQ    Q
          STQ*   Q3
          LDQ*   NR
          STQ-   I
          LDQ*   MR
L1        STA*   X1
          LDA-   1,Q        RADIX-4 TRANSFORM
          ADD-   2,Q
          ADD-   3,Q
          ADD-   4,Q
```

```
        ARS     2
        STA-    1,I
        RAO-    I
        LDA-    1,Q
        SUB-    2,Q
        ADD-    3,Q
        SUB-    4,Q
        ARS     2
        STA*    (QR),I
        LDA-    1,Q
        ADD-    2,Q
        SUB-    3,Q
        SUB-    4,Q
        ARS     2
        STA*    (LH),I
        LDA-    1,Q
        SUB-    2,Q
        SUB-    3,Q
        ADD-    4,Q
        ARS     2
        STA*    (Q3),I
        LDA*    X1
        INA     -1
        SAZ     3
        INQ     4
        JMP*    L1
OK      STQ*    NC
        LDA*    NC
        INA     -1
        SAN     3
        LDA*    SI
        STA-    I               RESTORE REGISTER I
        JMP*    (PFHA2)
        STA*    NC
        AND-    3
        SAZ     3
        LDQ*    MR
        LDA*    NR
        SAN     2
        LDQ*    NR
        LDA*    MR
        STA-    I
        LDA*    LH
L2      INA     -1
        STA*    QR
        LDA-    1,Q             RADIX-2 TRANSFORM
        ADD-    2,Q
        ARS     1
        STA-    1,I
        RAO-    I
        SUB-    2,Q
        STA*    (LH),I
```

```
        LDA*    QR
        SAZ     1
        JMP*    L2
        JMP*    OK+1
        BSS     SI,MR,NR,LH,QR,Q3,X1
        END
MON
```

```
            NAM     FWTA2
            ENT     FWTA2
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*           CALL FWTA2 (MR,NR,LH,BT)
*   TRANSFORM THE LH DATA SAMPLES IN ARRAY MR TO THE
*   WALSH DOMAIN, WITH UPDATE INTERVAL 2**BT,
*   USING ARRAY NR AS A BUFFER
*   EG      CALL FWTA2 (MR,NR,1024,6)
FWTA2       0       0
            LDQ*    FWTA2       PICK UP PARAMETERS
            LDA-    (2),Q
            STA*    MP          DATA ARRAY
            INA     -1
            STA*    MR
            LDA-    1,Q         BUFFER ARRAY
            INA     -1
            STA*    NR
            LDA-    2,Q         LENGTH OF ARRAY
            STA*    LH
            LDA-    3,Q         NO. OF FACTORED MATRICES
            STA*    BT
            INQ     4
            STQ*    FWTA2       RETURN ADDRESS
            LDQ*    (LH)
            QRS     1
            STQ*    LH
            QLS     1
            INQ     -1
            LDA*    (BT)        IF BT IS ODD, USE "IN PLACE"
            AND-    3           ARITHMETIC FOR COMPUTATION
            SAZ     2
            LDA*    MR
            SAN     1
            LDA*    NR
            STA*    K
            INA     1
            STA*    KH
L1          LDA*    (MR),Q
            ADD*    (MP),Q
            ARS     1
            STA*    (K),Q
            SUB*    (MP),Q
            STA*    (KH),Q
            INQ     -2
            SQM     1
            JMP*    L1
            ENQ     1
            LDA*    (BT)
            INA     -1
L2          STA*    BT
            STQ*    KH
```

```
        QLS     1
        STQ*    K
        INQ     1
        STQ*    MP
        AND-    3
        SAZ     3
        LDQ*    NR
        LDA*    MR
        SAN     2
        LDQ*    MR
        LDA*    NR
        STA-    I
        LDA*    LH
        ARS     1
        STA*    LH
L3      STA*    X3
        LDA*    KH
L4      STA*    X4
        LDA-    1,Q
        ADD*    (K),Q
        ARS     1
        STA-    1,I
        SUB*    (K),Q
        STA-    2,I
        LDA-    2,Q
        SUB*    (MP),Q
        ARS     1
        STA-    3,I
        ADD*    (MP),Q
        STA-    4,I
        LDA-    I
        INA     4
        STA-    I
        INQ     2
        LDA*    X4
        INA     -1
        SAZ     1
        JMP*    L4
        ADQ*    K
        LDA*    X3
        INA     -1
        SAZ     1
        JMP*    L3
        LDA*    BT
        INA     -1
        SAZ     2
        LDQ*    K
        JMP*    L2
        JMP*    (FWTA2)
        BSS     MR,MP,NR,BT,LH,K,KH,X3,X4
        END
MON
```

```
          NAM    LISTEN
          ENT    LISTEN
          EXT    ADBIN,DESK,NARRAY,Q8PSEN
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL LISTEN (MR,LG,IR,KS,LY)
*   SAMPLES A SIGNAL THROUGH ADC CHANNEL 0
*   AT (200000/IR) WORDS/SECOND, USING ARRAY MR
*   OF LENGTH LG AS A BUFFER FOR TRANSFERRING
*   THE A/D SAMPLES TO LY CYLINDERS ON THE DISK,
*   STARTING AT ADDRESS KS. LG MUST BE AN EVEN
*   FACTOR OF 15360, THE NO. OF WORDS/CYLINDER.
*   EG    CALL LISTEN (MR,3840,25,$4B00,12)
*
LISTEN    0      0
          LDQ*   LISTEN     PICK UP PARAMETERS
          LDA-   (2),Q
          STA*   MR         BUFFER ADDRESS
          STA*   NR
          LDA-   1,Q
          STA*   LG
          LDA*   (LG)       BUFFER LENGTH
          ARS    1
          STA*   LH         HANDLE BUFFER AS 2 HALVES
          ADD*   MR
          STA*   JR         TOP OF 2ND HALF
          INA    -1
          STA*   PR         END OF 1ST HALF
          ADD*   LH
          STA*   QR         END OF 2ND HALF
          LDA-   2,Q
          STA*   IR         SAMPLING RATE
          LDA-   3,Q
          STA*   KS
          LDA*   (KS)
          STA*   KS         STARTING ADDRESS ON DISK
          LDA-   4,Q
          STA*   LY         NO. OF CYLINDERS USED
          INQ    5
          STQ*   LISTEN     RETURN ADDRESS
          ENQ    0
          LDA    =N15360    WORDS/CYLINDER
          DVI*   (LG)
          STA*   KY
          ENQ    0
          ENA    80
          DVI*   KY
          STA*   IN         INCREMENT OF DISK ADDRESS
          RTJ+   Q8PSEN     "PAUSE 00001" TO PREPARE
          NUM    1          FOR INPUT PROCESS
          RTJ+   ADBIN      START BUFFERED INPUT
          BSS    MR,LG,IR
          LDA*   (LY)
```

```
L1      STA*    LY
        LDA*    KY          HANDLE EACH CYLINDER AS KY ARRAYS
L2      STA*    N2
        RTJ+    NARRAY      WAIT UNTIL I/P DATA FILLS
        BSS     PR          1ST HALF OF BUFFER
        RTJ+    DESK        TRANSFER DATA TO DISK
        BSS     NR
        ADC     LH
        NUM     2
        ADC     KS
        LDA*    KS
        ADD*    IN
        STA*    KS          UPDATE DISK ADDRESS
        RTJ+    NARRAY      REPEAT THE SAME PROCEDURE
        BSS     QR          FOR THE 2ND HALF
        RTJ+    DESK
        BSS     JR
        ADC     LH
        NUM     2
        ADC     KS
        LDA*    KS
        ADD*    IN
        STA*    KS
        LDA*    N2
        INA     -1
        SAZ     1
        JMP*    L2
        LDA*    KS
        INA     96          UPDATE DISK ADDRESS BY THE CYLINDER
        STA*    KS
        LDA*    LY
        INA     -1
        SAZ     1
        JMP*    L1
        LDQ*    QR
        LDA-    $21
        STA-    1,Q         END BUFFERED I/O
        JMP*    (LISTEN)
        BSS     IN,KS,KY,LH,LY,N2
        END
MON
```

```
          NAM     ADBIN
          ENT     ADBIN
*    THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL ADBIN (MR,LH,IR)
*    SAMPLES A SIGNAL THROUGH ADC CHANNEL 0 AT RATE
*    OF (200000/IR) WORDS/SECOND, WITH ARRAY MR OF
*    LENGTH LH AS BUFFER.
*    EG    CALL ADBIN (MR,3840,25)
*    TO END BUFFERED  I/O, SET MR(LH+1) = $8000.
ADBIN     0       0
          LDQ*    ADBIN       PICK UP PARAMETERS
          LDA-    1,Q
          STA*    LH
          LDA-    2,Q
          STA*    FY
          LDA-    (2),Q
          INQ     3
          STQ*    ADBIN       RETURN ADDRESS
          TRA     Q
          LDA*    (LH)
          AAQ     A
          STA*    LH
          INQ     -4          HEADER LOCATION
          STQ*    HR
          INA     -1          END OF BUFFER
          STA-    3,Q
          LDA     =N$E400
          STA-    (2),Q
          LDA     =N$9401
          STA-    1,Q
          LDA     =N$8155
          STA-    2,Q
          INQ     2
          STQ*    (LH)        ADDRESS FOR REPEAT OPERATION
          LDQ-    $2D         CLEAR DCT 1750
          ENA     1
          OUT     -1
          INQ     $4D         FUNCTION INTERFACE 1538
          LDA     =N$094D
          NOP
          OUT     -1
          LDQ     =N$0403     FUNCTION SAMPLE RATE UNIT 1572
          LDA     =N$6900
          NOP
          OUT     -1
          INQ     -1
          LDA*    (FY)
          OUT     -1
          LDQ     =N$0460     BUFFERED I/O INTERFACE
          LDA*    HR
          OUT     -1          I/O BEGINS
          JMP*    (ADBIN)
          BSS     HR,LH,FY
          END
MON
```

```
            NAM     NARRAY
            ENT     NARRAY
*
*    THIS SUBROUTINE, WHEN ACTIVATED BY
*          CALL NARRAY(K)
*    HALTS COMPUTER EXECUTION UNTIL
*     BUFFERED I/O REACHES LOCATION K
*
NARRAY      0       0
            LDQ     =N$0468    CHAINING BUFFER 1571
            NOP
L           INP     -1         NEXT MEMORY ADDRESS
            EOR*    (NARRAY)
            SAZ     1
            JMP*    L
            RAO*    NARRAY
            JMP*    (NARRAY)
            END
 MON
```

```
          NAM    PLABAK
          ENT    PLABAK
          EXT    DABOUT,DESK,NARRAY,Q8PSEN
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL PLABAK (MR,LG,IR,KS,LY)
*   OUTPUTS A SIGNAL THROUGH DAC CHANNEL 0 AT THE RATE
*   OF (200000/IR) WORDS/SECOND, USING ARRAY MR OF
*   LENGTH LG AS A BUFFER FOR TRANSFERRING THE DATA ON
*   LY CYLINDERS OF THE DISK, STARTING AT ADDRESS KS
*   LG MUST BE AN EVEN FACTOR OF 15360.
*   EG    CALL PLABAK (MR,3840,25,$4B00,12)
PLABAK    0      0
          LDQ*   PLABAK     PICK UP PARAMETERS
          LDA-   (2),Q
          STA*   MR         BUFFER ADDRESS
          STA*   NR
          LDA-   1,Q
          STA*   LG
          LDA*   (LG)       BUFFER LENGTH
          ARS    1
          STA*   LH         HANDLE BUFFER AS 2 HALVES
          ADD*   MR
          STA*   JR         TOP OF 2ND  HALF
          INA    -1
          STA*   PR         END OF 1ST HALF
          ADD*   LH
          STA*   QR         END OF 2ND HALF
          LDA-   2,Q
          STA*   IR         SAMPLING RATE
          LDA-   3,Q
          STA*   KS
          LDA*   (KS)
          STA*   KS         STARTING ADDRESS ON DISK
          LDA-   4,Q
          STA*   LY         NO. OF CYLINDERS USED
          INQ    5
          STQ*   PLABAK     RETURN ADDRESS
          ENQ    0
          STQ*   FL         FLAG FOR O/P PROCESS
          LDA    =N15360    WORDS/CYLINDER
          DVI*   (LG)
          STA*   KY
          ENQ    0
          ENA    80
          DVI*   KY
          STA*   IN         INCREMENT OF DISK ADDRESS
          RTJ+   Q8PSEN     "PAUSE 00002" TO PREPARE
          NUM    2          FOR OUTPUT PROCESS
          LDA*   (LY)
L1        STA*   LY
          LDA*   KY         HANDLE EACH CYLINDER AS KY ARRAYS
```

```
L2      STA*    N2
        RTJ+    DESK        TRANSFER DATA FROM DISK TO CORE
        BSS     NR
        ADC     LH
        NUM     3
        ADC     KS
        LDA*    FL          TEST FLAG FOR O/P PROCESS
        SAN     L3-*-1
        ENA     1
        STA*    FL          RESET FLAG
        RTJ+    DABOUT      START BUFFERED O/P
        BSS     MR,LG,IR
        JMP*    L4
L3      RTJ+    NARRAY      WAIT UNTIL O/P REACHES
        BSS     QR          END OF 2ND HALF BUFFER
L4      LDA*    KS
        ADD*    IN
        STA*    KS          UPDATE DISK ADDRESS
        RTJ+    DESK        REPEAT PROCEDURE FOR 2ND BUFFER
        BSS     JR
        ADC     LH
        NUM     3
        ADC     KS
        RTJ+    NARRAY
        BSS     PR
        LDA*    KS
        ADD*    IN
        STA*    KS
        LDA*    N2
        INA     -1
        SAZ     1
        JMP*    L2
        LDA*    KS
        INA     96          UPDATE DISK ADD. BY THE CYL.
        STA*    KS
        LDA*    LY
        INA     -1
        SAZ     1
        JMP*    L1
        LDQ*    QR
        LDA-    $21
        STA-    1,Q         END BUFFERED I/O
        JMP*    (PLABAK)
        BSS     FL,IN,KS,KY,LH,LY,N2
        END
MON
```

```
          NAM    DABOUT
          ENT    DABOUT
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL DABOUT (MR,LH,IR)
*   OUTPUTS A SIGNAL THROUGH DAC CHANNEL () AT RATE
*   OF (200000/IR) SAMPLES/SEC., WITH ARRAY MR OF
*   LENGTH LH AS BUFFER.
*   EG    CALL DABOUT (MR,3840,25)
*   TO END BUFFERED I/O, SET MR(LH+1) = $8000.
DABOUT    0      0
          LDQ*   DABOUT     PICK UP PARAMETERS
          LDA-   1,Q
          STA*   LH
          LDA-   2,Q
          STA*   FY
          LDA-   (2),Q
          INQ    3
          STQ*   DABOUT     RETURN ADDRESS
          TRA    Q
          LDA*   (LH)
          AAQ    A
          STA*   LH
          INQ    -4         HEADER LOCATION
          STQ*   HR
          INA    -1         END OF BUFFER
          STA-   3,Q
          LDA    =N$E400
          STA-   (2),Q
          LDA    =N$9401
          STA-   1,Q
          LDA    =N$8255
          STA-   2,Q
          INQ    2
          STQ*   (LH)       ADDRESS FOR REPEAT OPERATION
          LDQ-   $2D        CLEAR DCT 1750
          ENA    1
          OUT    -1
          INQ    $52        FUNCTION INTERFACE 1561
          LDA    =N$024D
          NOP
          OUT    -1
          LDQ    =N$0403    FN. SAMPLE RATE UNIT 1572
          LDA    =N$6900
          NOP
          OUT    -1
          INQ    -1
          LDA*   (FY)
          OUT    -1
          LDQ    =N$0460    BUFFERED I/O INTERFACE 1797
          LDA*   HR
          OUT    -1         I/O BEGINS
          JMP*   (DABOUT)
          BSS    HR,LH,FY
          END
MON
```

```
          NAM    EXTRAC
          ENT    EXTRAC
*
*   THIS SUBROUTINE,WHEN ACTIVATED BY
*         CALL EXTRAC (MR,LR,NR,LH,NF)
*   EXTRACTS THE MOST DOMINANT NF DATA FROM LH DATA
*   IN ARRAY MR & PLACES THEM IN DESCENDING ORDER
*   IN TERMS OF ABSOLUTE VALUES IN ARRAY LR.
*   NUMBERS FROM 0 TO NF-1 CORRESPONDING TO THE
*   LOCATIONS OF THE DOMINANT DATA IN ARRAY MR
*   ARE GENERATED & STORED IN ARRAY NR.
*   LOCATIONS FOR THE DOMINANT DATA ARE BLANKED.
*   EG    CALL EXTRAC (MR,LR,NR,1024,128)
*
EXTRAC    0    0
          LDQ* EXTRAC  PICK UP PARAMETERS
          LDA- (2),Q
          STA* MR      ADDRESS OF DATA ARRAY
          LDA- 1,Q
          STA* LR      DOMINANT DATA ARRAY
          LDA- 2,Q
          STA* NR      GENERATED LOC. # ARRAY
          LDA- 3,Q
          STA* LH      NO. OF DATA
          LDA- 4,Q
          STA* NF
          LDA* (NF)
          STA* NF      NO. OF DOMINANT DATA
          INQ  5
          STQ* EXTRAC  RETURN ADDRESS
          LDQ* (LH)
L1        INQ  -1      LOCATE /DATUM/ > 0
          SQM  L2-*-1
          LDA* (MR),Q  OBTAIN A DATUM
          SAP  1
          TCA  A       ABSOLUTE VALUE
          SAN  L3-*-1  /DATUM/ > 0
          JMP* L1
L2        LDA* NF      DATA ALL ZERO
          ENQ  0
          STQ* (LR)    BLANK REMAINING LR
          INA  -1
          SAZ  2
          RAO* LR
          JMP* L2+2
          JMP* (EXTRAC)  RETURN TO MAIN PROGRAM
L3        STQ* NZ       INDEX FOR /DATUM/ > 0
```

```
            STQ*  GN       HOLD GENERATED LOC. #
            STA*  AV       HOLD /DATUM/
L4          INQ   -1
            SQM   L5-*-1
            LDA*  (MR),Q   OBTAIN NEXT DATUM
            SAP   1
            TCA   A
            SAZ   5        DATUM 0, SKIP TEST
            SUB*  AV       COMPARE WITH AV
            SAM   3
            SAZ   2
            ADD*  AV       /DATUM/ > AV
            JMP*  L3+1     UPDATE AV & GN
            JMP*  L4
L5          LDQ*  GN       LARGEST DATUM LOCATION
            STQ*  (NR)
            LDA*  (MR),Q   LARGEST DATUM
            STA*  (LR)
            ENA   0
            STA*  (MR),Q   BLANK CORR. LOC. IN MR
            LDA*  NF
            INA   -1
            SAZ   EX-*-1
            STA*  NF
            RAO*  LR       UPDATE LR & NR
            RAO*  NR
            LDQ*  NZ
            JMP*  L1+2
EX          JMP*  (EXTRAC)  RETURN TO MAIN PROGRAM
            BSS   LR,MR,NR,LH,NF,AV,NZ,GN
            END
 MON
```

```
          NAM    RELOCA
          ENT    RELOCA
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL RELOCA (FA,IA,IN,LF,LI)
*   BLANKS ARRAY FA, AND THEN RELOCATE
*   THE LI DATA FROM ARRAY IA TO FA,
*   USING THE INDICES FROM ARRAY IN.
RELOCA    0      0
          LDQ*   RELOCA     PICK UP PARAMETERS
          LDA-   (2),Q
          STA*   FA
          LDA-   1,Q
          STA*   IA
          LDA-   2,Q
          STA*   IN
          LDA-   3,Q        LENGTH OF FA
          STA*   LF
          LDA-   4,Q
          STA*   LI
          INQ    5
          STQ*   RELOCA     RETURN ADDRESS
          ENA    0
          LDQ*   (LF)
X1        INQ    -1         BLANK ARRAY FA
          STA*   (FA),Q
          SQZ    1
          JMP*   X1
          LDA*   (LI)
X2        INA    -1
          STA*   LI
          LDA*   (IA)       OBTAIN DATUM
          LDQ*   (IN)       GET CORR. INDEX
          STA*   (FA),Q     RELOCATE DATUM
          LDA*   LI
          SAZ    3
          RAO*   IA
          RAO*   IN
          JMP*   X2
          JMP*   (RELOCA)
          BSS    FA,IA,IN,LF,LI
          END
 MON
```

```
          NAM   SYNHAD
          ENT   SYNHAD
*
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL SYNHAD (LR,MR,NR,LH,NF)
*   USES INVERSE HADAMARD TRANSFORM TO SYNTHESIZE
*   A SIGNAL OF LH WORDS AT ARRAY LR FROM NF CFTS.
*   AT ARRAY MR & THEIR CORRESPONDING CFT. NOS.
*   AT ARRAY NR.
*   EG    CALL SYNHAD (LR,MR,NR,1024,128)
*
SYNHAD    0    0
          LDQ* SYNHAD  PICK UP PARAMETERS
          LDA- (2),Q
          STA* LR      FINAL RESULT ARRAY ADDRESS
          LDA- 1,Q
          STA* MR      CFT. MAGNITUDE ARRAY
          LDA- 2,Q
          STA* NR      CFT. NOS. ARRAY ADDRESS
          LDA- 3,Q
          STA* LH      LENGTH OF ARRAY LR
          LDA- 4,Q
          STA* FN      NO. OF CFTS USED
          INQ  5
          STQ* SYNHAD  RETURN ADDRESS
          LDQ* (LH)
          STQ* LH
          ENA  0
L1        INQ  -1      CLEAR ARRAY LR
          STA* (LR),Q
          SQZ  1
          JMP* L1
          LDQ* (FN)
L2        INQ  -1
          STQ* FN
          LDA* (MR),Q  OBTAIN A CFT.
          SAN  1
          JMP* L4      SKIP IF CFT. 0
          STA* HO
          LDA* (NR),Q  GET CORR. CFT. NO.
          STA* PN
          LDQ* LH
L3        INQ  -1
          LDA* PN
          LAQ  A
          SPA* DP      PARITY OF (CFT.#).(POSITION #)
          SAN  2
          SUB* HO      PARITY ODD, MINUS CFT.
          JMP* *+2
          LDA* HO      PARITY EVEN, ADD CFT.
```

```
            ADD*  (LR),Q
            STA*  (LR),Q
            SQZ   1
            JMP*  L3
            LDQ*  FN
L4          SQZ   1
            JMP*  L2
            JMP*  (SYNHAD)
            BSS   LR,MR,NR,LH,FN,HO,PN,DP
            END
 MON
```

```
          NAM   SYNWAL
          ENT   SYNWAL
*
*   THIS SUBROUTINE, WHEN ACTIVATED BY
*         CALL SYNWAL (LR,MR,NR,LH,NF)
*   USES INVERSE WALSH TRANSFORM TO SYNTHESIZE
*   A SIGNAL OF LH WORDS AT ARRAY LR FROM NF CFTS.
*   AT ARRAY MR & THEIR CORRESPONDING CFT. NOS.
*   AT ARRAY NR.
*   EG    CALL SYNWAL (LR,MR,NR,1024,128)
SYNWAL    0     0
          LDQ*  SYNWAL  PICK UP PARAMETERS
          LDA-  (2),Q
          STA*  LR      FINAL RESULT ARRAY ADDRESS
          LDA-  1,Q
          STA*  MR      CFT. MAGNITUDE ARRAY
          LDA-  2,Q
          STA*  NR      CFT. NOS.ARRAY ADDRESS
          LDA-  3,Q
          STA*  LH      LENGTH OF ARRAY LR
          LDA-  4,Q
          STA*  FN      NO. OF CFTS. USED
          INQ   5
          STQ*  SYNWAL  RETURN ADDRESS
          LDQ*  (LH)
          STQ*  LH
          ENA   0
L1        INQ   -1      CLEAR ARRAY LR
          STA*  (LR),Q
          SQZ   1
          JMP*  L1
          LDQ*  LH
          INQ   -1
L0        QRS   1       DETERMINE NO. OF BITS
          INA   1       REQD. TO INDEX LH DATA
          SQZ   1
          JMP*  L0
          STA*  BT
          LDQ*  (FN)
L2        INQ   -1
          STQ*  FN
          LDA*  (MR),Q  OBTAIN A CFT.
          SAN   1
          JMP*  L4      SKIP IF CFT. = 0
          STA*  H0
          LDA*  (NR),Q  GET CORR. CFT. NO.
          TRA   Q
          ARS   1
          EAQ   A       GRAY CODE OF CFT. NO.
          LDQ*  BT
```

```
            STQ* DP
            ENQ  0
L5          STA* PN       REVERSE THE BITS
            AND- 3
            AAQ  Q
            LDA* DP
            INA  -1
            SAZ  5
            STA* DP
            QLS  1
            LDA* PN
            ARS  1
            JMP* L5
            STQ* PN
            LDQ* LH
L3          INQ  -1
            LDA* PN
            LAQ  A
            SPA* DP       PARITY OF (CFT.#).(POSITION #)
            SAN  2
            SUB* HO       PARITY ODD,MINUS CFT.
            JMP* *+2
            LDA* HO       PARITY EVEN, ADD CFT.
            ADD* (LR),Q
            STA* (LR),Q
            SQZ  1
            JMP* L3
            LDQ* FN
L4          SQZ  1
            JMP* L2
            JMP* (SYNWAL)  RETURN TO MAIN PROGRAM
            BSS  LR,MR,NR,LH,FN,HO,PN,DP,BT
            END
 MON
```

REFERENCES

(1) J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series", Math. of Computation, vol. 19, pp. 297-301, April 1965.

(2) J. Hadamard, "Resolution d'une question relative aux determinants", Bull. Sci. Math., ser. 2, vol. 17, pt. 1, pp. 240-246, 1893.

(3) J. L. Walsh, "A closed set of orthogonal functions", Amer. J. of Math., vol. 45, pp. 5-24, 1923.

(4) W. K. Pratt, J. Kane and H. C. Andrews, "Hadamard transform image coding", Proc. IEEE, vol. 57, no. 1, pp. 58-68, Jan. 1969.

(5) L. J. Ulman, "Computation of the Hadamard transform and the R-transform in ordered form", IEEE Trans. Computers, vol. C-19, no. 4, pp. 359-360, April 1970.

(6) S. J. Campanella and G. S. Robinson, "Digital sequency decomposition of voice signals", Proc. 1970 Symp. Walsh Functions, Washington D. C., pp. 230-237.

(7) C. Boesswetter, "Analog sequency analysis and synthesis of voice signals", Proc. 1970 Symp. Walsh Functions, Washington D. C., pp. 220-229.

(8) H. F. Harmuth, Transmission of information by orthogonal functions, Springer-Verlag, Berlin/New York, 1969.

(9) G-AEC Subcommittee on Measurement Concepts, "What is the Fast Fourier transform?", Proc. IEEE, vol. 55, pp. 1664-1674, Oct. 1967.

(10) H. Rademacher, "Einige Sätze über Reihen von allgemeinen Orthogonalfunktinon", Math. Annalen, vol. 87, pp. 122-138, 1922.

(11) D. A. Swick, "Walsh function generation", IEEE Trans. Info. Theory, vol. IT-15, no. 1, p. 167, Jan. 1969.

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

(13) L. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme", IRE Trans. Info. Theory, IT-4, pp. 48-49, 1954.

(14) I. A. Davidson, "The use of Walsh functions for multiplexing signals", Proc. 1970 Symp. Walsh Functions, Washington D. C., pp. 23-25.

(15) R. Kitai and K. H. Siemens, "A hazard-free Walsh function generator", IEEE Trans. Instrumentation and Measurements, pp. 80-83, Feb. 1972.

(16) A. R. Elliott and S. Mikhail, "A programmable Walsh function generator and its use in a high speed inverse transform apparatus", Proc. 1972 Symp. Walsh Functions, Washington D. C., pp. 173-176.

(17) A. R. Elliott, "A high speed binary rate multiplier", Proc. IEEE, vol. 59, no. 8, pp. 1256-1258, Aug. 1971.

(18) J. E. Whelchel Jr. and D. F. Guinn, "The Fast Fourier-Hadamard and its use in signal representation and classification", EASCON Convention Record, 1968, pp. 561-571.

(19) W. K. Pratt, "A bibliography on television bandwidth reduction studies", IEEE Trans. on Info. Theory, vol. IT-13, no. 1, pp. 114-115, Jan. 1967.

(20) A. Rosenfeld, "Bandwidth reduction bibliography", IEEE Trans. on Info. Theory, vol. IT-14, no. 4, pp. 601-602, July 1968.

(21) Special Issue on Redundancy Reduction, Proc. IEEE, vol. 55, no. 3, March 1967.

(22) K. Hoffman, Boush Spaces of Analytic Functions, Prentice Hall 1962, pp. 10-13.

(23) S. J. Campanella and G. S. Robinson, "A comparison of Walsh and Fourier transformations to speech", Proc. 1971 Symp. Walsh Functions, Washington D. C., pp. 199-205.

(24) S. J. Campanella and G. S. Robinson, "A comparison of orthogonal transformations for digital speech processing", IEEE Trans. Comm. Technology, vol. COM-19, no. 6, pp. 1045-1050, Dec. 1971.

(25) S. J. Campanella and G. S. Robinson, "Orthogonal transform feasibility study", Technical Report No. CL-TR-5-71, COMSAT Laboratories, Clarksburg, Maryland, Nov. 1971.

(26) W. K. Pratt, "Walsh functions in image processing and two-dimensional filtering", Proc. 1972 Symp. Walsh Functions, Washington D. C., pp. 14-22.

(27) S. Watanabe, "Karhunen-Loève expansion and factor analysis", Trans. Fourth Praque Conference on Info. Theory, 1965, pp. 635-660.

(28) A. Habibi and P. Wintz, "Optimum linear transformations for encoding 2-dimensional data", Technical Report No. TR-EE-69-15, Purdue University, May 1969.

(29) H. C. Andrews and K. L. Caspari, "A generalized technique for spectral analysis", IEEE Trans. Computers, vol. C-19, no. 1, pp. 16-25, Jan. 1970

(30) H. Enomoto and K. Shibata, "Orthogonal transform coding system for television signals", J. of the Inst. of TV Engineers of Japan, vol. 24, no. 2, pp. 99-108, Feb. 1970.

(31) W. K. Pratt, L. R. Welch and W. Chen, "Slant transform for image coding", Proc. 1972 Symp. Walsh Functions, Washington D. C., pp. 229-234.

(32) W. C. Brown, A digital waveform synthesizer using Walsh functions, McMaster University, Hamilton, Ontario, M. Eng. Thesis, 1971.

(33) W. W. Peterson, Error correcting codes, Wiley 1961, pp. 73-81.