DIGITAL INSTRUMENTATION FOR WALSH-FOURIER SPECTRUM MEASUREMENT

by

© K. Muniappan, B.E., M.E., M.Eng.

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

August 1979

DIGITAL INSTRUMENTATION FOR WALSH-FOURIER SPECTRUM MEASUREMENT

DOCTOR OF PHILOSOPHY (1979)
(Electrical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: Digital Instrumentation for Walsh-Fourier Spectrum Measurement

AUTHOR: K. Muniappan, B.E. (Madras, 1966)
M.E. (I.I.Sc, 1969)
M.Eng. (McMaster, 1976)

SUPERVISOR: Professor R. Kitai

NUMBER OF PAGES: xvi, 187

ABSTRACT

A review of Walsh Spectral Analysers (WSA) using direct and fast transform methods is presented. A serial processor is developed that uses long shift registers to perform a Hadamard transform. This is extended by adding additional hardware to yield coefficients in dyadic and sequency order. Incoming data is thereby stored in a permuted manner, followed by sequential retrieval and transfer to a Hadamard transform processor. This scheme is faster than an earlier processor described by Geadah and Corinthios. Also a new pipeline structure with identical stages is evolved.

The design of a Microprocessor-based Walsh-Fourier Spectral Analyser is given in detail. It uses an off-the-shelf single board microcomputer System 80/10 in conjunction with a special purpose board. The latter includes A/D conversion circuitry, a direct memory access controller, and a frequency multiplication module (FMM). The purpose of FMM is to generate 2^m sampling pulses (64 in the system built) within one cycle of the input signal; this feature is required to compute the Fourier/Walsh coefficients of a periodic signal without leakage error. The new FMM circuit is a substantial improvement over earlier designs and permits a higher frequency of operation at a low clock

frequency.

Two methods (one by Siemens and Kitai and the other by Tadokoro and Higuchi) of Walsh to Fourier conversion are reviewed. The conversion process is compared with the Cooley-Tukey FFT methods with respect to the number of multiplications, memory requirements, and effects of finite word length in computation.

The Walsh to Fourier conversion process is implemented through software to obtain 64 sine and cosine components. The instrument is interfaced to a HP 2647A intelligent graphic terminal for the display of Walsh and Fourier coefficients.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. R. Kitai for his supervision of the entire work and the valuable help in the writing and presentation of this thesis.

Thanks are also due to Dr. A.S. Gladwin, Dr. S.K. Sarna and Dr. P.C. Yip who, as members of the supervisory committee, have offered encouragement and many useful suggestions.

Special thanks are due to my friend Dr. M. Nallasamy at the Indian Scientific satellite Project, India for his initial financial assistance and encouragement which made this work possible.

The completion of this work would not have been possible without the financial assistance of McMaster University and of the National Science and Engineering Research Council of Canada.

The author wishes to thank Miss. P. Dillon of the Word Processing Centre for typing the thesis.

Finally, I am grateful to my father-in-law, Mr. G. Ayyavoo, my wife Kasturi and my son Ashok for their cooperation and constant encouragement during the course of the work.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

$\{\phi\}$	set of orthogonal functions
$x(t)$	analog input signal
T	time interval
M_n	subspace formed by the first n elements of $\{\phi\}$
ϕ_k	kth member in the set $\{\phi\}$
WSA	Walsh Spectral Analyser
$\{\phi\}_W$	set of sequency ordered Walsh functions
wal(k,t)	Walsh function of index k
cal(s,t)	even symmetry Walsh function of sequency s
sal(s,t)	odd symmetry Walsh function of sequency s
i_k, t_k	kth bit of the binary representation of i and t
W_j^k	Boolean representation of sampled Walsh function of index k. j represents sampling instant
g_i^k	ith bit of Gray code representation of k
$\bar{W}$	Walsh matrix
$\oplus$	modulo 2 sum or Exclusive-OR
B(k)	kth sequency ordered Walsh coefficient
A_s	sth cal coefficient
B_s	sth sal coefficient
a_k	kth cosine component
b_k	kth sine component
$\{\phi\}_p$	Paley ordered set of Walsh functions

LIST OF SYMBOLS AND ABBREVIATIONS (continued)

$Pal(k,t)$	kth member of $\{\phi\}_p$
$rad_k(t)$	kth Rademacher function
sgn	signum
$b(i)$	Gray code to binary conversion of i
$\{\phi\}_h$	Hadamard-ordered set of Walsh functions
$wal_h(k,t)$	kth member of $\{\phi\}_h$
$<i>$	Bit-reversal of i
N	number of samples
$X(i)$	Discrete ith sample of x(t)
$L(t)$	output of delta sigma modulator
D_{j+}	Pulse width proportional to X(j)
D_{j-}	Pulse width proportional to -X(j)
$\Delta\theta$	T/N
P_o	Pulse sequence derived from D_{j+}
N_o	Pulse sequence derived from D_{j-}
$\alpha(k)$	Pulse train equivalent to kth Walsh coefficient
$wal'(k,t)$	digital representation of wal(k,t)
FWHT	Fast Walsh Hadamard Transform
FWT	Fast Walsh Transform
$\underline{B}_h(N)$	Hadamard-ordered (Nxl) Walsh coefficient matrix
H(N)	(NxN) Hadamard matrix
$\otimes$	Kronecker product
$B_h(k)$	kth Hadamard-ordered Walsh coefficient

LIST OF SYMBOLS AND ABBREVIATIONS (continued)

A	Adder
S	subtractor
z^{-i}	delay operation by i sampling periods
C_i	ith bit of a binary counter C
P(N)	'Ideal' shuffle operator
I(N)	(NxN) identity matrix
T_p	Processing time
BRO	Bit reversal operation
$S_W(\cdot)$	Permutation operator for sequency ordered Walsh transform
$\underline{\Delta}$	equal by definition
$\bar{i}$	i operated by $S_W(\cdot)$
SR^r	shift right by r places
BR(•)	Bit reversal operator
$\bar{X}(\cdot)$	Data permuted by $S_W(\cdot)$
$B_d(k)$	kth dyadic-ordered Walsh coefficient
$S_D(\cdot)$	Permutation operator for dyadic-ordered Walsh transform
$\hat{i}$	i operated by $S_D(\cdot)$
$\hat{X}(\cdot)$	Data permuted by $S_D(\cdot)$
Gl ... GI	Logic elements
LCD	Level crossing detector
T_S^e	input signal period
T_C	clock period

LIST OF SYMBOLS AND ABBREVIATIONS (continued)

$a_{f,s}$	sth cal coefficient of cos $2\pi ft$; an element of the conversion matrix in fth row and sth column
$\underline{b}$	(N/2 x 1) sine coefficient matrix
F_b	sal to sine conversion matrix
F_a	cal to cosine conversion matrix
$x_b(t)$	band-limited signal
K	compensation matrix
F(k)	kth discrete Fourier coefficient
$\hat{F}_b$	the Walsh to Fourier conversion matrix for the discrete case
DFT	Discrete Fourier Transform
DIT	Decimation in time
DIF	Decimation in frequency
e_R	roundoff error random variable
$\sigma^2_{e_R}$	variance of roundoff error
e_s	scaling error random variable
$\sigma^2_{e_s}$	variance of scaling error
σ^2_{EF}	variance of roundoff and scaling errors combined of a DFT coefficient
σ^2_T	variance of scaling error in an output Walsh coefficient
σ^2_R	variance of roundoff error in an output sine/cosine component by the Walsh to Fourier conversion method

σ_S^2	variance of scaling error in an output sine/cosine component by the Walsh to Fourier Conversion
σ_{EW}^2	variance of error in an output Fourier coefficient by the Walsh to Fourier conversion method

CHAPTER 1

INTRODUCTION

In recent years there has been a growing interest in the study of orthogonal transforms in the area of digital signal processing. This is primarily due to the advances made in the computer technology and special purpose digital processors. Research efforts and application of such transforms include image and speech processing, selection in pattern recognition [1,2,3,4,5], analysis and design of communication systems [6], generalized Wiener filtering [7] and spectroscopy.

The basic idea behind the study of complete systems of orthogonal functions $\{\phi\}$, is that any function $x(t)$ defined in an interval $(0-T)$ which is square summable within the interval can be approximated by means of an orthogonal projection on the subspace M_n spanned by the first n elements of $\{\phi\}$,

$$x(t) \approx x_n = \sum_{k=1}^{n} \langle x(t), \phi_k \rangle \phi_k \qquad (1.1)$$

with the property that the mean square error can be made arbitrarily small by choosing n sufficiently large [8] (Bessel's inequality). ϕ_k in Eqn. (1.1) is the kth member

of the set $\{\phi\}$. The inner product is defined by,

$$\langle x(t), \phi(t) \rangle = \frac{1}{T} \int_0^T x(t)\, \phi(t)\, dt$$

The important advantages of a function representation by orthogonal series expansions are: (i) there are many orthonormal sets for which formulas for the nth element of the set have been worked out and are given in standard texts, (ii) the invariance of the inner product, and (iii) the ease of extending the projection to a subspace M_{n+1} without the need to be wholly recomputed if the projection on M_n is known; only $\langle x(t), \phi_{n+1} \rangle$ need to be evaluated. The signal representations of various orthogonal transforms are judged in terms of the mean square error criterion.

The literature abounds with algorithms and machine designs for the computation of the Discrete Fourier Transform [24,25,26], and digital spectral analysers that are based on Fast Fourier Transform (FFT) algorithms are commercially available [27,28]. While a Fourier analyser permits the characterization of signals and systems in the frequency domain, an alternative representation in terms of bi-valued orthogonal Walsh functions has led to the Walsh Spectral Analyser (WSA) which finds applications for the following reasons:

1. It can be efficiently implemented on either general-purpose or special-purpose computers due to the

binary nature of Walsh functions.

2. In certain applications, such as image processing [1], fast and computationally efficient Walsh analysis may be preferred over Fourier analysis.
3. For band limited signals, a measured Walsh Spectrum may be transformed to yield corresponding Fourier spectrum. The total process of Walsh Spectrum measurement followed by conversion to Fourier spectrum is faster than the Cooley-Tukey FFT algorithm in situations where up to 64 data samples are used [23]. For data lengths greater than 64, this method is superior to the FFT method in applications where the number L of Fourier components is relatively small compared with N. The number of multiplications in that case is approximately NL/6. A 64-point data analysis suffices in many applications such as amplifier distortion measurement, biomedical signal analysis, power-system spectral measurements and some areas of pattern recognition.

This thesis deals mainly with the design of a Microprocessor-based WSA using a fast transform algorithm. The approach followed is different from earlier instrumentation for this purpose [15,16,17] in that the input sequence is permuted using minimal LSI hardware, and a Fast Walsh Hadamard Transform (FWHT) is implemented.

Apart from the display, the instrument is contained entirely within an off-the-shelf Intel System 80/10 which consists of a cabinet containing an 8080 based single-board computer SBC 80/10, a card cage with ready-wired bus and power supply lines for up to three additional boards, and a power supply. The cabinet dimensions are 43.2 cm x 8.9 cm height, 50.8 cm depth (17 x 3.5 x 20 inches). Only one additional specially-designed board is used: It contains circuits for trigger level adjustment, A/D conversion, 1K of read-write memory, a frequency multiplier module (FMM) and a direct memory access (DMA) controller.

The FMM circuit [21] generates 64 equally-spaced pulses within one period of the input signal, so that the data window spans one cycle. The leakage error in the Fourier spectrum measurement is made negligibly small. Another feature of the circuit is that it does not require a high-speed clock for its proper functioning. The signal frequency may lie anywhere within the range 0.2 Hz to 10 KHz without any range switching. A measurement uses two cycles of input signal, the first being used for frequency determination and the second cycle for data aquisition. This is followed by processing of 24 msec duration, to yield the Walsh spectrum.

Given the Walsh coefficients of the signal, the corresponding Fourier coefficients are obtained by way of a

matrix multiplication so that both spectra are available. The Walsh to Fourier conversion process uses the matrix multiplication [22],

$$[a] = [K]^{-1} [F_a] [\underline{A}] \qquad (1.2)$$

$$[b] = [K]^{-1} [F_b] [\underline{B}] \qquad (1.3)$$

where

$[\underline{a}]$ = (N/2 x 1) cosine coefficient matrix

$[\underline{b}]$ = (N/2 x 1) sine coefficient matrix

$[\underline{B}]$ = (N/2 x 1) sal coefficient matrix

$[\underline{A}]$ = (N/2 x 1) cal coefficient matrix

$[K]^{-1}$ = diagonal (N/2 x N/2) compensation matrix

$[F_a]$, $[F_b]$ = (N/2 x N/2) conversion matrices for cosine and sine components, respectively.

The conversion process takes about 1 sec, most of the time being devoted to software multiplication.

The results (Walsh and Fourier components) computed are in binary form. They are converted into signed BCD numbers and transmitted to a Hewlett Packard graphics terminal type 2647A with processing capability for displaying numerical and graphic data. The Walsh/Fourier spectrum of a sinusoidal signal as displayed on the graphics terminal is shown in Fig. 1.1.

The instrument uses less than 2K words of program memory (PROM) and 1K words of RAM. The System 80/10 monitor is not made use of so that space is left for additional 2K

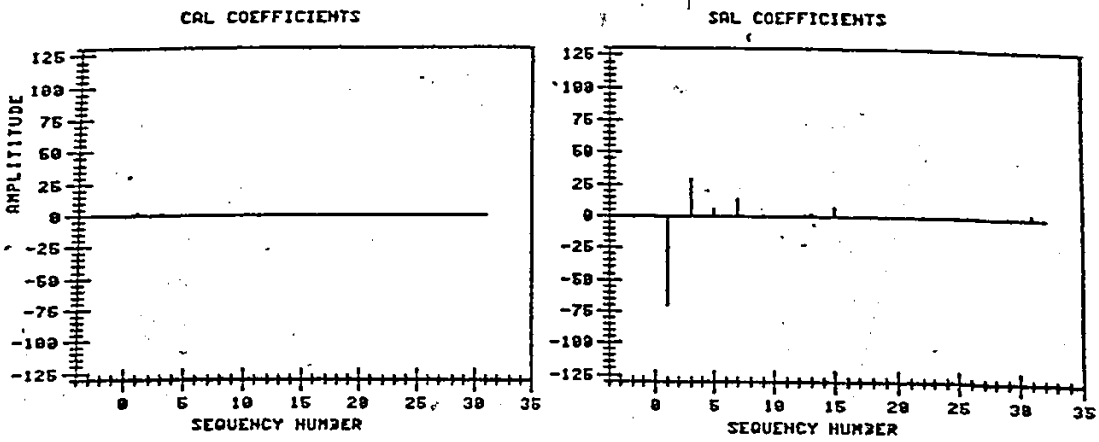


Fig. 1.1(a) Display of Walsh spectrum of 1kHz sinusoid

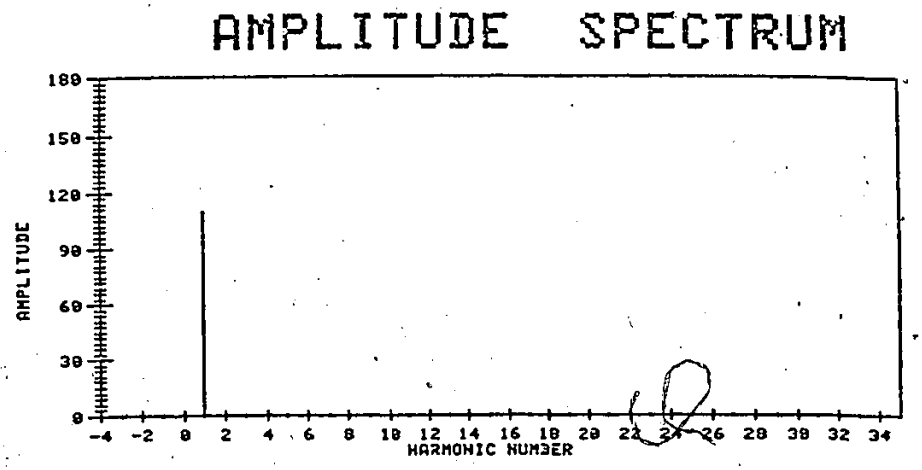


Fig. 1.1(b) Fourier amplitude spectrum of the same signal after Walsh to Fourier Conversion

words of ROM. Also two additional boards may be plugged into the card cage. The system therefore lends itself to extension to include other measurements at low cost. The use of a System 80/10 package and LSI controller type 8257 for DMA were found to reduce development costs and time in a significant way.

Walsh functions and their properties are introduced in Chapter 2. These functions are used in different orderings, so there is the need to define the orderings and to establish the relationship between them.

Numerous designs for a WSA are reported in the literature [9,13,14,15,16,17]. A brief review of the existing designs is presented in Chapter 3. To aid the description, the WSA's are classified into direct and Fast transform type. Long shift registers are well suited to the implementation of FWHT. Improved design techniques to adapt the FWHT processor to yield the Walsh coefficients in dyadic and sequency ordering are presented; they reduce the processing time when compared to an earlier method [17]. Also a pipeline architecture using identical stages is evolved.

Chapter 4 deals with the organization of the Microprocessor based Walsh-Fourier Spectral Analyser. A detailed description is given of the design of each major section shown in the block diagram of the instrument.

The process of obtaining the Fourier coefficients from the Walsh coefficients was first reported by Siemens and Kitai [22] and more recently by Tadokoro and Higuchi [23], using a different algorithm. The two methods are compared in Chapter 5.

The Walsh to Fourier conversion process is compared with the FFT methods with respect to the memory requirements and the effects of finite word length in addition to the number of multiplications given in reference [23].

The software of the WSA is described with the aid of flowcharts in Chapter 6. Here the discussion is limited to two special purpose subroutines, viz, WTFORM for the computation of Walsh transform and WFCON for the Walsh-to-Fourier Conversion. Data structure used in the system and its effect on the accuracy of the final results are discussed.

The limitations of the present design and suggestions for improvement are given.

CHAPTER 2

WALSH FUNCTIONS: DEFINITIONS AND ORDERINGS

2.1 Introduction

Numerous definitions, methods of generation and application of Walsh functions have appeared in the literature. This chapter is concerned with definitions and the notations used. The definitions outlined here find use in the design of Walsh functions generator. The properties of Walsh functions that are necessary to develop algorithms for the computation of Walsh coefficients and the Walsh-to-Fourier conversion process are outlined. The significance of different orderings of Walsh functions and the relationships between them are discussed. These aid the development of digital processors to yield coefficients in different orderings.

2.2 Walsh functions: Definition

The set of Walsh functions $\{\phi\}_w$ was originally defined by J.L. Walsh in an interval (0-1) recursively [29],

$$\phi_0(t) = 1 \qquad 0 \leq t < 1$$

$$\phi_1(t) = 1 \qquad 0 \leq t < 1/2$$

$$= -1 \qquad 1/2 \leq t < 1$$

$$
\begin{aligned}
\phi_2^1(t) &= 1 && 0 \le t < 1/4,\ 3/4 \le t < 1 \\
&= -1 && 1/4 \le t < 3/4 \\
\phi_2^2(t) &= 1 && 0 \le t < 1/4,\ 1/2 \le t < 3/4 \\
&= -1 && 1/4 \le t < 1/2,\ 3/4 \le t < 1 \\
\phi_{n+1}^{2k-1}(t) &= \phi_n^k(2t) && 0 \le t < 1/2 \\
&= (-1)^{k+1}\,\phi_n^k(2t-1) && 1/2 \le t < 1 \\
\phi_{n+1}^{2k}(t) &= \phi_n^k(2t) && 0 \le t < 1/2 \\
&= (-1)^k\,\phi_n^k(2t-1) && 1/2 \le t < 1
\end{aligned}
\qquad (2.1)
$$

where

$$k = 1, 2, \ldots, 2^{n-1},\ n = 1, 2, \ldots, \infty.$$

The functions in the set $\{\phi\}_w$ exhibit alternately even and odd symmetry with respect to the middle of the interval. The even and odd Walsh functions are given distinct notation, cal(s,t) and sal(s,t) repectively [6]. There is a close connection between sal and sine functions as well as between cal and cosine functions. They are related to wal(k,t) by

$$\text{cal}(s,t) = \text{wal}(k,t),\ k = 2s \qquad (2.2a)$$

$$\text{sal}(s,t) = \text{wal}(k,t),\ k = 2s-1 \qquad (2.2b)$$

where s is called the sequency [30]; that is one half the average number of zero-crossings per second.

Using this terminology, Harmuth [6] developed a recursive definition of Walsh functions in the form of a difference equation

$$\text{wal}\ (2k + p, t) = (-1)^{[k/2]+p} \{\text{wal}\ (k, 2(t + 1/4)) + (-1)^{k+p}\ \text{wal}\ (k, 2(t - 1/4))\} \quad (2.3)$$

where $p = 0$ or 1, $i = 0, 1, 2, \ldots$, $\text{wal}\ (0,t) = 1$ for $-1/2 \leq t < 1/2$; $[k/2]$ represents the integer part of $k/2$.

This definition covers only the interval $-1/2 \leq t < 1/2$; periodic Walsh functions can be formed by duplicating the functions over each successive interval. The first 8 Walsh functions in the interval 0 to 1 are shown in Fig. 2.1.

Another approach is to introduce Walsh functions through a definition in terms of the binary representation of indices. The formula due to Pratt, Kane and Andrews [31] is

$$\text{wal}(k,t) = (-1)^{\sum_{i=1}^{\infty} (k_i + k_{i+1})\ t_i} \quad (2.4)$$

where k is an integer between 0 to 2^{m-1} with the binary representation

$$k = (k_m\ k_{m-1} \ldots k_2\ k_1)_2$$

and t is a real number between 0 and 1 with the binary expansion

$$t = (t_1\ t_2 \ldots t_i).$$

Since $k_i = 0$ for all $i > m$, only the first m digits of t appear in Eqn. (2.4). Hence, $\text{wal}(k,t)$ can be represented by

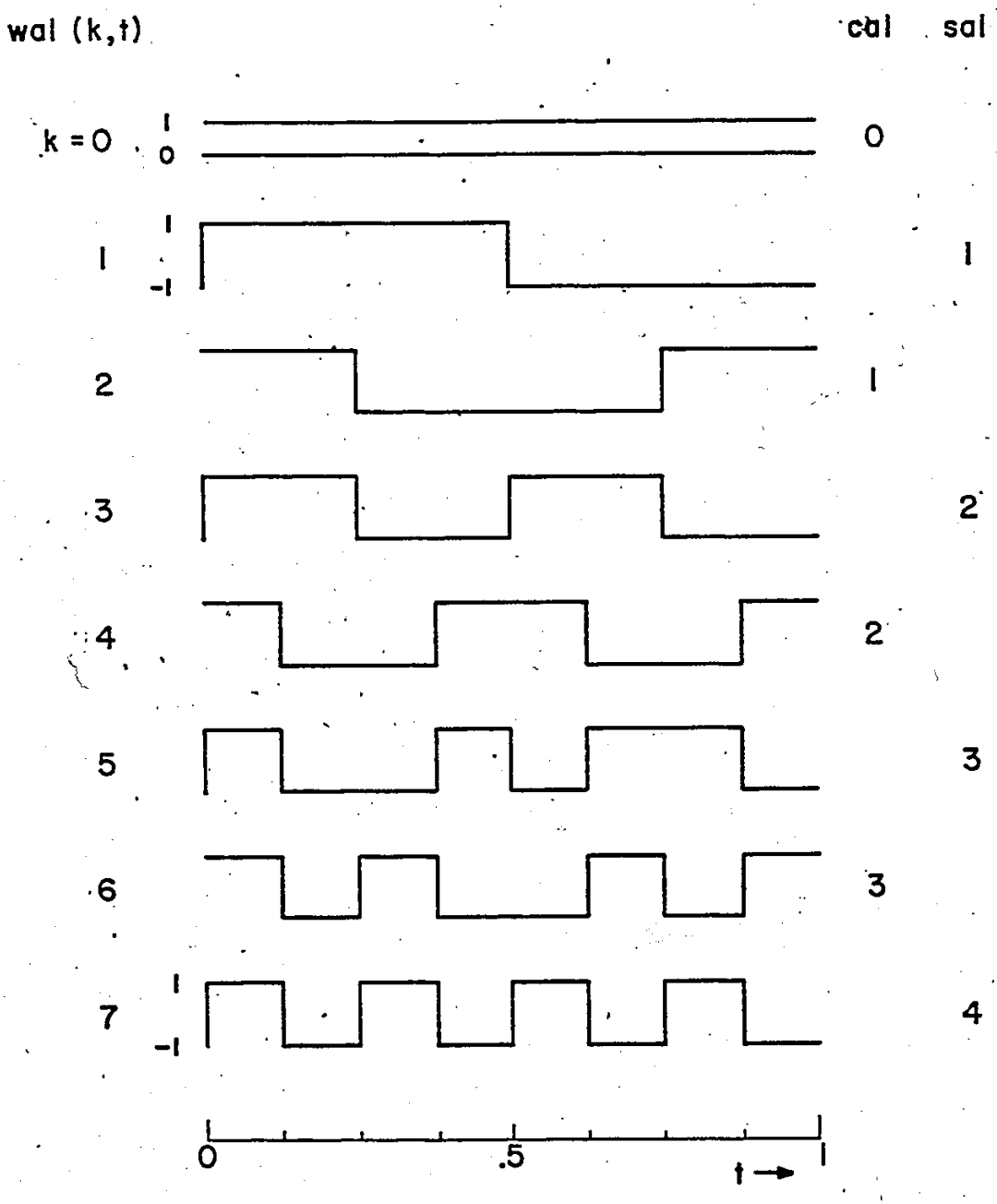


Fig. 2.1 The set of eight Walsh functions

a 2^m bit string W_j^k, $0 \leq j < 2^m$. Choosing to denote 1 by Boolean 0 and -1 by Boolean 1, we have

$$W_j^k = \sum_{i=1}^{m} (j_i + j_{i+1})\, k_{m-i+1} \bmod 2 \qquad (2.5)$$

j being an integer constructed by taking the first m binary digits of t.

The set of 2^m Walsh functions can be thus represented by a $(2^m \times 2^m)$ Boolean matrix which is a useful representation in the computation of the Discrete Walsh transform. The Boolean matrix can be considered as the sampled version of the continuous set of wal(k,t) in Fig. 2.1, the samples being taken at $t = j/2^m$ for $j = 0, 1, \ldots, 2^m-1$. As an example the Boolean matrix $\bar{W}$ for m = 3 is given in Fig. 2.2. The expression within the brackets represents binary to Gray code conversion [32]. Thus Eqn. (2.5) becomes

$$W_j^k = \sum_i g_i^k\, j_{m-i+1} \qquad (2.6)$$

where g_i^k denotes the ith bit of Gray code representation of k. The design of some Walsh function generators [33,34,35, 36] are based on Eqn. (2.6).

2.3 Properties of Walsh Functions

The properties of Walsh functions which can be established using the definitions given before are:

$$\overline{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \begin{matrix} \text{wal}(0,t) \text{ ; cal}(0,t) \\ \text{wal}(1,t) \text{ ; sal}(1,t) \\ \text{wal}(2,t) \text{ ; cal}(1,t) \\ \text{wal}(3,t) \text{ ; sal}(2,t) \\ \text{wal}(4,t) \text{ ; cal}(2,t) \\ \text{wal}(5,t) \text{ ; sal}(3,t) \\ \text{wal}(6,t) \text{ ; cal}(3,t) \\ \text{wal}(7,t) \text{ ; sal}(4,t) \end{matrix}$$

Fig. 2.2 Walsh-ordered discrete Walsh functions

for m = 3

(i) Walsh functions form a complete set of mutually orthogonal functions in the unit interval and hence the possibility of using them for signal approximation or representation. Due to the orthogonality of Walsh functions, the following relation applies:

$$\int_0^1 \text{wal}(k,t)\ \text{wal}(i,t)\ dt = \begin{cases} 0 & k \neq i \\ 1 & k = i \end{cases} \qquad (2.7)$$

It can be shown that the product $[\bar{W}]\ [\bar{W}]^T$ is $2^{-m}[I]$ where $[I]$ is the identity matrix.

(ii) $\text{wal}(k,j) = \text{wal}(j,k)$. Hence $\bar{W}$ is symmetric.

(iii) The product of two Walsh functions yields another Walsh function (closure property)

$$\text{wal}(i,t)\ \text{wal}(k,t) = \text{wal}((i \oplus k),\ t)$$

where $i + k = \Sigma_r\ 2^r (i_r + k_r)_{\text{mod}2}$ and i_r, k_r represent the rth bit of the binary representation of i and k respectively. As an example consider the multiplication of wal(6,t) and wal(12,t). Using the binary representation for 6 and 12, one obtains 10 for modulo 2 addition:

$$\begin{array}{rl} 6 & 0110 \\ +\ 12 & \underline{1100} \\ 10 & 1010 \end{array}$$

The multiplication of Walsh functions is associative:

$$[\text{wal}(i,t)\ \text{wal}(k,t)]\ \text{wal}(\ell,t) = \text{wal}(i,t)\ [\text{wal}(k,t)\ \text{wal}(\ell,t)]$$

Walsh functions form a group with respect to multiplication. (A group $[G,\cdot]$ is an algebraic structure, where G is a set

and $\cdot$ is a composition on that set such that the axioms of associativity, closure, identity, and inverses are valid [37].) The inverse of an element in the set $\{\phi\}_w$ is the element itself; the identity element is wal(0,t).

2.4 Walsh and Fourier Series Expansion

A signal x(t) defined in an interval (0-T) can be represented in terms of the sequency ordered Walsh series according to

$$x(t) = \sum_{k=0}^{\infty} B(k)\ \mathrm{wal}(k,t) \tag{2.8}$$

where the kth Walsh coefficient B(k) is given by

$$B(k) = \frac{1}{T}\int_0^T x(t)\ \mathrm{wal}(k,t)\ dt \tag{2.9}$$

Eqn. (2.8) can be written in terms of cal and sal coefficients A_s, B_s in the form

$$x(t) = A_o + \sum_{s=1}^{\infty} \{A_s\ \mathrm{cal}(s,t) + B_s\ \mathrm{sal}(s,t)\} \tag{2.10}$$

where

$$A_s = \frac{1}{T}\int_0^T x(t)\ \mathrm{cal}(s,t)\ dt \tag{2.11}$$

$$B_s = \frac{1}{T}\int_0^T x(t)\ \mathrm{sal}(s,t)\ dt \tag{2.12}$$

A Fourier series representation of the signal is given by

$$x(t) = \frac{a_o}{2} + \sum_{k=1}^{\infty} \left[a_k \cos \frac{2\pi kt}{T} + b_k \sin \frac{2\pi kt}{T}\right] \qquad (2.13)$$

where cosine and sine coefficients a_k and b_k are defined by

$$a_k = \frac{2}{T} \int_0^T x(t) \cos \frac{2\pi kt}{T} \, dt \qquad (2.14)$$

$$b_k = \frac{2}{T} \int_0^T x(t) \sin \frac{2\pi kt}{T} \, dt \qquad (2.15)$$

2.5 Ordering of Walsh Functions

Walsh functions in three different orderings are used [38,39,40,41]. The mathematical nature of Walsh functions is such that no universally used ordering exists. This is in sharp contrast to the complex exponential functions that are always ordered by their frequency. For the complex exponentials are characters of the group under modulo N addition, which has a natural ordering by arithmetic value. Geometrically we may consider integers to be points on the real line, which has only one dimension and hence one ordering. On the other hand, Walsh functions are characters of the dyadic group, which is the group of binary vectors under bit-wise addition modulo-2. The space formed by these vectors, called dyadic space, has many dimensions but only two points along each axis (0 or 1). Since there is no natural ordering of the dyadic group, there is no single ordering of Walsh functions. The three types of orderings

used are: (i) sequency or Walsh, (ii) dyadic or Paley, and (iii) Natural or Hadamard.

2.5.1 Sequency or Walsh Ordering

This is the ordering which was originally employed by Walsh, and is characterised by the fact that wal(k,t) has k sign changes in the interval (0-1). The number of sign changes is used as generalised frequency [30]. This ordering resembles the ordering of sinusoidal orthogonal functions with increasing frequency and is favoured by researchers in most engineering applications [42,43].

Fast algorithms for computing Walsh coefficients in sequency ordering require data shuffling. Details are given in Chapter 3, where transform techniques are considered.

2.5.2 Dyadic or Paley Ordering

This ordering was first used by Paley [7] when he proposed alternatives to the original definition of Walsh. Let the set of Walsh functions in dyadic order be denoted by

$$\{\phi\}_p = \{Pal(k,t)\}_{k=0}^{N-1} \qquad (2.16)$$

Index k in Eqn. (2.16) represents the kth member in the set. The merits of Paley ordering of Walsh functions are [44],

1. Pal(k,t) has a simpler recursive definition:

$$\begin{aligned} Pal(2k,t) &= Pal(k,2t); && 0 \le t < 1/2 \\ &= Pal(k,2t-1); && 1/2 \le t < 1 \end{aligned} \tag{2.17}$$

$$\begin{aligned} Pal(2k+1,t) &= Pal(k,2t); && 0 \le t < 1/2 \\ &= -Pal(k,2t-1); && 1/2 \le t < 1 \end{aligned} \tag{2.18}$$

In contrast, the sign to be attached to wal(2k,t) and wal(2k+1,t) must be determined by splitting k into two parts $2^m + i$, $0 \le i < 2^m$ and using

$$\begin{aligned} wal(2k,t) &= (-1)^{i+1}\, wal(k,2t-1) && 1/2 < t < 1 \\ wal(2k+1,t) &= (-1)^{i}\, wal(k,2t-1) \end{aligned} \tag{2.19}$$

2. Pal(k,t) can be expressed as a product of Rademacher functions:

$$Pal(k,t) = \prod_i [rad_i(t)]^{k_i} \tag{2.20}$$

where

$$k = \sum_{i=0}^{n} k_i\, 2^i;\ k_i \in (0,1)$$

Rademacher functions are square waves:

$$rad_i(t) = \text{sgn}\,[\sin(2^{i-1} 2\pi t)];\ i = 1, 2, \ldots, 3 \tag{2.21}$$

3. Gibbs [44] showed that Walsh functions are eigen solutions of logical differential equations, and the equation that gives Pal(k,t) is simpler.

4. Dyadic ordering is preferred in filtering applications due to the existence of dyadic convolution theory, analogous to arithmetic covolution used in linear

time-invariant system.

5. Yuen [45] computed the Walsh spectra of a large class of continous functions, including exp(t), sin(t), cos(t) and several powers of t and compared each spectrum in sequency order with that in dyadic order. He showed that in general the dyadic-ordered spectrum shows better convergence towards the higher-order end. The good convergence property of a dyadic ordered Walsh Spectrum has some significant implications in the case of function approximation. It means that an estimate can be made regarding the magnitude of the expansion coefficients and certain higher order coefficients need not be computed. The bounds on the expansion have been derived by Yuen.

Paley ordered Walsh functions are related to sequency ordered Walsh functions according to

$$Pal(k,t) = wal[b(k),t] \qquad (2.22)$$

where b(k) is Gray code to binary code conversion of k [7]. This is illustrated below for m = 3.

Pal(k,t)	k	b(k)	wal(b(k),t)
k = 0	0 0 0	0 0 0	b(k) = 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 1	7
5	1 0 1	1 1 0	6
6	1 1 0	1 0 0	4
7	1 1 1	1 0 1	5

The formula for a Walsh function in Paley order can be written [7],

$$Pal(k,j) = (-1)^{\sum_{i=1}^{m} k_{m-i+1} j_i} \tag{2.23}$$

k_i, j_i are the ith binary digits in the binary representation of k and j.

2.5.3 Natural or Hadamard Ordering

A third ordering called Natural or Hadamard ordering was proposed by Henderson [89]. It has the advantage that the computation of a finite number of Walsh coefficients (normally an integral power of 2) from a given data sequence is faster than that for dyadic or sequency-ordered coefficients. The algorithms used (and called Fast Walsh Hadamard Transform) are discussed in the next chapter. The set of Walsh functions in this order

$$\{\phi\}_h = \{wal_h(k,t)\}_{k=0}^{N-1} \tag{2.24}$$

is related to the sequency ordered Walsh functions by [7]

$$wal_h(k,t) = wal[b(<k>),t] \tag{2.25}$$

where <k> is obtained by bit reversal of k and b<k> is the Gray code to binary code conversion of <k>. For purposes of illustration, Eqn. (2.25) is evaluated for N = 8 and is given below [7].

$wal_h(k,t)$	k	<k>	b<k>	wal[b(<k>),t]
k = 0	0 0 0	0 0 0	0 0 0	b(<k>) = 0
1	0 0 1	1 0 0	1 1 1	7
2	0 1 0	0 1 0	0 1 1	3
3	0 1 1	1 1 0	1 0 0	4
4	1 0 0	0 0 1	0 0 1	1
5	1 0 1	1 0 1	1 1 0	6
6	1 1 0	0 1 1	0 1 0	2
7	1 1 1	1 1 1	1 0 1	5

The formula for a discrete Walsh function in Hadamard order is given by [7]

$$wal_h(k,j) = (-1)^{\sum_{i=1}^{m} k_i j_i} \qquad (2.26)$$

where k_i and j_i are the binary digits of k and j as used before.

CHAPTER 3

WALSH SPECTRAL ANALYSER

3.1 Concepts of the Walsh Spectral Analyser (WSA)

The purpose of the WSA is to compute the Walsh coefficients $\{B(k)\}_{k=0}^{N-1}$ of a signal defined in a finite interval (0-T) as the inner products of a set of orthogonal Walsh functions $\{wal(k,t)\}_{k=0}^{N-1}$ and the input signal x(t), by

$$B(k) = \frac{1}{T} \int_0^T wal(k,t)\ x(t)\ dt \qquad (3.1)$$

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

In the case of sampled input signals, integration in Eqn. (3.1) is replaced by summation. If there are N uniformly spaced samples in the interval (0-T), Eqn. (3.1) can be modified to

$$B(k) = \frac{1}{T} \sum_{i=0}^{N-1} wal(k, \frac{iT}{N})\ X(\frac{Ti}{N})\ ; \qquad (3.2)$$

$$i = 0, 1, \ldots, N-1$$

where X(Ti/N) is a sample of x(t) at the instant Ti/N.

Taking the sampling interval T/N as unity, Eqn. (3.2) reduces to

$$B(k) = \frac{1}{N} \sum_{i=0}^{N-1} wal(k,i)\ X(i) \qquad (3.3)$$

Varied approaches are followed to obtain a finite number of Walsh coefficients defined by Eqns. (3.1) and (3.3). They are discussed in sections 3.2, 3.3.

3.2 Review of Walsh Spectral Analysers

3.2.1 Analog Processors

Walsh Spectral analysers can be classified into either Analog or Digital type depending on the nature of elements used in building the instrument. Early processors were of the analog type, for example the design of Gethoffer [46] uses an analog shift register to store N sampled values which are combined in an operational amplifier (one for each coefficient) to obtain the output coefficient as an analog voltage. This process is illustrated for B(1) in Fig. 3.1. Harmuth [6] employed similar methods in the design of sequency filters.

Analog transformation is suitable for relatively fast real-time operation and for small data length. But it is not attractive for waveform analysis where many parallel outputs are desired from the view-points of accuracy (particularly at low frequencies), economy and reliability. The complexity of digital integrated circuits (ICS) grows with time. The advent of the Microprocessor as a general-purpose digital electronic block, whose function is determined by programming, is revolutionary. We can look

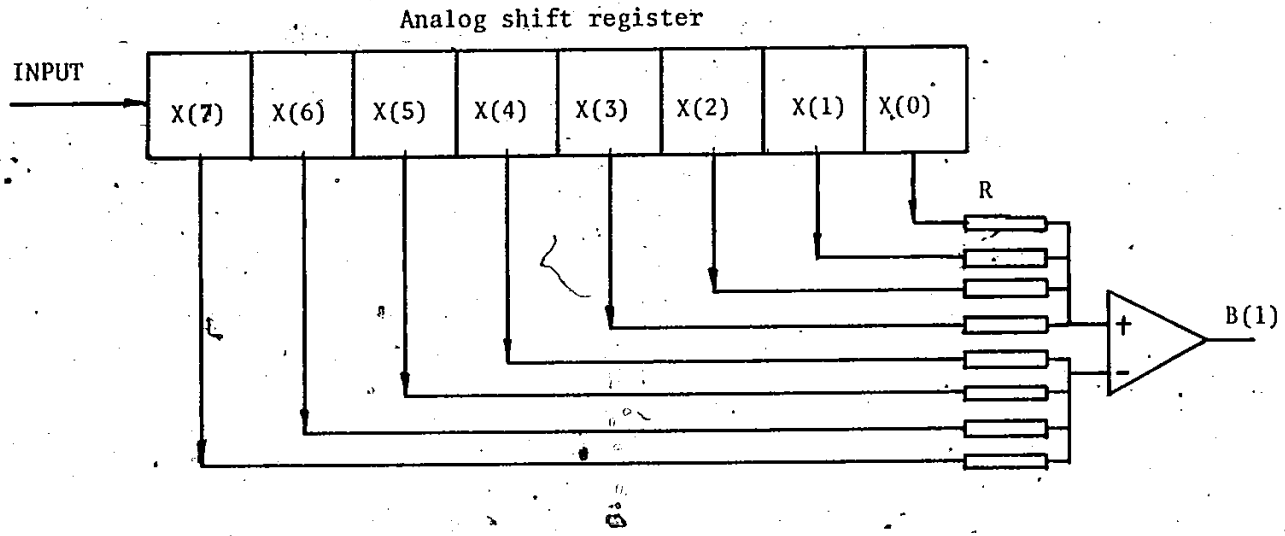


Fig. 3.1 Schematic diagram of an Analog Processor for computing B(1) given eight data points X(0),.....X(7) .

for many technological innovations to improve digital circuit performance, cost, reliability, and flexibility. Low frequency signal analysis is not a problem with digital techniques. These considerations favour the choice of digital techniques in the design of instrumentation.

3.2.2 Digital Processors

Digital WSA's are classified into two viz 1) Direct method, 2) Fast Walsh Transformation method. Both accept N discrete quantized input samples and compute Walsh coefficients according the Eqn. (3.3); but the computing procedures differ.

Direct Methods [9,13,47]

A direct type WSA is schematically illustrated in Fig. 3.2. Each input signal sample is multiplied with a Walsh function of desired sequency (the multiplication reduces to that of determining the sign of the result as Walsh functions take only values ±1); the intermediate results are accumulated in an adder/subtractor. To obtain N coefficients, an N-ary Walsh function generator, N adder/subtractors are required. Walsh function generators are also required for transmission of signals by Walsh carriers and in signal synthesis using inverse Walsh transform apparatus and in real-time Walsh spectrum measurements.

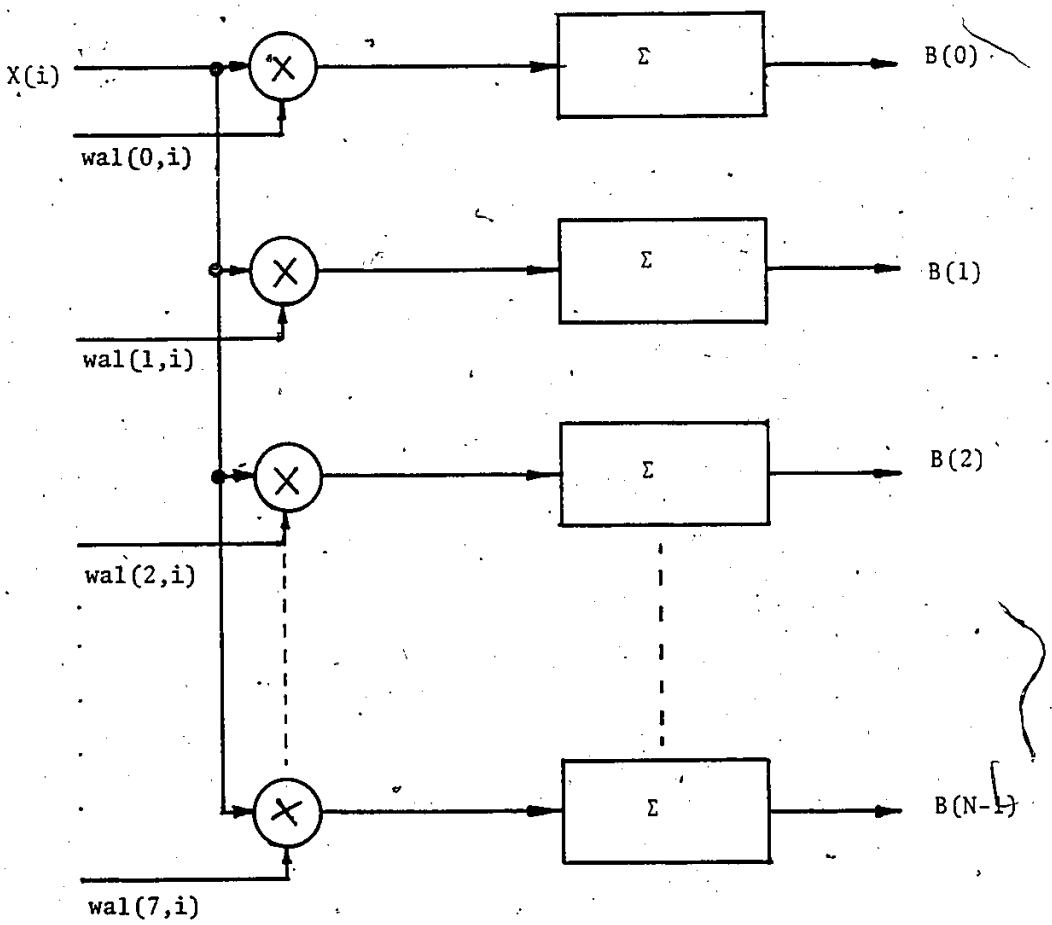


Fig. 3.2 Schematic diagram of a N-coefficient Direct-type Walsh Spectral Analyser

Many contributions have been made to Walsh function generators [33,34,36,48], many of which have been reviewed [50].

A WSA of this type was first designed by Siemens and Kitai [9,13] and recently by Ashouri and Constantinides [15]. The instrument developed by Siemens is designed to yield Walsh spectral coefficients of periodic waveforms in real-time. The computation takes two cycles of input signal; the first cycle is devoted to measure the period of the input signal using digital methods and this period information is used to generate Walsh functions during the second cycle. An array of 64 Walsh functions are generated simaltaneously, using a hazard-free type generator [36]. The generator is clocked by 64 pulses generated by a "Digital Frequency Multiplier" during the second cycle. The input sample is multiplied by Walsh functions and the multiplied outputs are accumulated, there being one accumulator for each coefficient. A special purpose accumulator using counter methods is used. The digital values of samples obtained from an Analog-to-Digital converter (A/D) are fed into the accumulator in a serial manner, starting with the least significant bit. The accumulator has the merit of giving the final result in the desired binary coded decimal form.

The frequency range of input signal analysed was

limited to about 30 Hz by the digital frequency multiplier circuit. An improved multiplier to extend the frequency range was investigated in the present work, and is described in detail in Chapter 4.

The design due to Ashouri and Constantinides [15] follows an approach which is similar to the above, but was designed to compute the Walsh spectrum of Delta-Sigma modulated signals. Delta-sigma modulation is a method of encoding an analog signal x(t) to give at the output L(t) which is bi-valued (±1) according to the following criterion:

$$\int_{t_1}^{t_2} x(t)\, dt = \int_{t_1}^{t_2} L(t)\, dt \qquad (3.4)$$

The kth Walsh coefficient for a delta-sigma modulated signal is then,

$$B(k) = \sum_{t=0}^{N-1} L(t)\ wal(k,t) \qquad (3.5)$$

Since L(t) and wal(k,t) can take only two values ±1, L(t) wal(k,t) is realized using exclusive-OR gates. The output is accumulated in an integrator for coefficients in analog form or in an up-down counter which can be realized by using readily available counter ICS (for an example TTL Bi-directional Counter 74191/74193).

Knauer [47] proposed a Hadamard processor following a

similar parallel approach. His instrument computes only 8 coefficients and was used in video-compression applications. This was considered to be one of the choices in the design of a Real-time Walsh-Hadamard/ Cosine Image Processor by Hein and Ahmed [51].

A hybrid Real-time Walsh waveform analyser differing in the final accumulation process from the previous processors was designed by Tanada and Sano [14], using both analog and digital techniques. Assuming that a N coefficient representation of a signal defined in a period (0-T) is desired, the time interval is divided into N equal parts. The average value of the signal during each sub-interval $\Delta\theta$ is obtained by analog integrators and transformed into a pulse. The pulse width (duration) is given by

$$D_{i+} = h\,\Delta\theta\,(1 + \frac{x(i)}{E}) \qquad (3.6)$$

where $0 < h \leq 1/2$, and E and h are constants. For example, $x(i) = -E$, 0 and $+E$ corresponds to time widths $D_{i+} = 0$, $h\Delta\theta$ and $2h\Delta\theta$. A reversed quantity for $-x(i)$ is necessary for multiplication with Walsh functions. This is represented by

$$D_{i-} = h\Delta\theta\,(1 - \frac{x(i)}{E}) = 2h\Delta\theta - D_{i+} \qquad (3.7)$$

that is, the time width D_{i-} can be obtained from a pulse of width $2h\Delta\theta$ by dividing it into D_{i+} and D_{i-}. With a pulse sequence P_o consisting of D_{i+} with a logic duration of 1 as its width and another pusle sequence N_o consisting of D_{i-},

multiplication can be performed by exclusively selecting either of the sequences according to the sign of Walsh function. Representing +1, -1 of Walsh functions by logic 0 and 1 respectively, the multiplication can be stated as follows:

If a Walsh function assumes logic 0, select P_o; otherwise select N_o for the summation process. The pulses P_o and N_o selected on the above basis during each time interval are combined to get a pulse train $\alpha(k)$ corresponding to the kth Walsh coefficient,

$$\alpha(k) = \sum_{i=1}^{N} \{[P_o \overline{wal'(k,i)}] \cup [N_o\ wal'(k,i)]\} \qquad (3.8)$$

where wal'(k,i) is the digital representation of Walsh function wal(k,i) with the mapping mentioned earlier. The output pulse train $\alpha(k)$ delivered by the multiplier is transformed into an analog voltage representing the Walsh coefficient B(k) by an adder consisting of high-speed current switches with an integrator connected to them. Alternatively time widths D_{i+} can be quantized with a sufficient number of clock pulses and the analyser can be constructed with only digital circuits after time-width transformation. Quantization of time width introduces errors in the final computed coefficients; the error is inversely proportional to the clock frequency used in the process. The summation in Eqn. (3.8) can then be affected

using only up-down counters, similar to the earlier processors discussed, and two's complement arithmetic is avoided.

The direct methods perform N arithmetic operations (only additions or subtractions) for each coefficient (Eqn. (3.3)); a total of N^2 operations are needed for a N-coefficient processor. Fast Transform methods discussed in sections 3.3 perform only $N\log_2 N$ arithmetic operations. As a result, from the computational point of view, direct methods are inefficient when compared with transform methods and are costlier in terms of hardware. In the literature, it is often said that Direct methods are faster and operate in "real-time". The term "real-time" is loosely defined in practice. In electronic systems, processes are described as occurring in real time, only when they keep pace with the chronology of events in the "real world" - i.e. in the environment we perceive events. A processor can collect a finite segment of data and process it; while processing the previous segment it can acquire and store more data. This permits the generation of coefficients that are continuously updated. If the processor is fast enough, a contiguous stream of data is acquired, producing a series of coefficients (each slightly "out of date" when generated) and no data is lost.[7] A processor with this capability can be certainly called a "real-time" processor and can also be

built using the Fast Transform approach using less hardware. Direct methods differ from the Fast Transform methods in that the former type delivers output coefficients instantaneously at the end of data acquisition (zero response time) while in the latter case there is a finite lag between the end of data acquisition and the delivery of output coefficients. High cost of hardware and bulkier equipment using the direct approach may be justified in applications requiring zero response time.

3.3 Fast Transform Algorithms and their Hardware Implementations

The use of Fast Fourier Transform (FFT) algorithms for the fast computation of Discrete Fourier Transform is well known. With the use of Walsh functions in engineering applications, similar fast algorithms to compute Walsh coefficients in Hadamard/Natural order and sequency order (hereafter called Fast Walsh-Hadamard Transform - FWHT and Fast Walsh Transform FWT respectively) were developed.

3.3.1 Walsh-Hadamard Transform (FWHT)

It is well known that an efficient way to implement a FWHT of a N length data sequence ($N = 2^m$) is to decompose the transform matrix H(N) in Eqn. (3.9) into m matrices that have many zero elements, thereby reducing the number of

arithmetic operations [16]. The FWHT is given by

$$\underline{B}_h(N) = H(N)\ \underline{X}(N) \qquad (3.9)$$

where $\underline{B}_h(N)$ = (Nx1) coefficient matrix (vector) in Hadamard order

$H(N)$ = (NxN) Hadamard matrix

$\underline{X}(N)$ = (Nx1) data vector

For example the decomposition for H(8) is as follows:

$$H(8) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \qquad (3.10)$$

$$= G_0 \cdot G_1 \cdot G_2 \qquad (3.11)$$

$$= P\ P\ P \qquad (3.12)$$

$$= Q\ Q\ Q \qquad (3.13)$$

where

$$G_0 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \qquad (3.14)$$

$$G_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix} \qquad (3.15)$$

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad (3.16)$$

$$P = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \qquad (3.17)$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad (3.18)$$

The factorization in Eqn. (3.11) is due to the fact that Hadamard matrix of order $N = 2^m$ can be generated by the successive Kronecker product of core matrix $H(2)^s$ of order 2 which is the lowest order Hadamard matrix [7].

$$H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (3.19)$$

(The Kronecker product $U \otimes R$ of two matrices is obtained by multiplying U with each element r_{ij} of R and substituting the multiplied matrices $r_{ij}\,U$ for the elements of r_{ij} in R.) For example,

$$H(8) = H(2) \otimes H(2) \otimes H(2) \qquad (3.20)$$

The computation of Walsh coefficients using the decomposition in Eqn. (3.11) is known as Fast Hadamard-ordered Walsh-Hadamard Transform (FWHT).

The signal flow-graph shown in Fig. 3.3 illustrates the computational process represented by Eqn. (3.11). It involves three stages (iterations) of matrix multiplication corresponding to G_2, G_1 and G_o in that order. The signal flow-graph in Fig. 3.3 resembles to that of radix-2 Decimation-in-Time FFT algorithms [24]; the multiplication associated with the FFT algorithms are absent in Fig. 3.3. The following remarks on (FWHT) can be made for the general case $N = 2^m$.

1. The number of iterations required is $N \log_2 N$.
2. The ith iteration induces 2^{i-1} partitions of size $N/2^{i-1}$ on N data points.
3. Each iteration consists of repeated application of a basic operation widely known as the butterfly-operation in Digital Signal Processing.

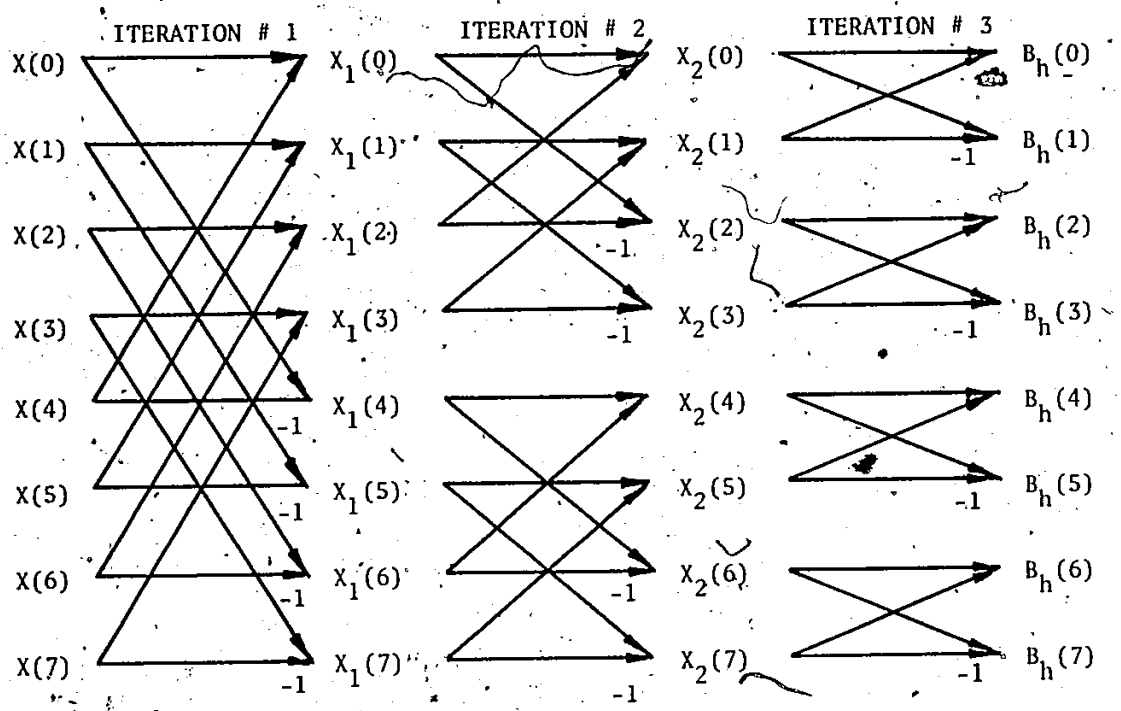


Fig. 3.3. FWHT Signal flow-graph for N = 8.

A butterfly operation combines two inputs to give two outputs according to

$$X(i) = X(i) + X\,(i + N/2^j) \qquad (3.21)$$

$$X\,(i + N/2^j) = X(i) - X\,(i + N/2^j) \qquad (3.22)$$

where $i = 0, 1, \ldots, (N/2)-1$

$j = 1, 2, \ldots, m = \log_2 N$

It is seen from Eqns. (3.21) and (3.22) that intermediate results computed in an iteration are stored in the same memory locations in which original data being transformed were stored. This feature, known as in-place computation, does not require extra memory.

4. The number of arithmetic operations for the process is $N \log_2 N$. (Note that arithmetic operations involve only additions and subtractions in the case of FWHT.)

This algorithm is adopted for software implementation of FWHT in a general purpose computer [7,52] due to the in-place computational advantage (the same approach is adopted in a Microprocessor based Walsh-spectral analyser developed in the present work).

3.3.2 Pipe-line Processor

A pipe-line hard-wired Hadamard Walsh processor was designed by Ashouri and Constantinides [15] using the FWHT algorithm according to the signal flow-graph shown in Fig. 3.3 and is similar to the one described by Gorginsky and

Works [25] for a pipe-line FFT processor. The structure of the processor for N = 8 is given Fig. 3.4. It consists of m stages; each consisting of a shift register SR_i of $N/2^i$ bits, an adder A_i, subtractor S_i and switches SW_i. The switches are controlled by a m-bit binary counter C. Considering the operation of the first stage, SW_1 is in position 1. N/2 data points are entered sequentially into SR_1. At the (N/2 + 1)th sample, switches SW_1 change to position 2 and $-X(i + N/2) + X(i)$ is entered into SR_1, i being the index of the data in the last bit of SR_1. The sum $X(i) + X(i + N/2)$ is formed by A_1 and entered into the second stage. After N samples, SW_1 changes to the position 1; now $-X(i + N/2) + X(i)$ from SR_1 is entered into the second stage. These operations are the ones defined by the iteration number 1 in Fig. 3.3. The operation of the other stages is similar, except that the switches in the succeeding stage operates twice as fast as in the preceding stage; they perform successive iterations of FWHT. The binary output of the counter C identifies the order of the transformed output. It can be seen that there is a delay of N sampling periods between the instants at which the ith sample in a data window of N points is entered into the processor and the ith coefficient is available at the output. A comparison of this structure with a new pipeline structure is given in section 3.3.

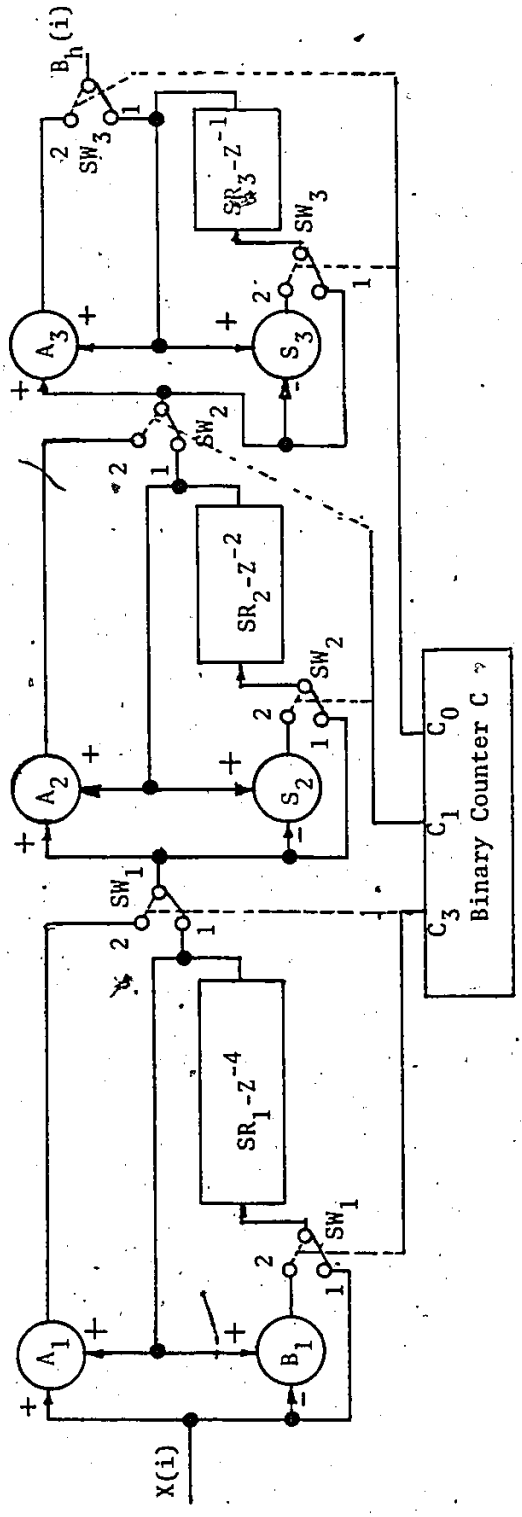


Fig. 3.4 Organization of a Pipeline FWHT Processor

3.3.3 Parallel Array Processor

Elliot and Shum [16] designed a Parallel Array Processor based on the decomposition given by Eqn. (3.12). Their design uses two arrays of N registers (the word array is used in the sense of representing storage elements for N data points); one array for the N input data points and the other for storing intermediate results. Each pair of result registers is connected to a fixed pair of adders/subtractors. To begin with, N data points are acquired in the data registers. At the end of data acquisition, the data registers' outputs are clocked into the result registers. The output of the adders/subtractors are fed back to the result registers based on Eqn. (3.12) and given by

$$X(i) = X(j) + X(j+1) \qquad i = 0, 1, \ldots, N/2 - 1$$
$$X(i + N/2) = X(j) - X(j+1) \quad j = 2i \qquad (3.23)$$

The process of feedback is repeated m times to obtain the Hadamard ordered coefficients. Due to the parallel computation scheme used, its response time is shorter than the previously discussed pipeline structure. For example, 64 coefficients could be obtained in about 200 nanoseconds with TTL ICs. This processor's response time is close to that of the direct type processor. The parallel structure with increased hardware (when compared to Pipeline or Serial organization) can be justified in applications with fast

response time as discussed earlier.

3.3.4 Serial Organization: Shift Register Implementation

A processor which performs the butterfly operations of FWHT algorithms given in Eqns. (3.11 - 3.13) serially is termed as a serial processor. (A pipeline and parallel type processors discussed earlier perform m and N/2 butterfly operations at a time respectively.) This feature could result in a low cost solution to the computation of FWHT and may meet the requirements of many engineering applications. The architecture of a Serial Processor using long shift registers, based on Eqn. (3.13) is illustrated in Fig. 3.5. The computation according to Eqn. (3.13) is to fetch two operands, representing two data points separated by N/2 sampling intervals for a butterfly operation from shift register memory, and to store the intermediate results in another memory. Note that in-place computational advantage of the algorithm in Eqn. (3.11) is lost in this process, but results in identical operations in all iterations.

The processor consists of four shift registers SR1-4 of N/2 bits long, an adder A, a subtractor S and four switches SW_a, SW_b, SW_c and SW_d. At the start, all the switches are in the position 1. The first step is to shift the data X(i), i = 0, 1, ..., N-1 from external memory into SR1 and SR2. It is assumed that external memory is used as

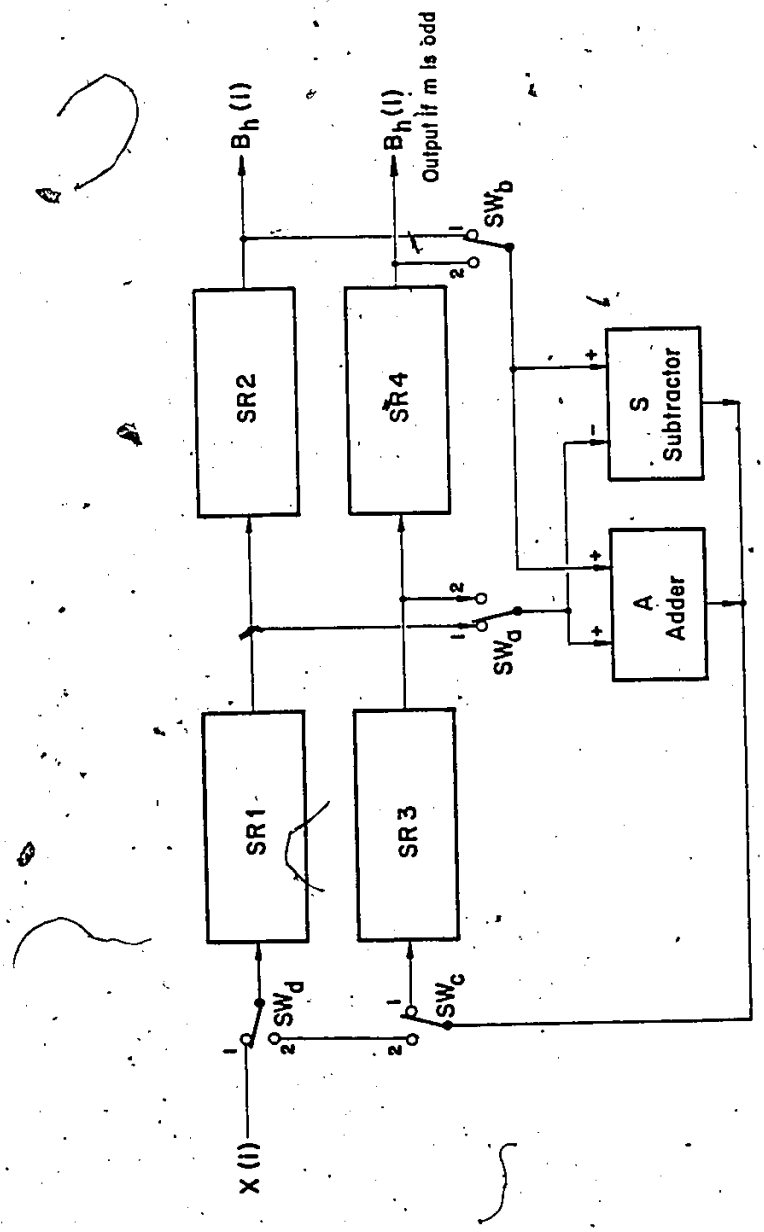


Fig. 3.5 Processor organization using long shift registers

a buffer to acquire data, while the previous data is processed in real-time environment. In non-real time application, data can be entered directly into SR1 and SR2. After N shifts SW_d changes to the position 2 and the first iteration commences. Indexing the data in SR1-SR2 from right to left, X(0) and X(N/2) are operands to A and S. The outputs of A and S are shifted into SR3-SR4 in that order. When S output is shifted into SR3-SR4, new operands to A and S are fetched by shifting the contents of SR1 and SR2. After N/2 operations SW_a, SW_b and SW_c change to position 2; the roles of SR1-SR2 and SR3-SR4 are interchanged. The operands to A and S are obtained from SR3-SR4 and the outputs of A and S are shifted into SR1-SR2. This is repeated m times, transformed coefficients are outputted after m iterations. As the results are outputted serially, new data for processing can be inputted simaltaneously.

Geadah and Corinthios [17] derived a decomposition for the Hadamard transform with a view of implementation using a shift register type organization as follows

$$H(N) = \prod_{i=1,2,\ldots}^{m} P(N)\, D \tag{3.24}$$

where

$$D = I(N/2) \times H(2) \tag{3.25}$$

$$= \begin{matrix} I(N/2) & I(N/2) \\ I(N/2) & -I(N/2) \end{matrix} \tag{3.25a}$$

where I(N/2) is (N/2 x N/2) identity matrix. Expression (3.24) is identical to Q in Eqn. (3.13). P(N) is the 'ideal' permutation matrix, which operates on a vector of dimension N according to

$$P(N)\ \text{col}\ [X(0),\ X(1)\ \ldots\ X\left(\frac{N-2}{2}\right)\ X(N/2)\ \ldots\ X(N-2)\ X(N-1)] \tag{3.26}$$

$$= \text{col}\ [X(0),\ X(N/2),\ X(1),\ X(N/2+1)\ \ldots\ X(N-2)/2,\ X(N-1)]$$

The operations in Eqn. (3.26) are equivalent to storing the results of butterfly operations in Eqn. (3.25) sequentially. In the digital implementation, Geadah and Corinthios employed four shift registers, each N/2 bits long. The outputs of the adder and subtractor resulting in a butterfly operation are shifted in two lower shift registers. After all the butterfly operations are over in an iteration, the results are permuted into upper shift registers according to Eqn. (3.26). This could have been avoided, if the results of the butterfly operation were shifted sequentially into the two lower shift registers cascaded as one register and this set used as data memory for the next iteration as is done in the processor in Fig. 3.5. Since the processor shown in Fig. 3.5 involves only N shifts in an iteration instead of 3N/2 shifts in [17], its processing time is shorter than the latter by a factor of 1.5. The hardware requirements of both processors are roughly the same.

Since an add/subtract operation can be performed in a

few nanoseconds, the processing time is limited by the speed with which data can be entered into shift registers. Let f_s be maximum shifting frequency. The processing time T_p is then mN/f_s. For f_s = 10 MHz and N = 1024, T_p = 10x1024/$(10x10^6)$ = 1.024 msec. This would permit real-time processing at a throughput rate of 1 MHz.

The processor organization shown in Fig. 3.5 can be modified for pipeline processing as shown in Fig. 3.6. It consists of m identical stages; each performing one iteration in Eqn. (5.13). The outputs of A and S are inputted to the next stage. To start with, assume that all the switches are in position 1. The input/output of A and S are entered into upper shift registers. The operands to A and S are from lower shift registers. After N samples are inputted, the switches change to position 2 and the roles of the upper and lower shift registers are interchanged. This organization speeds up the computation by m times, when compared to a single stage structure. For example, the pipeline processor could be used for real-time processing up to 10 MHz throughput rate, assuming the maximum shifting frequency of 10 MHz as before.

A comparison of the performance characteristics of the pipeline structure described here (called Type 1) with that described in Section 3.3.2 (called Type 2) is given below.

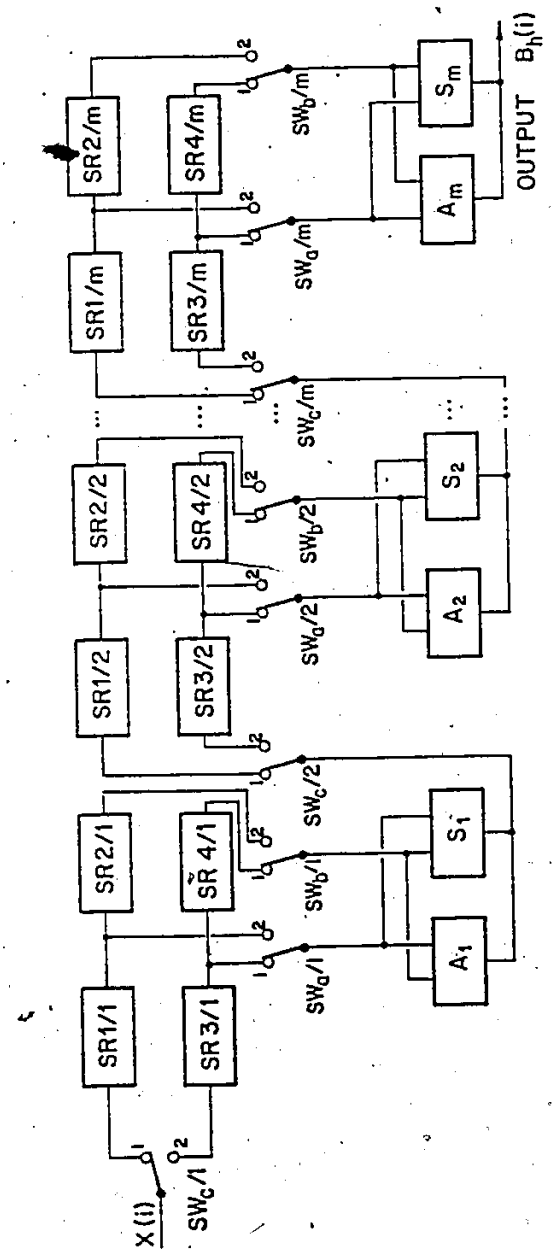


Fig. 3.6 Pipeline Processor

1. The processing speed in a Type 1 processor is limited by the time required to shift one word and perform one addition. In Type 2, there is a cascaded delay of m additions and m shifts. This suggests that Type 1 is faster than Type 2 by a factor of about m.
2. In terms of hardware, both types use the same number of adders/subtractors and switches, but Type 1 uses 2mN memory words and Type 2 uses only N memory words.
3. Type 1 uses identical modules/stages; not so for Type 2.
4. There is delay of mN sampling periods between the incoming data and the corresponding output coefficients in Type 1 and N sampling periods in Type 2.

3.3.5 Fast Walsh Transform Algorithms (FWT) and their Digital Hardware Implementation

One implementation approach is to first compute the FWHT for which efficient algorithms and processors were discussed in the preceding sections. Subsequently, FWT coefficients can be obtained using Eqn. (2.25). However, since this approach involves Gray code-to-binary conversion in addition to bit reversal, it is not a very efficient procedure in the case of software implementation in a computer. In the case of hardware implementation, this

involves more hardware to permute the output coefficients as well as extra processing time. This approach had been used in [16].

A FWT algorithm due to Manz [53] has a signal flow-graph as shown in Fig. 3.7 for N = 8. It is essentially a simple modification of the FWHT signal flow-graph. The first step is to permute the N data points in bit-reversed order. Example: For N = 8, X(1) = X(001), where index 1 is represented by 3-bit binary number is exchanged with X(100) = X(4) due to the bit-reversal operation. This is a time consuming operation [54], if implemented in software. This can be effected through hardware during the data acquisition phase with no increased complexity in the circuits and is followed in a digital processor built in the present work.

The process consists of m iterations and in ith iteration data is partitioned into 2^{i-1} blocks of equal size consisting of $N/2^{i-1}$ points. Two types of butterfly operations are performed alternating with blocks in an iteration starting with the normal type as used in FWHT. The other type is called reversed butterfly operation and is illustrated in Fig. 3.8. Normal butterfly operations are described by

$$X_{k+1}(i) = X_k(i) + X_k(i+2) \quad (3.27a)$$

$$X_{k+1}(i+1) = X_k(i+1) + X_k(i+3) \quad (3.27b)$$

$$X_{k+1}(i+2) = X_k(i) - X_k(i+2) \quad (3.27c)$$

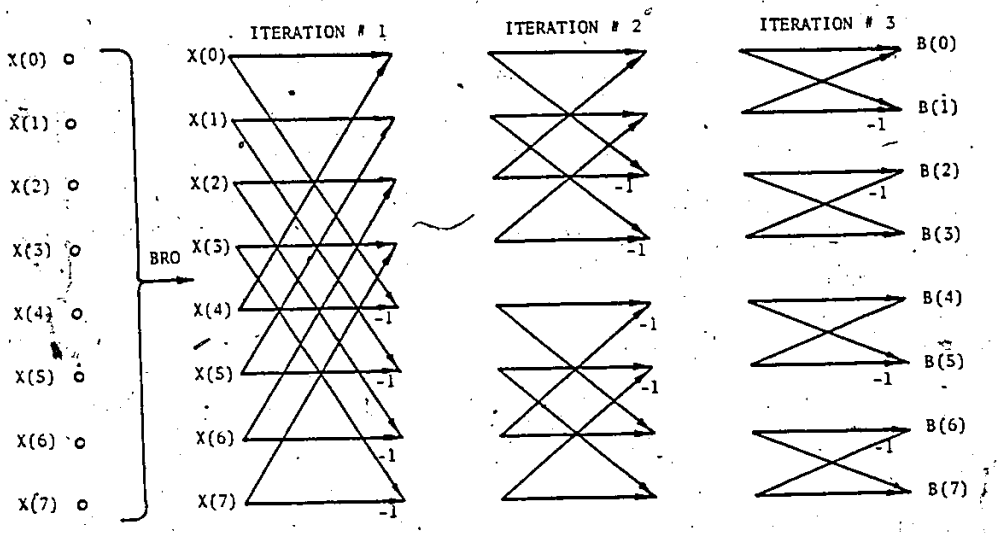


Fig. 3.7 FWT signal flow-graph for N = 8

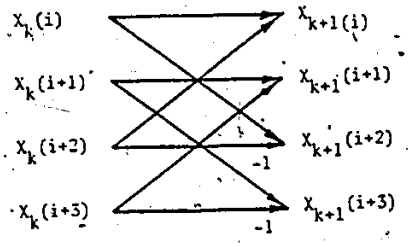


Fig. 3.8(a) Normal butterfly operations

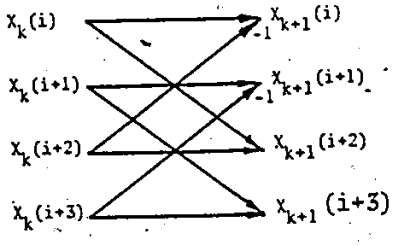


Fig. 3.8(b) Reverse butterfly operations

$$X_{k+1}(i+3) = X_k(i+1) - X_k(i+3) \qquad (3.27d)$$

However, reversed butterfly operations are described by

$$X_{k+1}(i) = X_k(i) - X_k(i+2) \qquad (3.28a)$$

$$X_{k+1}(i+1) = X_k(i+1) - X_k(i+3) \qquad (3.28b)$$

$$X_{k+1}(i+2) = X_k(i) + X_k(i+2) \qquad (3.28c)$$

$$X_{k+1}(i+3) = X_k(i) + X_k(i+3) \qquad (3.28d)$$

Except for these two differences, FWHT and FWT have similar structures. In the case of software implementation, this calls for the incorporation of two subroutines for normal and reversed butterfly operations instead of one in the case of FWHT, as well as provision for the choice between the two in a block. This would increase program memory and processing time slightly, when compared with the FWHT.

The pipeline processor in Fig. 3.4 could be modified to implement the signal-flow graph in Fig. 3.7. The adder and subtractor units in Fig. 3.7 are interchanged periodically. For example, the A and S in the first stage do not interchange; in the second stage they are interchanged for every 4 inputs and in the third stage for every two inputs. But still one needs a bit-reversal permutation operation, before data is entered into the processor. There exists a permutation operator [56] such that FWHT of the permuted data would yield coefficients in sequency order. This would eliminate the need to interchange the adders and subtractors in Fig. 3.7. It also

avoids the use of two types of butterfly operations mentioned earlier in the software implementation. The permutation operator for this objective is developed in Section 3.3.6. In real-time applications, data comprising N samples is acquired first and stored in one segment of memory and then the processing of the acquired data commences. Meanwhile new data is acquired and stored in another segment of input memory. If one could store the data in a permuted sequence, the FWHT would then deliver sequency ordered coefficients. The extra time necessary to permute the FWHT coefficients from a processor is avoided. Also extra memory needed for permutation is eliminated. The performance characteristics of a processor using this approach is compared with that designed by Geadah and Corinthios [17] in Section 3.4.

3.3.6 Permutation Operator for Input Data to Obtain FWT Through FWHT of Permuted Data [56]

The set of N orthogonal basis Walsh functions in Hadamard/Natural order (rows of Hadamard matrix) are defined [43] as

$$wal_h(k,j) = (-1)^{<k,j>} \qquad (3.29)$$

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

$$j = 0, 1, \ldots, N-1$$

where k,j are integers between 0 and N-1 with binary

expansions,

$$k = k_{m-1}\, 2^{m-1} + k_{m-2}\, 2^{m-2} \ldots + k_s\, 2^s \ldots + k_o\, 2^o \qquad (3.30)$$

$$j = j_{m-1}\, 2^{m-1} + j_{m-2}\, 2^{m-2} \ldots + j_s\, 2^s \ldots + j_o\, 2^o \qquad (3.31)$$

$$k_s,\ j_s \in [0,1]$$

and

$$\langle k,j \rangle = \sum_{s=0}^{m-1} k_s\, j_s \qquad (3.32)$$

Index k in Eqn. (3.29) refers to the kth Walsh function in Hadamard order and index j refers to the discrete time with sampling period taken as unity.

$$B_h(k) = \sum_{j=0}^{N-1} wal_h(k,j)\ X(j) \qquad (3.33)$$

Let $S_w(\cdot)$ be a permutation operator which maps a set of integers $\{0, 1, \ldots, N-1\}$ one-to-one into itself (a bijection mapping). It is defined as follows.

Let $$S_w(k) \triangleq \bar{k} \qquad (3.34)$$

Let the m bit binary expansion of k and $\bar{k}$ be,

$$k = k_{m-1}\, k_{m-2} \ldots k_s \ldots k_o \qquad (3.35)$$

$$\bar{k} = \bar{k}_{m-1}\, \bar{k}_{m-2} \ldots \bar{k}_s \ldots \bar{k}_o \qquad (3.36)$$

k and $\bar{k}$ are related as follows

$$\bar{k}_s = k_{m-1};\ \text{for } s = 0 \qquad (3.37)$$

$$= k_{m-s} \oplus k_{m-s-1};\ s \in \{1,2,\ldots,m-1\}$$

The operator $S_w(\cdot)$ performs the bit reversal of the binary to Gray code conversion of k. From Eqn. (2.25), it can be seen that

$$B(k) = B_h(\bar{k}) = B_h(S_w(k)) \tag{3.38}$$

$$= \sum_{j=0}^{N-1} wal_h(\bar{k},j)\ X(j) \tag{3.39}$$

From the definition of $wal_h(k,j)$ in Eqn. (3.29),

$$wal_h(\bar{k},j) = wal_h(k,\bar{j}) \tag{3.40}$$

using Eqns. (3.39) and (3.40),

$$B(k) = \sum_{j=0}^{N-1} wal_h(k,\ S_w(j))\ X(j) \tag{3.41}$$

$$= \sum_{\ell=0}^{N-1} wal_h(k,\ell)\ X(S_w^{-1}(\ell)) \tag{3.42}$$

$S_w^{-1}(\cdot)$ is the inverse operator and is given by [56]

$$S_w^{-1}(k) = \sum_{r=0}^{m-1} \oplus\ SR^r[BR(k)] \tag{3.43}$$

where $\Sigma \oplus$: bit per bit mod. 2 summation

SR^r : shift right r bits with left zero filled

$BR(\cdot)$: Bit reversal operator.

Eqn. (3.42) can be written as follows:

$$B(k) = \sum_{\ell=0}^{N-1} wal_h(k,\ell)\ \bar{X}(\ell) \tag{3.44}$$

$\bar{X}(\ell)$ is the permuted sequence and is given by

$$X(\ell) \rightarrow \bar{X}(S_w(\ell)), \qquad \ell \in \{0,1,\ldots,N-1\} \tag{3.45}$$

The permutation according to Eqn. (3.45) is illustrated by an example for N = 8.

ℓ	ℓ	$S_w(\ell) = \bar{\ell}$	$\bar{\ell}$	$X(\ell)$	$\bar{X}(\ell)$
0	0 0 0	0 0 0	0	$X(0)$	$\bar{X}(0) = X(0)$
1	0 0 1	1 0 0	4	$X(1)$	$\bar{X}(1) = X(7)$
2	0 1 0	1 1 0	6	$X(2)$	$\bar{X}(2) = X(3)$
3	0 1 1	0 1 0	2	$X(3)$	$\bar{X}(3) = X(4)$
4	1 0 0	0 1 1	3	$X(4)$	$\bar{X}(4) = X(1)$
5	1 0 1	1 1 1	7	$X(5)$	$\bar{X}(5) = X(6)$
6	1 1 0	1 0 1	5	$X(6)$	$\bar{X}(6) = X(2)$
7	1 1 1	0 0 1	1	$X(7)$	$\bar{X}(7) = X(5)$

It is also illustrated graphically in Fig. 3.9. It will be seen later that mapping of $X(\ell)$ to $\bar{X}(\ell)$ can be achieved readily through hardware with no extra computing time.

3.3.7 Permutation Operator for Input Data to Obtain Coefficients in Dyadic Order through FWHT of Permuted Data

Let $B_d(k)$ denote the kth Walsh coefficent in dyadic order. Let $S_D(\cdot)$ be a permutation operator which maps a set of integers $\{0, 1, \ldots, N-1\}$ one-to-one onto itself (a bijection mapping). It is defined as follows:

$$\text{Let} \qquad S_D(k) \triangleq \hat{k} \tag{3.46}$$

Let the m-bit binary expansion of k and $\hat{k}$ be

$$k = k_{m-1}\, k_{m-2} \cdots k_o \tag{3.47}$$
$$\hat{k} = \hat{k}_{m-1}\, \hat{k}_{m-2} \cdots \hat{k}_o$$

k and $\hat{k}$ are related by

$$\hat{k}_s = k_{m-1-s} \tag{3.48}$$

Eqn. (3.48) is a bit-reversal operation. From Eqns. (2.22)

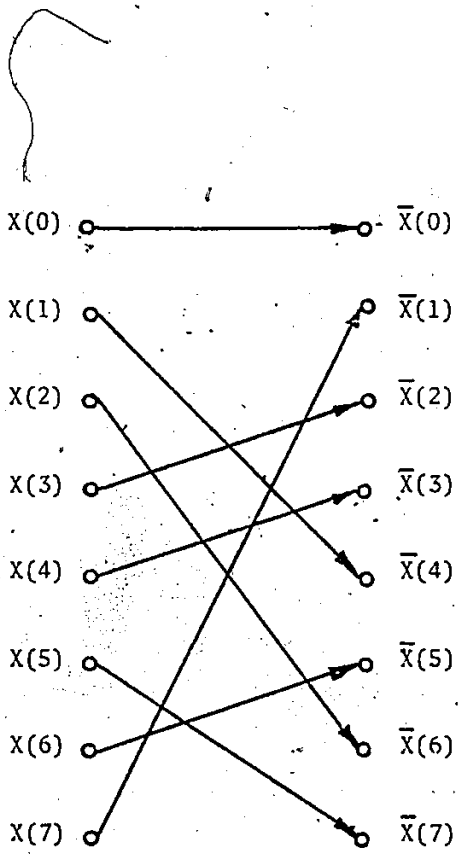


Fig. 3.9 Mapping of $X(\ell)$ to $\bar{X}(\ell)$ for sequency ordered transform by $S_W(.)$

and (2.25) $$B_d(k) = B_h(\hat{k}) \tag{3.49}$$

$$= \sum_{j=0}^{N-1} wal_h(\hat{k},j)\ X(j) \tag{3.50}$$

using Eqn. (3.40) in (3.50),

$$B_d(k) = \sum_{j=0}^{N-1} wal_h(k,S_D(j))\ X(j) \tag{3.51}$$

$$= \sum_{\ell=0}^{N-1} wal_h(i,\ell)\ X(S_D^{-1}(\ell)) \tag{3.52}$$

where $S_D^{-1}(\cdot)$ is inverse operator equal to $S_D(\cdot)$ in this case.

$$= \sum_{\ell=0}^{N-1} wal_h(k,\ell)\ \hat{X}(\ell) \tag{3.52a}$$

where $\hat{X}(\ell)$ is the permuted data sequence. The permutation can be described by

$$X(\ell) \rightarrow \hat{X}(S_D(\ell)) \tag{3.53}$$

$\ell \in \{0, 1, \ldots, N-1\}$

The permutation according to Eqn. (3.53) is illustrated by an example for N = 8.

ℓ				ℓ			$S_D(\ell) = \hat{\ell}$	$\hat{\ell}$	$X(\ell)$	$\hat{X}(\ell)$
0	0	0	0	0	0	0	0	0	$X(0)$	$\hat{X}(0) = X(0)$
1	0	0	1	1	0	0	4	4	$X(1)$	$\hat{X}(1) = X(4)$
2	0	1	0	0	1	0	2	2	$X(2)$	$\hat{X}(2) = X(2)$
3	0	1	1	1	1	0	6	6	$X(3)$	$\hat{X}(3) = X(6)$
4	1	0	0	0	0	1	1	1	$X(4)$	$\hat{X}(4) = X(1)$
5	1	0	1	1	0	1	5	5	$X(5)$	$\hat{X}(5) = X(5)$
6	1	1	0	0	1	1	3	3	$X(6)$	$\hat{X}(6) = X(3)$
7	1	1	1	1	1	1	7	7	$X(7)$	$\hat{X}(7) = X(7)$

Its graphical representation is shown in Fig. 3.10.

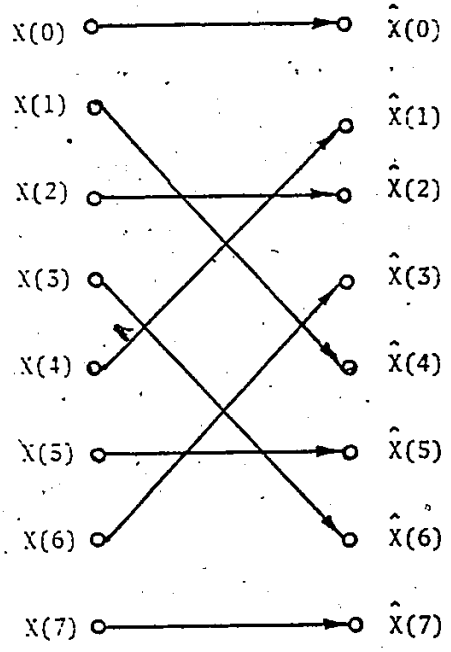


Fig. 3.10 Mapping of $X(\ell)$ to $\hat{X}(\ell)$) for dyadic ordreded transform by $S_D(.)$

3.4 New Design Approach for a Digital WSA to Obtain Coefficients in Natural, Dyadic and Sequency Orderings

In sections 3.3.6 and 3.3.7, the permutation operators $S_W(\cdot)$ and $S_D(\cdot)$ were derived with the view that FWHT of the permuted data sequence would yield output coefficients in sequency and dyadic orderings respectively. This enables one to adapt a FWHT processor for obtaining transform coefficients in different orderings. In designing a digital WSA, the following approach is considered.

1. Discrete samples obtained from a signal are stored in the processor memory in the permuted sequence. Example: X(0) is stored in memory location 0 and X(1) is stored in memory location 7 for sequency order (from Fig. 3.9). In terms of hardware, this involves providing address information for the memory location, assuming that one uses semiconductor Random Access Memory (RAM).

2. The data from RAM is retrieved sequentially and is sent to the serial processor described in section 3.3.4. In the case of software implementation the FWHT on the permuted data is performed as discussed in section 3.3.1.

The schematic diagram of hardware called a Permuting Module to realize data shuffling according to Eqns. (3.45) and (3.53) is described with the aid of Fig. 3.11. It consists of an address counter AC, three tri-state gates G1, G2, G3 and a set of EXCLUSIVE-OR-gates. AC is m bits long

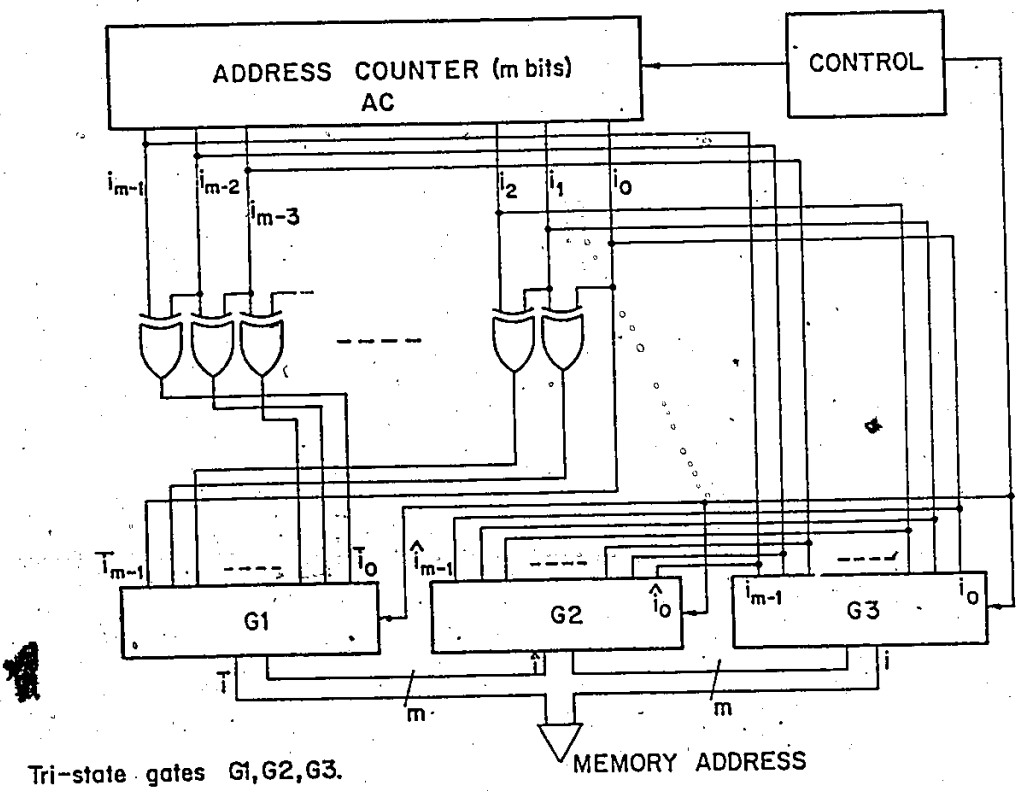


Fig. 3.11 Special purpose Permuting Module

where $m = \log_2 N$. Let the decimal representation of AC outputs be i. From Fig. 3.11, it can be seen that input to G1 is $S_w(i)$ and to G2, it is $S_D(i)$. During the data acquisition phase, one of the three gates G1, G2 and G3 will be enabled by control circuitry; G1 for sequency ordered coefficients, G2 for dyadic ordered coefficients and G3 for Natural ordering. As a consequence, data stored in the memory would represent $\bar{X}(i)$, $\hat{X}(i)$ and $X(i)$ respectively. During data retrieval for further processing in a serial processor described earlier, only G3 is enabled. In the case of computer implementation, PM could be considered as a kind of Direct Memory Access (DMA). This is described in detail in Chapter 4 where the Microprocessor based WSA is discussed in detail.

We compare the new approach with that proposed by Geadah and Corinthios [17]. Their algorithms using shift registers require a reshuffling of the intermediate results of an iteration; the reshufflings vary from iteration to iteration and with output coefficients orderings. This process takes N/2 shift operations for N/2 butterfly operations and an additional N shift operations for data reshuffling in an iteration. This slows down the processing speed by a factor of $3m/2(m+1)$ when compared with the new approach. When pipelined for higher input data rate, the

new design uses similar modules (assuming that one iteration uses one module) in contrast to the use of dissimilar modules in the case of [17].

CHAPTER 4

MICROPROCESSOR-BASED WALSH SPECTRAL ANALYSER

4.1 Design Principles

In a Microprocessor-based Walsh Spectral Analyser, one possible approach to obtain $\{B(k)\}_{k=0}^{N-1}$ is to get $\{B_h(k)\}_{k=0}^{N-1}$ by performing a FWHT through software and subsequently reorder $B_h(k)$ stored in system memory according to Eqn. (2.25). This method is not attractive due to time-consuming reordering operation in software implementation or extra hardware required as stated in Chapter 3. The other approach is to implement Manz's algorithm [53] which requires a reordering of inputs in bit-reversed order. This is time-consuming through software, so a hardware implementation is preferred. Until now the bit-reversal operation has been implemented only through software in Microprocessor-based signal processors [55]. In Section 3.3.6, it is shown that if the data is permuted according to Eqn. (3.45), then FWHT of the permuted data sequence delivers $\{B(k)\}_{k=0}^{N-1}$. The use of two kinds of butterfly operations of Manz's algorithm is also avoided; the processing time and program length are decreased due to the reduced bookkeeping operations by using one kind of butterfly operation. The permutation operation is realized

through a DMA operation during the data acquisition phase. In brief, the design principles are as follows:

(1) The input signal is sampled and converted into digital values. The acquired data sequence is stored in the system memory in a permuted order through DMA.

(2) After the completion of data acquisition, the FWHT is performed according to the signal flow-graph of Chapter 2.

4.2 System Organization

The organization of a Microprocessor-based WSA is shown in Fig. 4.1. For descriptive purposes, the operations of the instrument are divided into four distinct phases viz; (1) Data acquisition, (2) Computation, (3) Control, and (4) Display of results. The operating principles of functional blocks shown in Fig. 4.1 are described in the following sections. The circuit details of these blocks are dealt with in Section 4.3.

4.2.1 Data Acquisition

Fast transform methods employed for spectrum computation accept an input data sequence of N samples and deliver N output coefficients. Normally $N = 2^m$, where m is an integer. As a result, an on-line instrument for this purpose is required to acquire 2^m samples of input signal

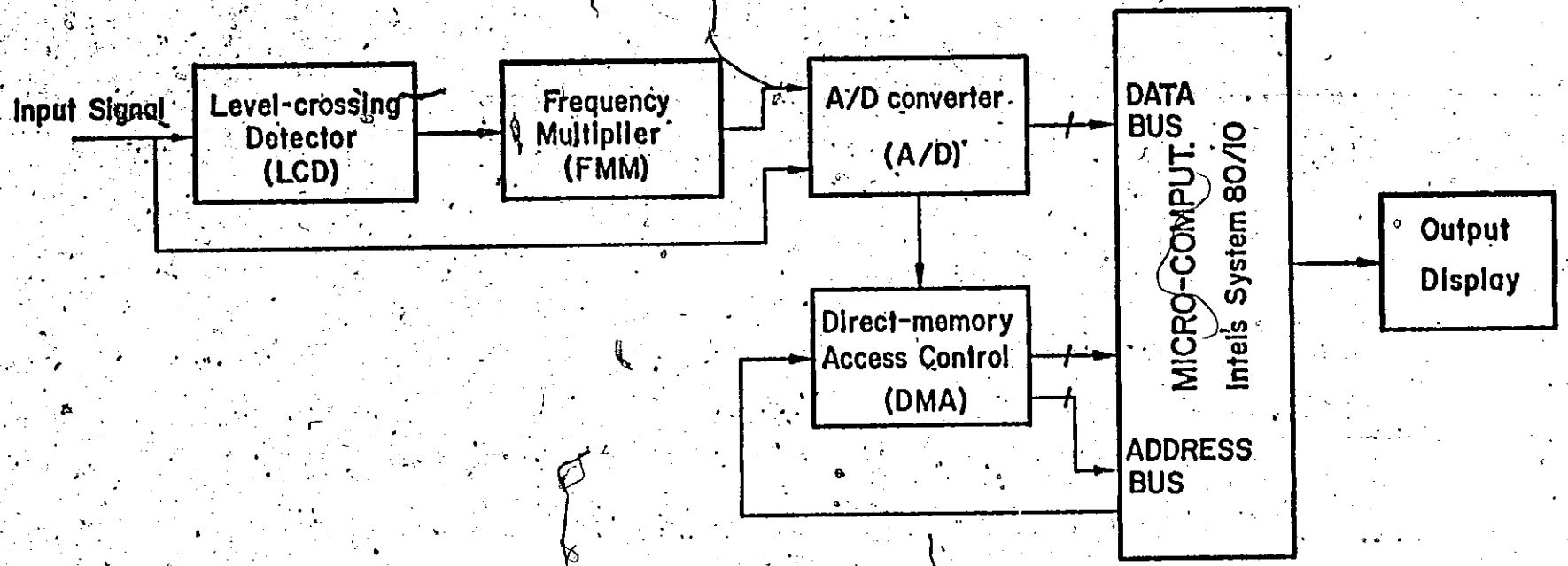


Fig. 4.1 Organization of a Microprocessor-based WSA

for one measurement cycle. For periodic signals, it is desired that 2^m samples span an integral number of periods to avoid leakage errors in the computation of discrete Fourier coefficients [19]. Automatic generation of sampling pulses at this rate is realized using the Frequency Multiplication Module (FMM) shown in Fig. 4.1. The FMM, in conjunction with a level crossing detector (LCD), determines the period T_S of x(t) digitally in the first cycle, after the process is started. Using the obtained period information T_S, equally spaced pulses at a repetition rate of $2^m/T_S$ are generated in the second cycle. The new frequency multiplier circuit [21] is described in Section 4.3.4. A multiplication factor of 64 is used in the system built to obtain a 64-coefficient representation of the signal in a period. The output pulses of FMM form "convert commands" to the Analog to Digital Converter (A/D) employed to digitize x(t). An eight bit A/D converter with output representation in two's complement form is used; this code is well suited to binary arithmetic used in the microprocessor system.

The output of the A/D is stored in Read-write memory (SBC 80/10 uses static RAM) in a permuted sequence according to Eqn. (3.45), starting from a base address in the RAM. Assume that memory locations starting from 0000 to 003F hexadecimal (0-63 decimal) are assigned for data storage.

The 64 output samples obtained from the A/D converter are indexed as X(0000), X(0001), X(0002), ... X(003F) (in 4 digit hexadecimal representation). In the case of sequential storage, X(0000) would be stored in memory location 0000, X(0001) in 0001 and so on. The memory address for the next sample storage can be obtained by a simple increment operation of an address counter. But when the data is to be stored in a permuted sequence according to Eqn. (3.45), the computation of memory address for next sample storage is not straightforward. For example, X(0001) should be stored in memory location that would be occupied by X(0020) in the case of sequential storage. Hence the address of the memory location is to be obtained by adding an offset dependent upon the data index to the base address as shown below for X(0001).

```
  0000 0000 0000 0000   Base address
+ 0000 0000 0010 0000   Offset
  0000 0000 0010 0000   (0020)
```

It can be seen from Eqn. (3.45) that for a data length 64, only the six lsbs of the offset are affected; the remaining bits are all zero. As a result, the addition operation could be avoided by choosing a base address with the six lsbs zero. The six lsbs of the address counter which is incremented by one after the storage of each sample are modified by hardware as shown in Fig. 4.2 so that the final

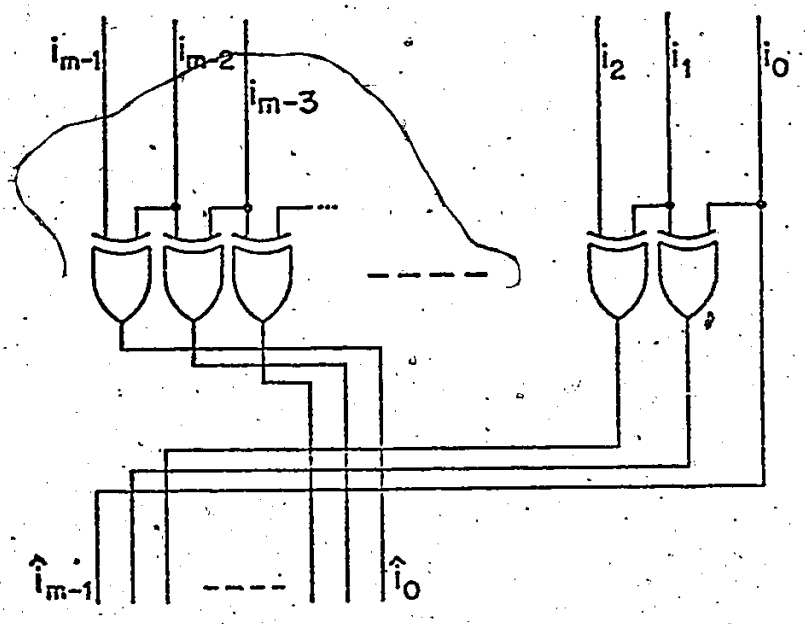


Fig.4.2 Circuit for modification of six lsbs of offset

output contains the proper address for data storage. The DMA block in Fig. 4.1 has been specially designed around the LSI peripheral chip 8257 for this purpose. Hardware details are described in Section 4.3.7.

4.2.2 Computation and Control

The system 80/10 in Fig. 4.1 is used for the following purposes in measuring Walsh coefficients:

(1) To generate control and data information so that external circuit blocks are properly initialized for data acquisition

(2) To perform the FWHT according to the signal flow-graph in Fig. 3.3.

(3) To output the coefficients for display.

These are realized through software residing in System's SBC-80/10 program memory. A brief description of SBC-80/10 and software details are included in Section 4.3.1 and Chapter 6.

In addition to software-generated command signals, various other signals such as the FMM start signal, the control signals for enabling tri-state buffers at appropriate instants, and the DMA termination signal etc. are derived using TTL circuits that are mounted on the special purpose board. They will be described later.

4.3 Hardware Design

4.3.1 System 80/10

The discussion is limited to a functional description. Circuit details are given in reference [57].

The SBC 80/10 is a complete computer system with central processing unit (CPU), system clock, read/write memory, non-volatile Read-only memory (ROM), Input/output (I/O) ports and drivers, serial communication interface, bus control logic and drivers all residing on a 6.75 x 12 inch printed circuit card. The System 80/10 block diagram is shown in Fig. 4.3.

The 8080A contains six 8-bit general purpose registers and an accumulator. The six registers can be addressed individually or in pairs, providing both single and double precision operators. The 8080A has a 16-bit program counter and allows the CPU to address directly 64K memory locations. It has a 16-bit stack pointer register which controls the addressing of a last in/first out stack located within any portion of memory. The stack permits almost unlimited subroutine nesting. Memory and I/O devices can be interfaced through the 16-bit address bus and 8-bit bi-directional data bus.

The 8080A instruction set includes five different types of instructions [58].

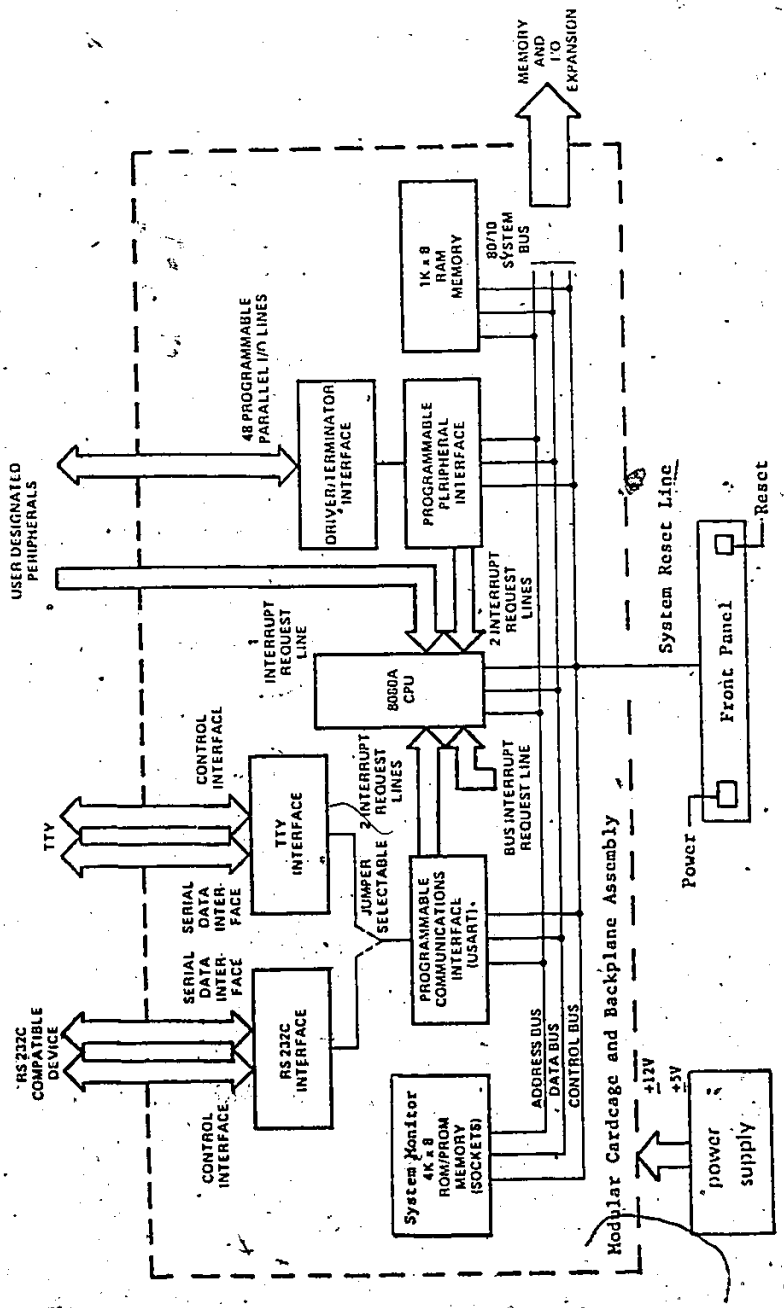


Fig. 4.3 System 80/10 Block diagram

(1) Data Transfer type - move data between registers or between registers and memory.

(2) Arithmetic group - add, subtract, increment or decrement data in registers or in memory.

(3) Logical group - AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory.

(4) Branch group - conditional or unconditional jump instructions, subroutine call instructions and return instructions.

(5) Stack, I/O and Machine control group - including I/O instructions as well as instructions for maintaining the stack and internal control flags.

The execution time of instructions varies from 4 to a maximum of 17 clock cycles; the clock frequency is 2 MHz.

The SBC-80/10 contains space for 1K of 8-bits words, using type 8111 static RAMS; addresses of RAM are from 3COO (hexadecimal) to 3FFF. Non-volatile read-only memory (ROM) of up to 4K, 8-bit words in increments of 1K can be installed in sockets provided for this purpose; addresses are from 0000 to 0FFF. ROMs used in the system are type 8708, erasable and electrically programmable.

Communication between external peripherals and the SBC-80/10 is through 48 programmable parallel I/O lines via two 8255 Programmable Peripheral Interface (PPI). Also serial communication is available via an 8251 universal

synchronous/asynchronous receiver/transmitter (USART). A Teletype is interfaced through this module. The address of parallel and serial I/O ports are given in Table 4.1.

Table 4.1: I/O Addresses of SBC 80/10

I/O Address	Function
E4	PPI1 Port A
E5	PPI1 Port B
E6	PPI1 Port C
E7	PPI1 Control Port
E8	PPI2 Port A
E9	PPI2 Port B
EA	PPI2 Port C
EB	PPI2 Control Port
EC/EE*	USART Data Port
ED/EF*	USART Control Port

* Jumper selectable

The SBC-80/10 has an external bus which includes system data and address buses and a clock. The DMA module and the A/D converter are interfaced to the system SBC-80/10 through this bus. The bus clock which is derived from the oscillator of the processor clock is used in the FMM module; no separate clock is needed. As the bus signals are required for the special purpose circuit board, they are listed in Table 4.2. A HP Graphics Terminal 2647A [59] is

Table 4.2 System bus

PIN ASSIGNMENTS FOR CONNECTOR P1
(System Bus)

	(COMPONENT SIDE)			(CIRCUIT SIDE)		
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Signal GND
	3	VCC	+ 5VDC	4	VCC	+ 5VDC
	5	VCC	+ 5VDC	6	VCC	+ 5VDC
	7	VDD	+12VDC	8	VDD	+12VDC
	9	VBB	- 5VDC	10	VBB	- 5VDC
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15	BPRN	Bus Pri. In	16	▷	
	17	BUSY/	Bus Busy	18	▷	
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknow	24	▷	
SPARES	25	AACK/	Special	26		
	27			28		
	29			30		
	31	CCLK/	Constant Clock	32		
	33	▷		34		
INTERRUPTS	35	▷		36	▷	
	37	▷		38	▷	
	39	▷		40	▷	
	41	▷		42	INTR1/	Interrupt request
ADDRESS	43	ADRD/	Address Bus	44	ADRF/	Address Bus
	45	ADRC/		46	ADRD/	
	47	ADRA/		48	ADRB/	
	49	ADR8/		50	ADR9/	
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
DATA	59	▷	Data Bus	60	▷	Data Bus
	61	▷		62	▷	
	63	▷		64	▷	
	65	▷		66	▷	
	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77	VBB ▷	-10VDC	78	VBB ▷	-10VDC
	79	VAA	-12VDC	80	VAA	-12VDC
	81	VCC	+ 5VDC	82	VCC	+ 5VDC
	83	VCC	+ 5VDC	84	VCC	+ 5VDC
	85	GND	Signal GND	86	GND	Signal GND

▷ Used by Intellec™ MDS Bus.

interfaced through the USART which provides an RS 232C interface for serial I/O data communication.

A software monitor for the system 80/10, contained in two ROMs 8226 is available. This aids the development of application software with the following capabilities:

(1) it gives the user access to console input and output routines as well as paper tape input and output control software.

(2) along with a TTY, it provides access to memory and registers and has control commands to begin execution and to display or alter the contents of memory or registers. The system monitor resides in two ROMS and only 2K of ROM is available for the user. The system monitor is used only for program development. In its final form, the instrument dispenses with the system monitor, leaving space for an additional 2K of ROM space for user's program.

4.3.2 Level Crossing Detector (LCD)

The purpose of LCD is to generate a TTL-compatible square wave signal with period equal to that of input signal x(t) for frequency multiplication purposes. It is designed using a high speed comparator LM311 (G4) as shown in Fig. 4.4(a). When the positive going x(t) crosses V_{ref} set by potentiometer P1, G4 output goes LO marking the beginning of a period. The output goes HI when negative-going x(t)

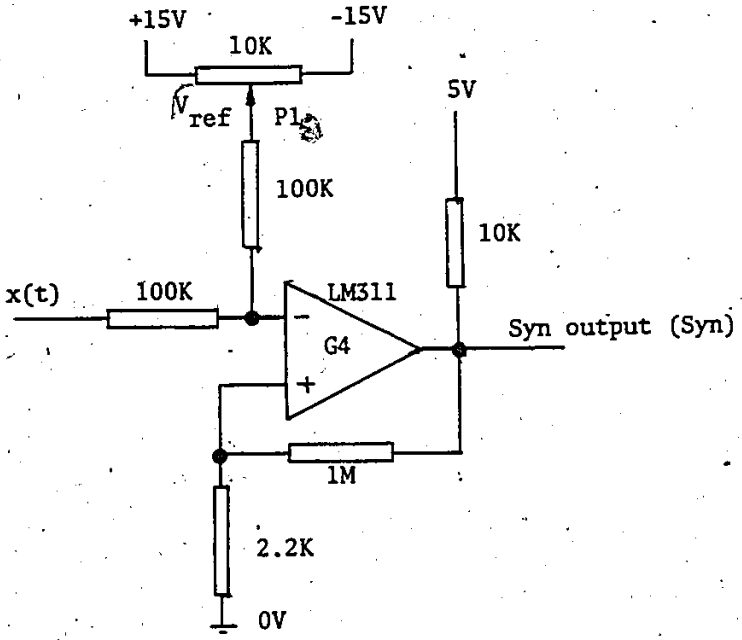


Fig. 4.4(a) Circuit diagram of LCD

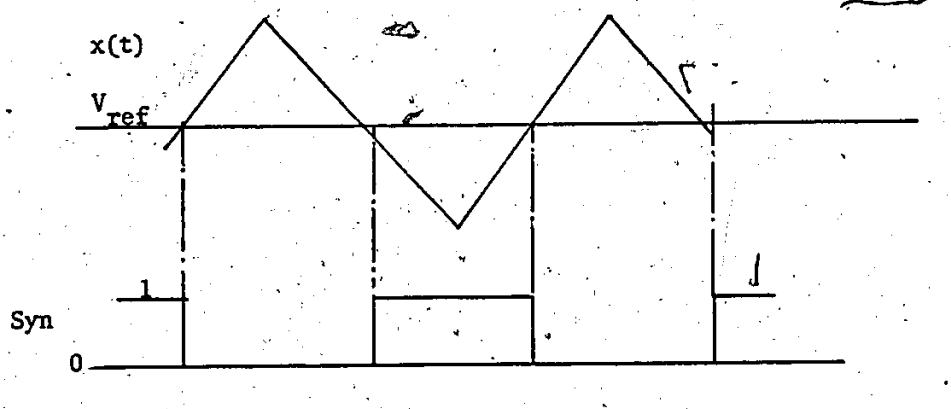


Fig. 4.4(b) Output waveform of LCD

crosses V_{ref}. This is illustrated in Fig. 4.4(b). A 50 mV hysterisis is introduced into the circuit through feedback resistors to prevent false triggering by noise during period measurement.

4.3.3 Frequency Multiplication Module (FMM)

Analog methods which perform frequency multiplication often employ phase-locked loop (PLL) techniques. However the response of PLL is slow and is not satisfactory for low frequency signals. Krishnamurthy et al. proposed an analog method using an analog integrator, a set of comparators and logic gates [60]. The principle of operation is explained with reference to Fig. 4.5 for a multiplication factor of 3. The frequency multiplication system consists of a square wave generator SG, Integrator I, comparators CR1, CR2 and exclusive-OR gates EOR1 and EOR2, all connected as shown in Fig. 4.5. The triangular waveform at the output of I is compared with preset reference voltages VR1 and VR2 equal to 1/3 and 2/3 respectively of the triangular wave amplitude in CR1 and CR2. The output waveforms corresponding to circuit blocks are given in Fig. 4.6. The outputs are then exclusive-ORed in the gate EOR1 and finally the output of gate EOR1 is exclusive-ORed with square wave A. The output waveform F will have a frequency equal to 3 times the input frequency. Though the above system is designed for fixed

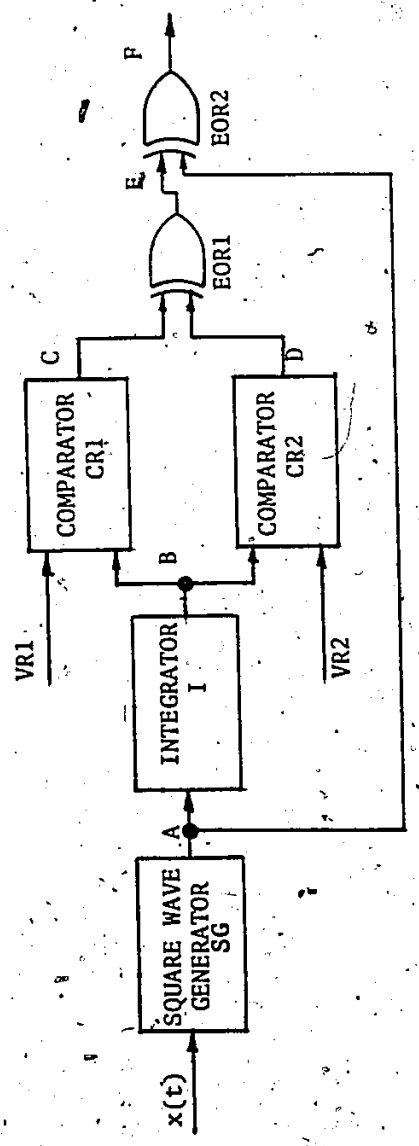


Fig. 4.5 Block diagram of the Frequency trippler

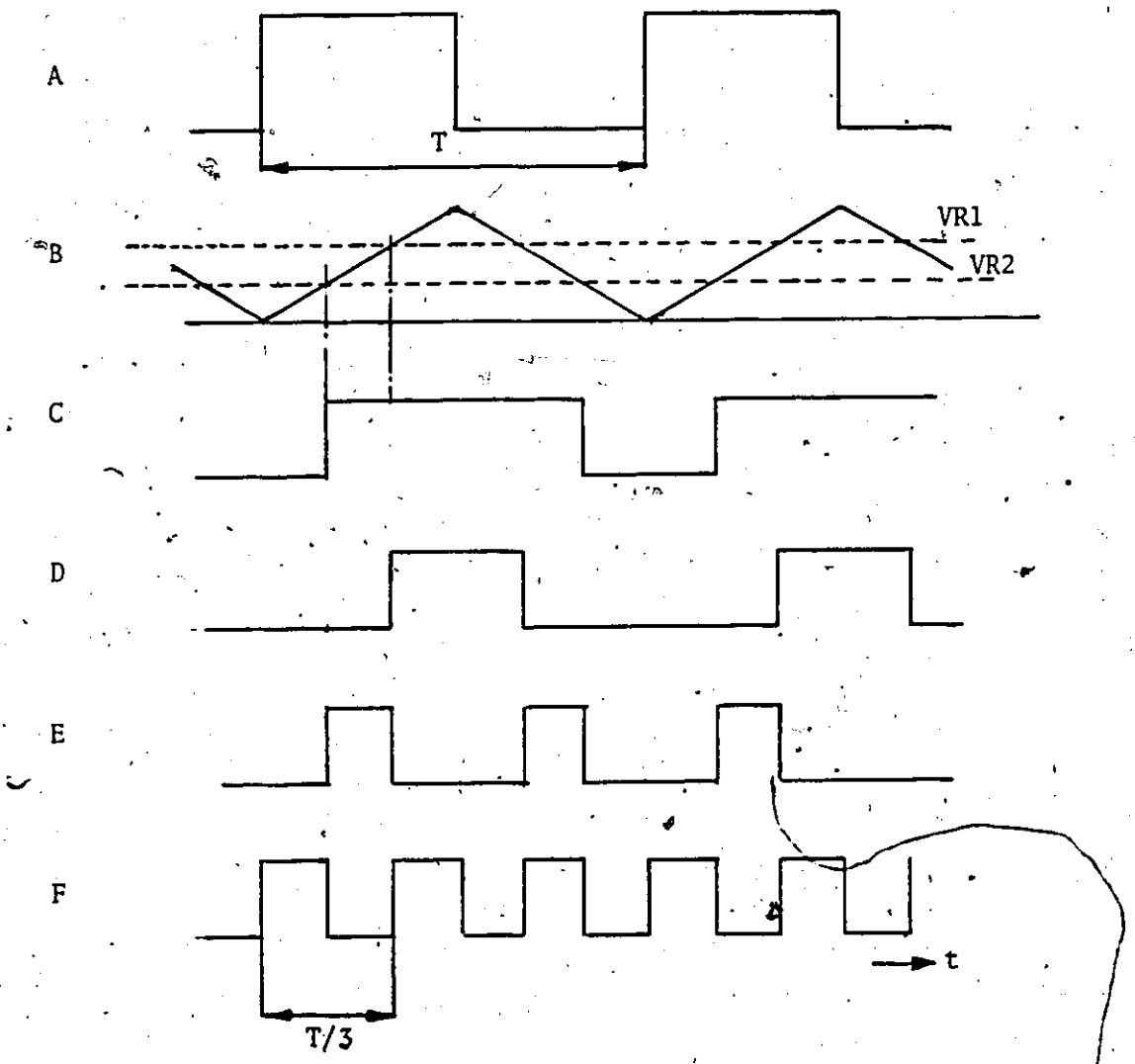


Fig. 4.6 Waveforms of the Frequency trippler circuit

frequency signals, it can be adapted to varying frequency by deriving the reference voltage to the comparators from the output of the Integrator I by using a peak detector [61].

The above principle can be extended to realize frequency multiplication by a factor n by using (n-1) comparators with reference voltages of magnitude V/n, 2V/n, ..., (n-1)V/n where V represents the peak of triangular wave. The low frequency operation may be limited by the accuracy of the integrator. The accuracy of pulse positioning will be poorer than that for digital methods. Also the number of comparators increases with n and may not be attractive in terms of the number of ICs used when compared to digital methods. The frequency range of operation of this system is lower than that for digital methods.

There are also digital methods using rate multipliers but these usually do not generate equally-spaced pulses [62].

A digital technique to produce equally spaced pulses with fast response to changes in input frequency (one period) was first developed by Siemens and Kitai [9] and other systems using the same basic principle have since been reported [63, 65]. These methods are restricted to low frequency input signals by the maximum clocking rate for IC devices; for example, 6.6 kHz maximum at a clock frequency

of 55 MHz, and a multiplication factor of 64 (output pulse frequency: 363 kHz) using Schottky TTL. The output frequency is limited by the speed of TTL counter ICs. Due to the finite set up time for up/down mode control, independent up and down counters are used when operated with a 55 MHz clock. In the new method [21] the maximum output frequency is limited by the settling time of a D/A converter rather than by the speed of digital components. Only one up/down counter is used. In the system built, a low cost D/A converter was used having a 1 μsec settling time. This limits the maximum output pulse frequency to 660 kHz at a clock frequency of only 2 MHz. The maximum output pulse frequency would be increased five times, using the D/A converters with 200 ns settling time that are now available from leading manufacturers. Since the new method has evolved from the earlier one, we first describe the earlier method and its limitations.

Earlier Method

The basic principle of operation may be followed from Fig. 4.7 which includes three binary counters. Counters F and I are up-counters, while down-counter DC is presettable via its load input LD. Initially all three counters are cleared and the clock is connected to counters F and I in cascade. The clock generates pulses of period T where $T << T_S$ (T_S is the period of the input signal). During the first

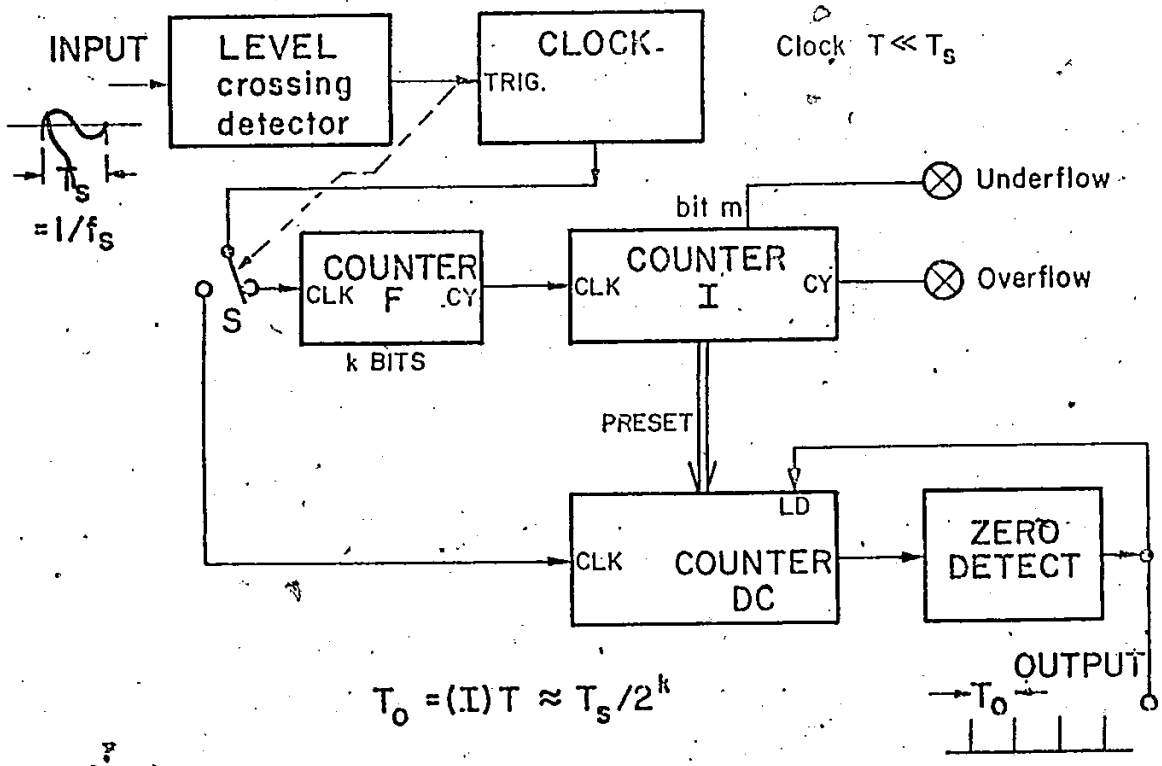


Fig. 4.7 Block diagram of Digital Frequency Multiplier - Earlier metod

cycle of the signal, the beginning of which is marked by a trigger signal from the LCD, the input period is determined in terms of accumulated clock pulse count in F and I. At the end of the first cycle DC is preset by loading the contents (I) of counter I; and switch S is operated so that the clock only feeds counter DC. When the counter contents (DC) reaches zero, an output pulse is generated which also reloads DC with (I). During this phase of operation, DC receives pulses at a rate which is 2^k times faster (k being the number of binary stages of counter F) than the rate for counter I during the first cycle, so there are 2^k uniformly spaced pulses during time T_s. In Walsh spectral measurement applications, a second phase of duration T_s suffices (i.e. the data window spans one cycle of the signal).

In processes of this basic type, an error arises because residue (F) in the counter F is ignored. The output pulse period is $T_o = (I)T$ instead of $(I)T + (F)T/2^k$. The timing error between consecutive output pulses is

$$\frac{(F)T/2^k}{(I)T + (F)T/2^k} \qquad (4.1)$$

which is roughly $-1/(I)$ maximum. This error is made small by increasing the minimum count $(I) = 2^{m-1}$ say, relative to the maximum possible value of (F) viz. 2^k-1. Table 4.3 gives errors for different value of m, based on a multiplication factor of 64. Also the maximum input

Table 4.3 Timing Error of FMM

k	m (bits)	Max. Timing error %	Minimum clock pulse count	Max input frequency for 10 MHz clock
6	8	-0.763	8192	1221 Hz
6	10	-0.192	32768	305 Hz
6	12	-0.048	131072	76 Hz

frequency for a clock frequency of 10 MHz is given for each case.

The counter I has a latched underflow indicator at its mth bit to establish that $(I) \geq 2^{m-1}$ in any measurement. An overflow indicator is also provided. The maximum error in Eqn. (4.1) can be reduced to $\pm 1/2^m$ (magnitude halved, but error which may be positive or negative) by initializing counter F to half range (MSB of F=1) instead of zero [9].

An additional timing error is due to the finite clock frequency which introduces a maximum error -T in the period determination. The normalized maximum timing error is

$$-\frac{1}{2^k(I) + (F)} \qquad (4.2)$$

and is negligible in comparison with Eqn. (4.1).

The new system, a block diagram of which is shown in Fig. 4.8, uses the information contained in the counter F so as to obviate the error given by Eqn. (4.1). At the end of the first cycle the contents of the counter F and I give the

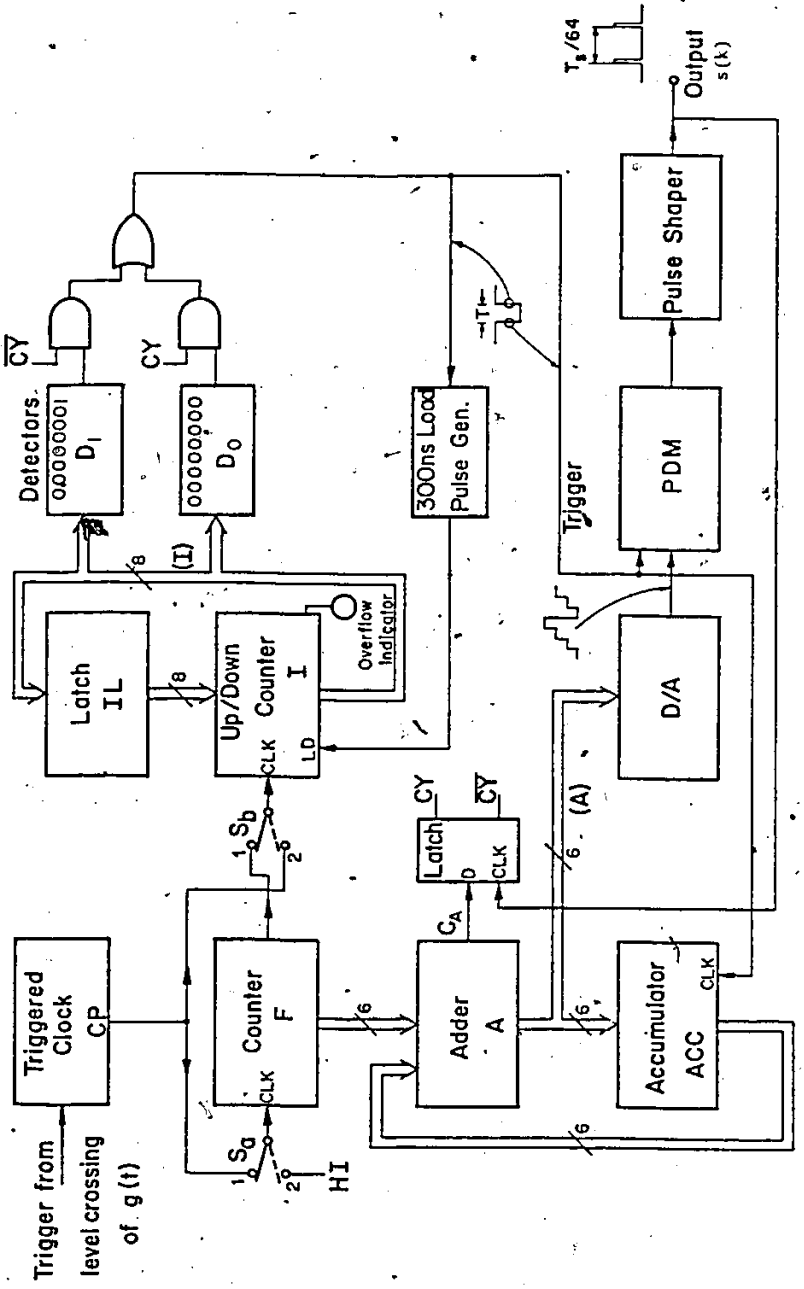


Fig.4.8 Block diagram of Digital Frequency Multiplier - New method

period T_s of the signal in terms of the number of clock pulses, as in the earlier systems. Division by 2^k yields an integer part (I) and a fractional part $\psi = (F)/2^k$. The process is such that successive output pulses are delayed relative to those for the earlier method by Δ, 2Δ, 3Δ, ... where $\Delta = \psi T$ and $0 \leq \psi < 1$. These successively-increasing delays are achieved by means of an accumulating adder shown as A and ACC, a D/A converter and a linear pulse duration modulator (PDM). The up-counter I and the down-counter DC of Fig. 4.7 are replaced by a presettable up-down counter I and a latch IL in Fig. 4.8 which serve the same purpose as before.

Principle of Operation

When initializing the instrument, switches S_a and S_b are in position 1 so that counters P and I are in cascade. The counter I is in the up-count mode, and all digital storage devices are cleared. Before proceeding, we note that if (F) happens to be zero at the end of the first cycle, there should be a zero delay in PDM. It is not practical to make a monostable circuit whose delay commences at zero, so the PDM circuit is designed to provide a delay between T for (F) = 0 and slightly less than 2T i.e. $(2-1/2^k)T$ when (F) is maximum. The circuit uses delay $\Delta = [1+\psi]T$, and to compensate for this, the PDM circuit is triggered when counter I reaches one instead of zero in its

down count mode. The one state of the counter I is detected by D_1 in Fig. 4.8. In the pulse delay process, a carry CY may be produced by adder A. When a CY occurs, the counter is allowed to count down to zero as detected by D_0; this corresponds to a delay of the triggering of the PDM by T secs, as described shortly.

The first phase, of duration T_S, is no different from that of the system in Fig. 4.7. At its termination switches S_a and S_b are changed to position 2, the contents (I) of I are latched in IL and counter I is changed to count down. At the beginning of the second phase, the output of adder A is (F). The digital output of A is converted into an analog voltage by the D/A converter, and applied to the PDM circuit. Counter I proceeds to count down and on reaching one, the PDM circuit is triggered to produce the first output pulse, the PDM delay being $\Delta_1 = [1+\psi]T$. Accumulator ACC is clocked so that (A) changes to 2(F), corresponding to a PDM delay $\Delta_2 = [1+2\psi]T$ and so on. Overflow of adder A first occurs at the rth output pulse which satisfies the condition $\Delta_r = [1+r\psi]T \geq 2T$; then (A) falls to $r\psi-1$, the carry latch CY is set, and D_0 is enabled so that the trigger pulse to the PDM is delayed by one clock pulse. The process of producing uniformly-timed output pulses continues in this way during the second cycle of the input signal; adder A overflows from time to time and the trigger pulse to PDM is

delayed by time interval T whenever it does. The D/A converter output during this cycle is a staircase voltage which falls in value whenever CY = 1.

Regardless of the state of CY, the NOR gate in the detector circuit is low for a time duration T. The trailing edge of this output pulse is used to reload the counter I, so although the PDM circuit is triggered at (I) = 1 when CY = 0, reload takes place when (I) falls to zero. Likewise, when CY = 1, the reload takes place when (I) falls to -1 (two's complement).

Timing of Output Pulses

The duration of the rth PDM output pulse is given by

$$\Delta_r = \{1 + [r\psi]_{\text{mod } 1}\} T \qquad (4.3)$$

Output pulses occur at times t_1, t_2, etc. from the instant at which the second cycle commences:

$$t_1 = [(I)-1]T + \Delta_1$$

$$t_2 = [(I) + (I)-1]T + \Delta_2 = [2(I)-1]T + \Delta_2$$

etc. Also with every overflow of the adder, there is a cumulative delay T in the output pulse timing, so that in general, the time of occurence of the rth output pulse is

$$t_r = [r(I) - 1 + \sum_r CY_r] T + \Delta_r \qquad (4.4)$$

where $\Sigma_r CY_r$ is the sum of carries upto and including r.

From Eqns. (4.3) and (4.4) the interval between any two successive output pulses is,

$$t_r - t_{r-1} = (I)T + \psi T = (I)T + (F)T/2^k \qquad (4.5)$$

which is exactly the desired interval. The expressions (4.3) to (4.5) ignore the time quantization errors in the measurement of the period T_S of the signal in the first phase due to the finite clock frequency, and errors due to nonlinearity and imperfect adjustment of the PDM circuit. It is seen that I fills the role of a coarse digital timer, while F is used for fine timing adjustment. Small errors in the D/A converter or in the PDM circuit disturb the output pulse positions slightly, but they do not give rise to cumulative errors in output pulse timing.

Example

To illustrate the functioning of the circuit, the following are assumed:

Clock frequency 100 KHz ($T = 10$ μs)

Input signal frequency 59.6 Hz ($T_S = 16.7785$ ms)

Multiplication factor $2^k = 64$ (F has 6 bits)

The required PDM characteristic for the above clocking rate is shown in Fig. 4.9. A unit increment in (A) gives rise to a 5/32 μs increment in output pulse duration, and the maximum pulse duration is $10(2-1/64)$ μs < 20 μs.

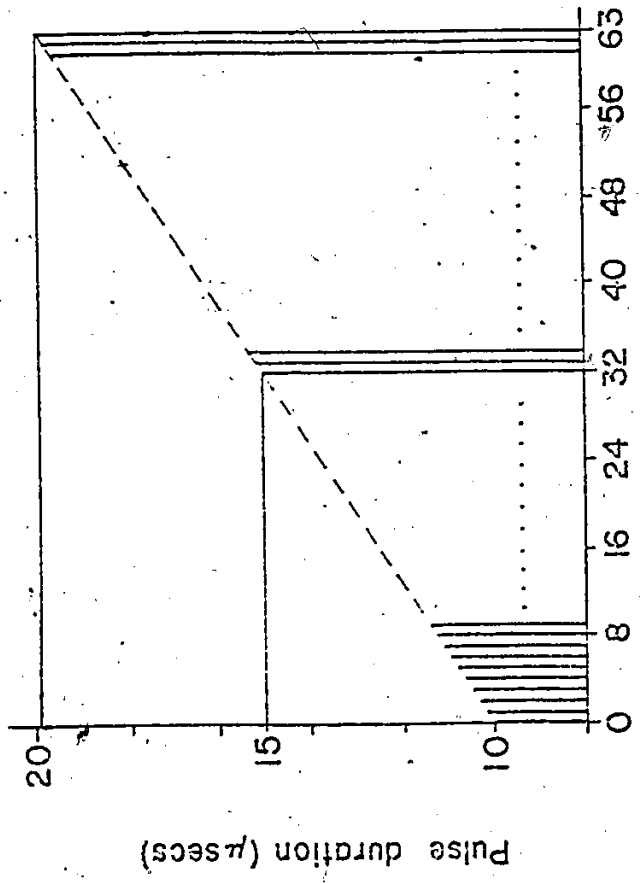


Fig. 4.9 PDM characteristics for a clock frequency of 100 KHz

In this example, $T_S/T = 1677.85$ and since the counters ignore fractions, their contents at the end of the first cycle are (I) = 26, (F) = 13, so that $\psi = 13/64$. The ideal output pulse period is $T_S/64 = 262.16$ μs. Table 4.4 lists the output (A) of the adder and carry CY, the PDM output pulse duration Δ_r, and the times of occurrence t_r of the output pulses, using Eqn. (4.4). The first eleven pulses suffice. The fluctuations of (A) and the occurrence of the carry are evident. The intervals between successive output pulses are constant at 262.03 μs, being low by about 0.05% due to the clock quantization.

Table 4.4 FMM Characteristics

Output Pulse No.	(A)	CY	Δ_r (μs)	t_r (μs)
1	13	0	12.03	262.03
2	26	0	14.06	524.06
3	39	0	16.09	786.09
4	52	0	18.13	1048.13
5	1	1	10.16	1310.16
6	14	0	12.19	1572.19
7	27	0	14.22	1834.22
8	40	0	16.25	2096.25
9	53	0	18.28	2358.28
10	2	1	10.31	2620.31
11	15	0	12.34	2882.34

Maximum Speed

The maximum output frequency of the system is determined by the settling time of the D/A converter and the recovery time of the PDM circuit. The maximum duration of the PDM circuit output pulse is slightly less than 2T, so if this circuit recovers in a time that is less than T, the minimum value of (I) is 3. This assumes that the D/A converter settles within 2T. In the system constructed the settling time of the D/A converter determines the maximum clock frequency. A D/A converter type 7520 with an operational amplifier type 350J was used. A clock frequency of 2 MHz produces a maximum pulse output frequency of 660 KHz. For k = 6, the corresponding maximum input frequency is 10.3 KHz.

The PDM circuit used provides an output pulse whose duration is independent of analog input voltage variations during the triggered state. This is used to advantage by updating the D/A converter during the triggered state. Figure 4.8 shows that the PDM trigger also clocks the accumulator. Carry CY is prevented from changing during early D/A output updating since clocking of the CY latch takes place only when an output pulse is generated.

Input Signal Frequency Range

The minimum count (I) and the number of binary stages in I determine the frequency range covered. Thus if I and

its associated modules in the system are 12 bits long, a frequency range of 1365 to 1 in input signal is accommodated.

Timing Errors

Timing errors between any two consecutive pulses are due to the following:

a) Error due to the finite clock rate, given by Eqn. (4.2). For (I) = 26, as in the example, the maximum error is 0.058% at (F) = 0. The maximum error at (I) = 3 is 0.52%, which compares with the error obtained with the system described in [9] for minimum (I) = 511.

b) The spacings between consecutive output pulses vary because of nonlinearity of the D/A converter and of the PDM characteristic. Let these, combined, give a time error δ in PDM pulse duration. The maximum pulse timing error, normalised with respect to T_o, is $\delta/(I)T$ and is usually negligible compared with the clock quantization error.

4.3.4 Frequency Multiplication System: Circuit Description: Functional Organization

For descriptive purposes, the frequency multiplication system circuitry can be functionally divided into five blocks as follows:

(1) Control circuitry

(2) Fractional counter F and accumulator ACC

(3) Integral counter I

(4) Digitally controlled Pulse duration Modulator

(5) Trigger and load pulse generator.

The circuitry details of each block are described in the following sections.

Control Circuitry

The circuit diagram is shown in Fig. 4.10. It responds to signals from a control panel or from a Micro-computer to which this system could be interfaced and starts the pulse multiplication process. The Control Signals are TTL Compatible.

2.1 START: The negative going edge of the pulse sets flip-flop G5 and enables "Trigger from level crossing of input signal" to Status Counter SC input through NAND gate G6 and inverter G7.

2.2 "Trigger from level crossing of input signal x(t)": is derived from the input signal whose positive going edge marks the beginning of a cycle of the input signal x(t). This signal, gated by the output of flip-flop G5 activates counter SC.

Control circuitry also incorporates a data multi-plexer (MPXR). The counter SC is 2 bits long and is cleared initially so that its two outputs PHASE1 and PHASE2 are both

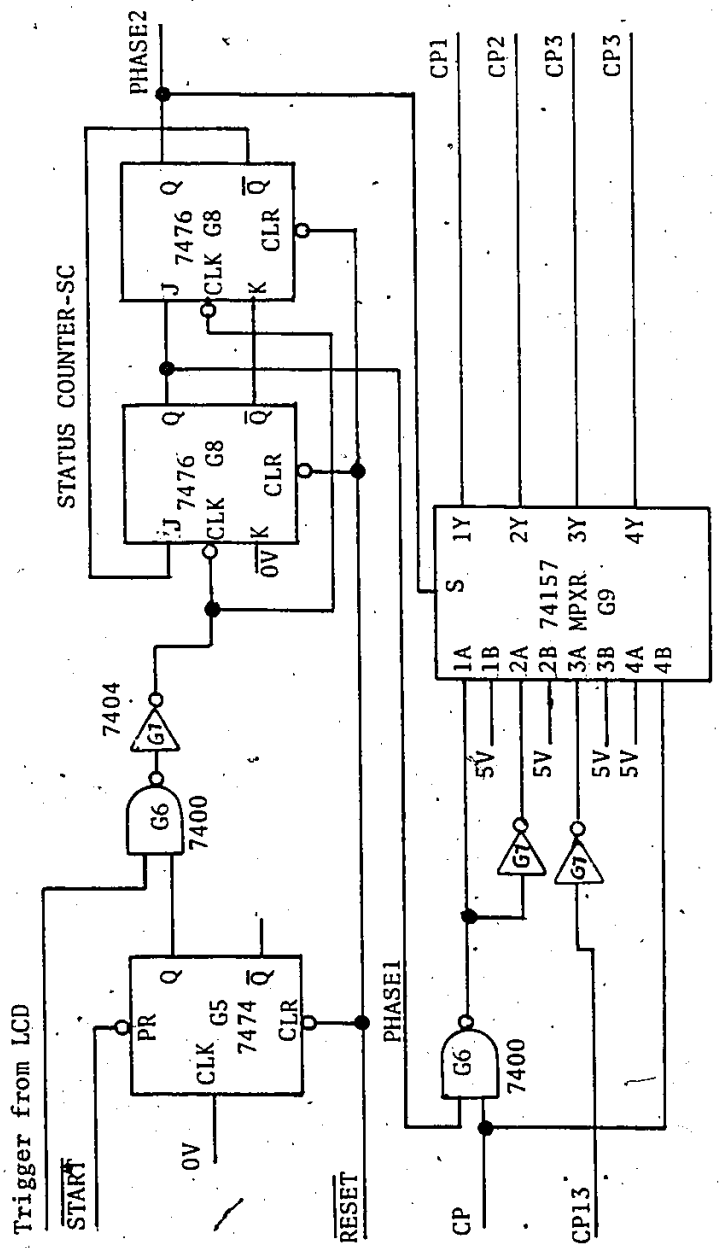


Fig. 4.10 Control circuitry of FMM

LO. The first trigger from level crossing of input x(t) changes PHASE1 output to HI and marks the first phase of frequency multiplication process i.e. determination of the period of input signal x(t). The second trigger changes PHASE2 to HI and (PHASE1)(PHASE2) represents the second phase of the multiplication process.

The PHASE2 signal is the selector input to MPXR consisting of TTL quadruple 2 to 1 Data Selector 74157 (G9); it represents switches S_a and S_b shown in the block diagram Fig. 4.8. A LO at input S corresponds to position 1 of S_a and S_b in Fig. 4.8. MPXR in this position gates A inputs to Y outputs. HI on S corresponds to position 2 of S_a and S_b; MPXR gates B inputs to Y outputs in the position 2. The four pairs of input signals are:

a) system clock CP gated by PHASE1 signal is applied to 1A input; 1B input is held HI. 1Y output CP1 forms clock pulses to counter forming least two significant bits of fractional counter F.

b) system clock CP gated by PHASE1 signal is applied to 2A input; 2B input is held HI. 2A input is logical complement of 1A input. The corresponding 2Y output CP2 sends clock pulses to counter forming four most significant bits of the fractional counter F.

c) The carry output of the fractional counter F followed by inversion is applied to 3A input; 3B input is

held HI. The corresponding 3Y output CP3 delivers clock pulses to integral counter I in up-count mode.

d) 4A input is held HI; system clock CP is connected to 4B input. The corresponding 4Y output supplies clock pulses to the integral counter in down-count mode.

Fractional Counter F and Accumulator ACC:

The circuit diagram is shown in Fig. 4.11. Counter F is 6 bits long and consists of two cascaded binary counters 7476 and 74161. Carry CP13, generated when the 74161 counter overflows, drives the integral counter I in the up-count mode.

Accumulator ACC consists of a 6 bit TTL latch 74174 and serves as a register to store the output of 6 bit adder A. The accumulator is loaded with the output of adder A by a trigger pulse as explained later. The digital output of the adder (SB0-SB5) is the sum of the contents of the fractional counter (F) and the contents of ACC. Adder A consists of a 7482 and a 7483 in cascade.

Integral Counter I

The circuit diagram of an 8 bit integral counter I is shown in Fig. 4.12; additional cascaded stages may be added if desired. Countup pulses to I are the carry output pulses of the fractional counter F and count-down pulses are the clock pulses CP. The contents (I) of the counter I at the beginning of second period of input signal x(t) are saved in

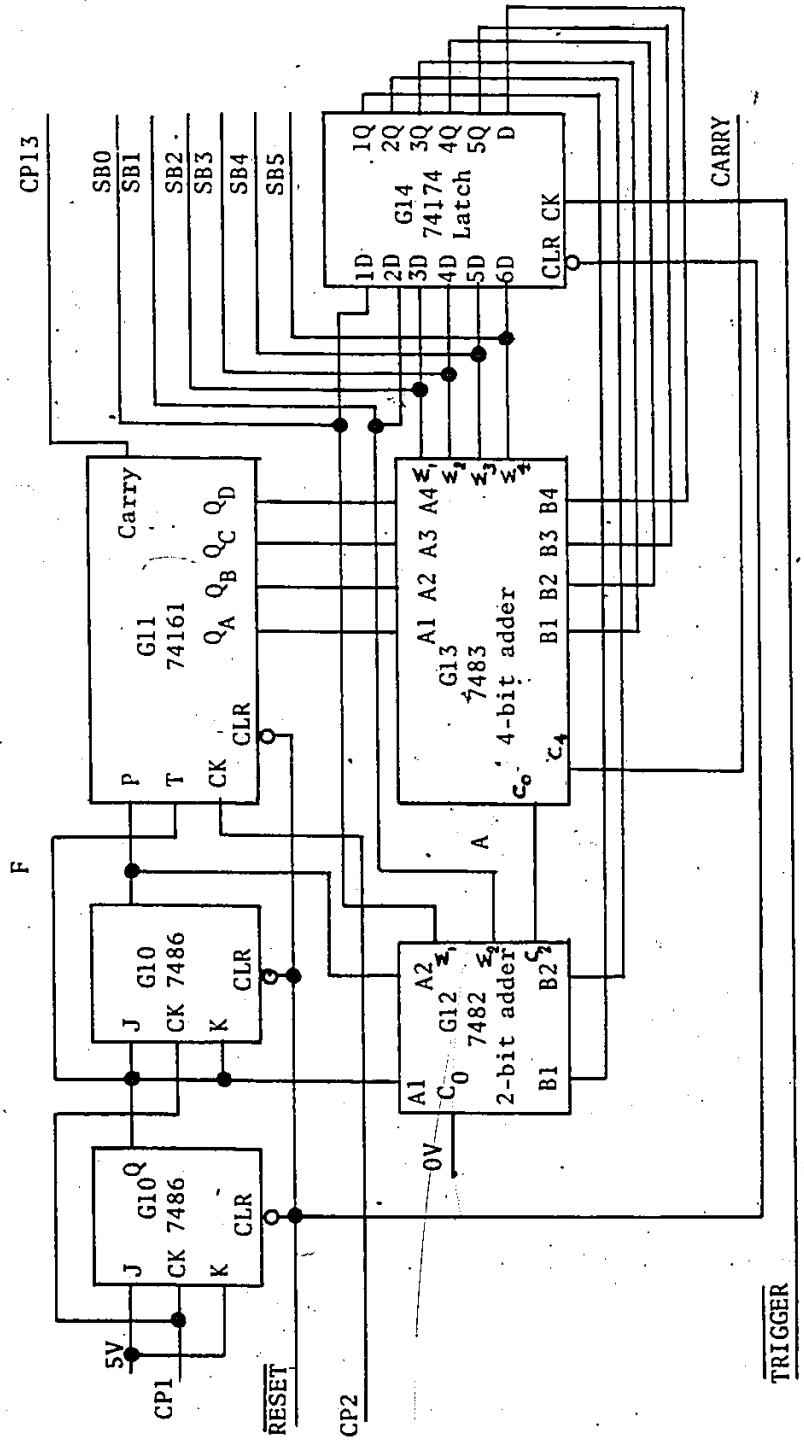


Fig. 4.11 Fractional counter and accumulator

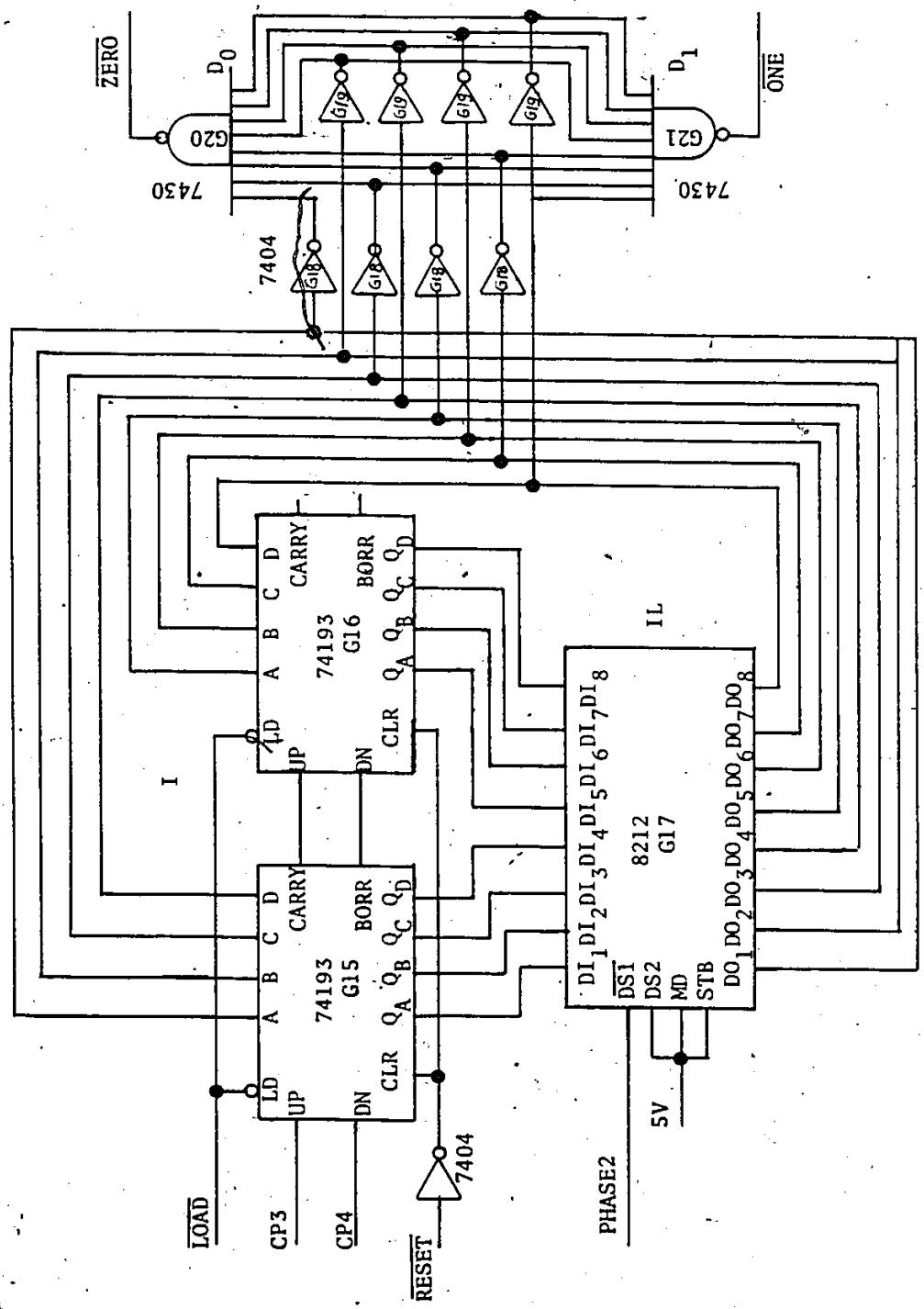


Fig. 4.12 Integral counter

an 8212 8-bit latch IL. PHASE2 signal which marks the beginning of the second period is used as clock to latch (I) into IL.

The 00000000 and 00000001 states of the counter I are decoded as negative going signals designated as $\overline{\text{ZERO}}$ and $\overline{\text{ONE}}$ respectively by using Hex inverters 7404 and 8 input NAND gates 7430. These signals are combined with the carry of adder A to generate the trigger pulse to PDM and the load pulse to the counter I.

Digitally Controlled Pulse Duration Modulator (PDM):

The PDM is required to deliver output pulses of duration varying from 1 to 2 clock periods. The pulse duration is linearly related to input voltage. Referring to Fig. 4.13, the PDM consists of D/A converter followed by analog pulse duration modulator.

D/A Converter

A type AD 7520 10 bit multiplying type converter with current output is used. The reference voltage is -3.7 volts derived from -5 volts by zener diode D1 and resistor R1. The current output of the D/A is converted to voltage by a BB3507J summing operational amplifier AMP [66]. A dc offset voltage from potentiometer P1 is also applied to the summing amplifier through a 10K input resistor R2. The summing amplifier output is applied to the input of the analog pulse duration modulator.

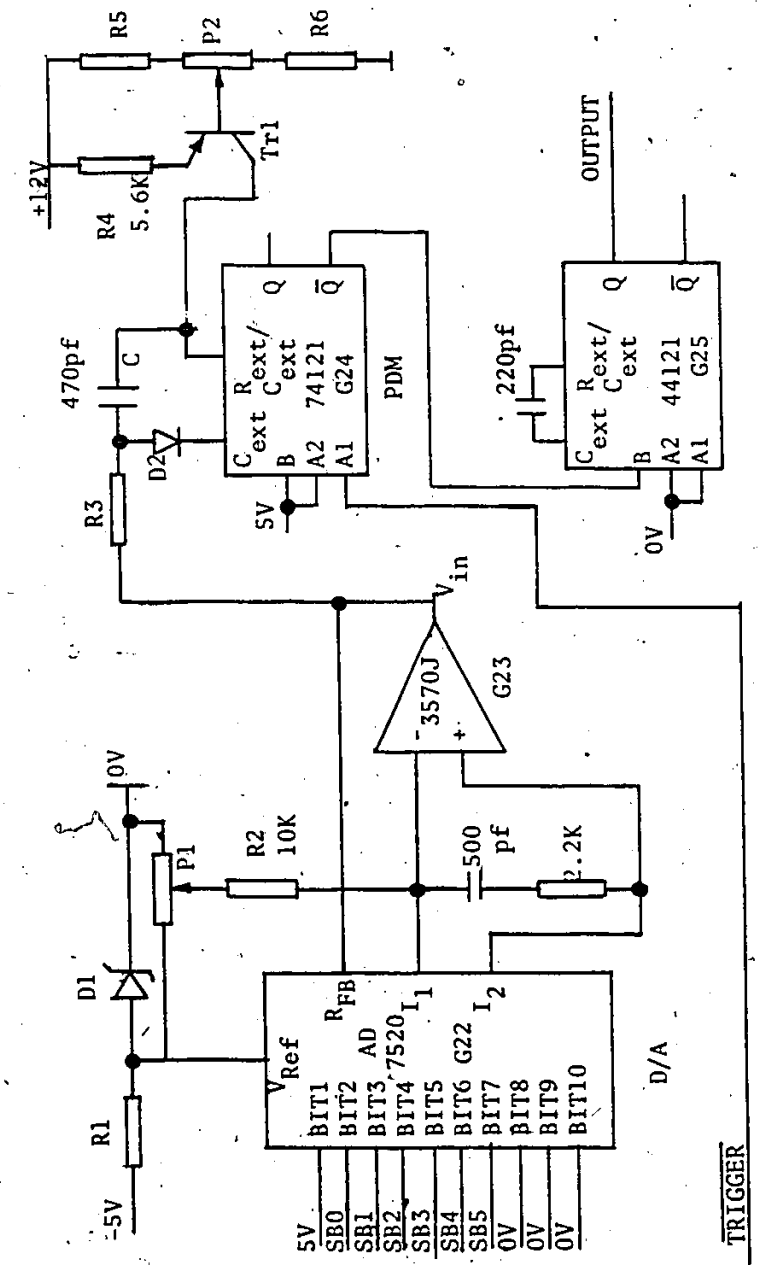


Fig. 4.13 Digitally controlled Pulse Duration Modulator

Analog Pulse Duration Modulator

The pulse duration modulator uses a TTL monostable multivibrator 74121. Its operation is explained by referring to Fig. 4.13. In the quiescent state the output at C_{ext} is HI (typically 4.8V). Diode D2 is reverse biased, provided that V_{in} is less than the voltage at the C_{ext} pin. Timing capacitor C is charged to a value $(V_{in} - V_{off})$ where V_{off} is the voltage at the R_{ext}/C_{ext} pin in the quiescent state (typically 0.7 V). When triggered by an input pulse, the voltage V_d at C_{ext} goes LO (0.7 V) and the voltage at R_{ext}/C_{ext} goes negative with respect to ground. Capacitor C charges linearly, the constant charging current I being adjustable by means of potentiometer P2. When the voltage at R_{ext}/C_{ext} reaches the threshold value V_{th} (typically 0.7 V), the monostable reverts back to the quiescent state. The pulse duration T_D is given by

$$IT_D = CV_{in} - C\,(V_{off} + V_d - V_{th}) \qquad (4.6)$$

The second term is cancelled by offsetting V_{in} in the quiescent state.

Once the PDM circuit is triggered, the duration of the output pulse is not affected by a change in V_{in}. This characteristic of the circuit is used to advantage in providing an early updating of the D/A converter output, which now settles while the PDM is in the triggered state.

The measured voltage vs pulse duration characteris-

tics obtained for the PDM with C = 470 pf, R3 = 500 ohms and I = 0.760 MA is shown in Fig. 4.14.

Triggering and Load Control

As explained previously, the PDM is activated during each output period at one of the following instants,

(a) CY = 0 and (I) = 1

(b) CY = 1 and (I) = 0

The trigger pulses commence at the second period of the input signal x(t). This is realized by using combinational logic NAND and NOR gates as shown in Fig. 4.15.

The load pulse to the integral counter I lags behind trigger pulse by one clock period T. Once the PDM circuit is triggered, its output pulse is not affected by a change in its analog input voltage. This permits early updating of the D/A converter by the leading edge of the PDM trigger pulse. Referring to Fig. 4.9, it is also used to update the accumulator ACC, rather than by using the output pulse. Early updating will, on occasion, affect the carry from adder A. A D-flip-flop is therefore included; it is clocked by the output pulse, and provides the trigger control signals CY and $\overline{CY}$. This makes the width of the trigger pulse to be T. The load pulse to counter I is derived by triggering a monostable multivibrator from the trailing edge of the trigger pulse. Separate decoding circuits for the triggering and load pulse are avoided.

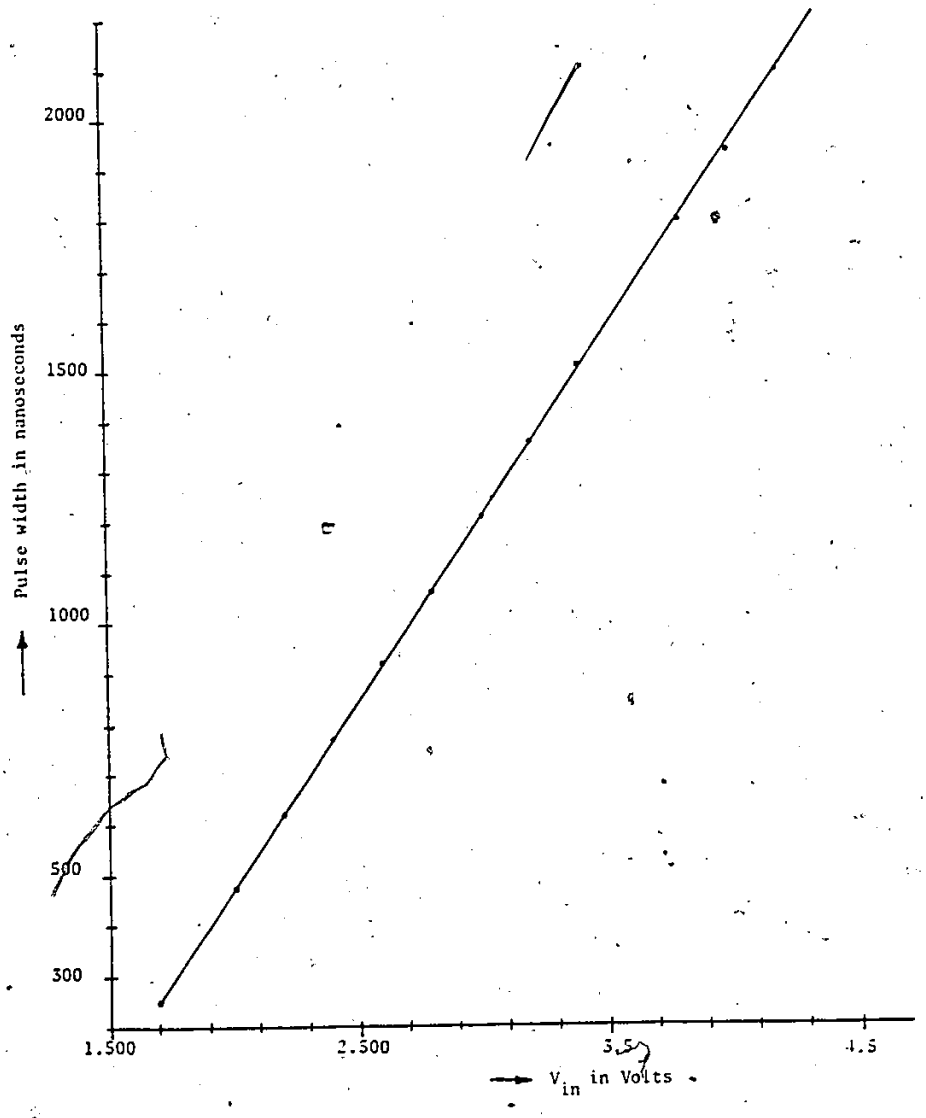


Fig.4.14 Pulse width vs control voltage characteristics

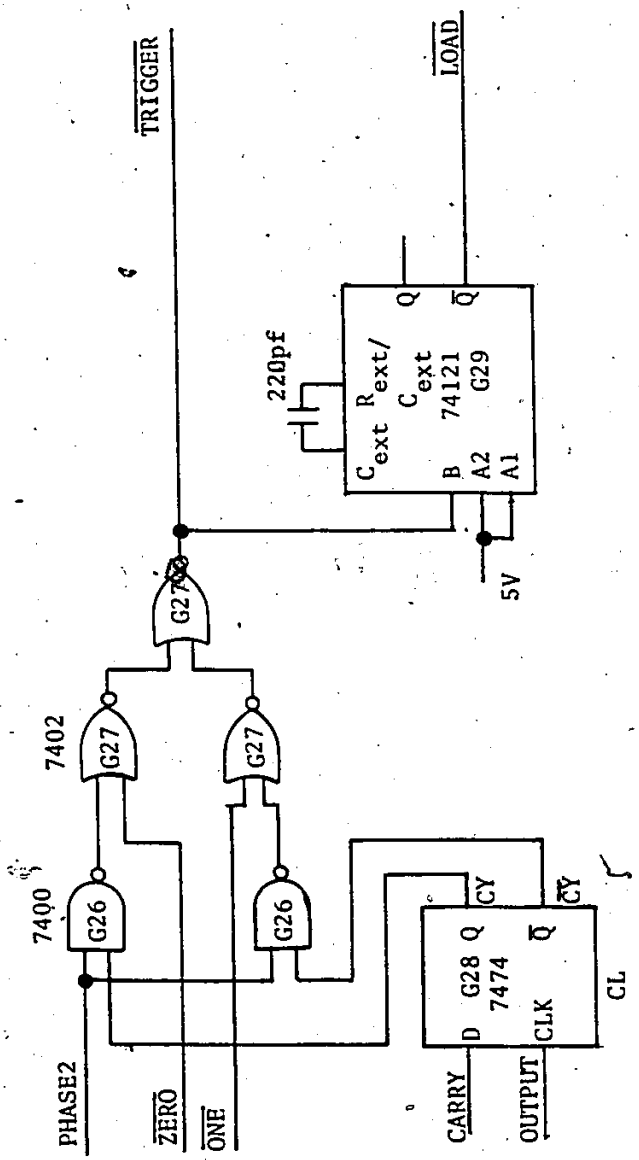


Fig. 4.15 Triggering and load control

4.3.5 A/D Converter

The analog input signal x(t) is quantized by A/D converter ADC 82AG (successive approximation type) in 2.8 μsec. It accepts bipolar signals within ±5 volts range and provides an 8 bit digital output in complementary offset binary/two's complement representation. The circuit diagram is shown in Fig. 4.16(a).

The output pulses of FMM form convert commands. A conversion begins with the negative going of a command pulse. Simultaneously the status output ST goes HI. Completion of a conversion is marked by ST returning to LO; a write signal is generated at this instant to write the output of A/D in the System 80/10 memory. The timing diagram in Fig. 4.16(b) illustrates the above sequence of events. The output of A/D is interfaced to the System 80/10 data bus through an eight-bit bipolar 8212 input/output port. The mode of operation is determined by the "DENB" signal as seen below, which is derived by combining $\overline{\text{DACK1}}$ and A/DR/ of DMA controller.

DENB	STB	DS1.DS2	MD	output
1	0	0	0	tri-state
0	0	1	1	DATA in

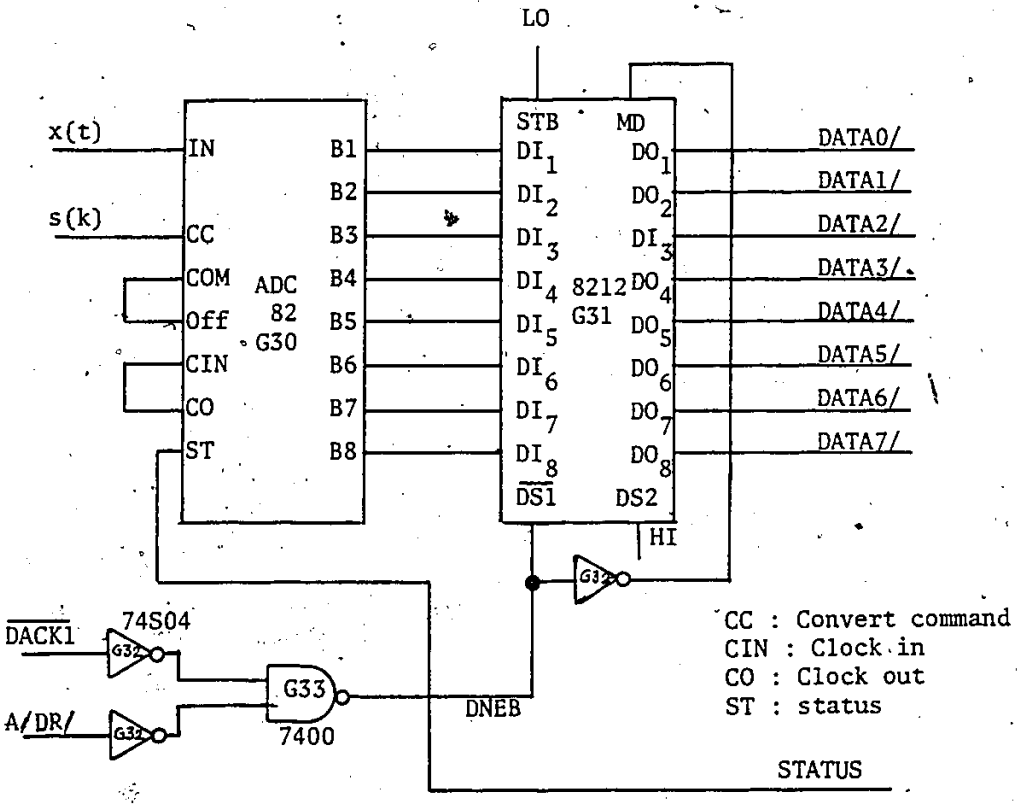


Fig. 4.16(a) Circuit diagram of A/D converter

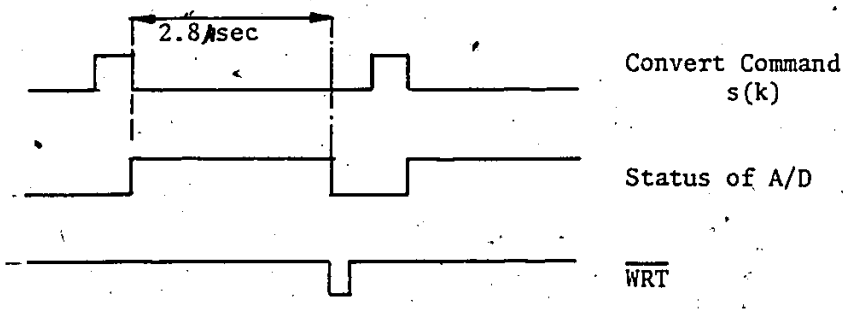


Fig. 4.16(b) A/D converter Timing diagram

The ±15V supplies required for A/D are derived from +5V using a type 546 Dual Power supply module [67].

4.3.6 DMA

The DMA function is to generate, upon request, memory addresses for the A/D data to be deposited directly in the System 80/10 RAM. The Programmable DMA controller 8257 shown in Fig. 4.17(a) has four channels [68]. It operates in three modes; (1) DMA read, which causes data to be transferred from memory to a peripheral, (2) DMA write, which causes data to be transferred from a peripheral to memory, and (3) DMA verify, which does not actually involve the transfer of data. In the instrumentation built for WSA only channel 1 in mode 2 is used. Each channel includes two sixteen-bit registers; (1) DMA address register, (2) Terminal count register. The former is loaded with the address of the first memory location to be accessed (memory location 0400 Hexadecimal is used as the starting address). The value loaded into the low-order 14 bits of the terminal count register specifies the number of DMA cycles minus one before terminal count TC output is activated. The two most significant bits of the terminal count register specify the type of DMA operation as follows:

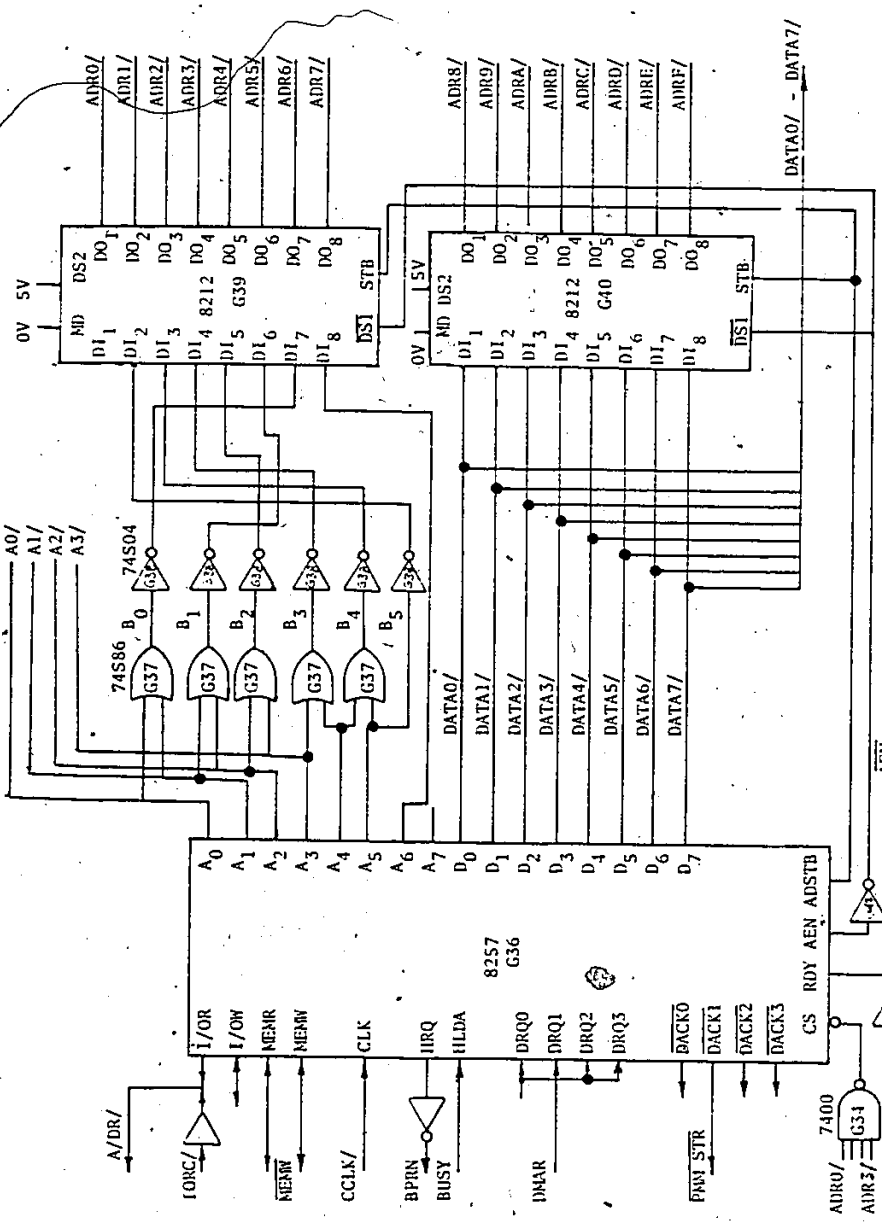


Fig. 4.17(a) DMA controller circuit diagram

Bit 15	Bit 14	Type of DMA operation
0	0	verify DMA cycle
0	1	write DMA cycle
1	0	read DMA cycle
1	1	illegal

The 8257 registers are programmed by the system 80/10 using I/O instructions, and are configured as I/O devices with the four MSBs of address data as 0000 (1111 on the system bus); these four bits are decoded by NAND gate G34 to provide a chip select (CS) signal to the 8257 as shown in Fig. 4.17(a). The DMA request DRQ1 is generated through software via the output port EB. Upon receiving a DMA transfer request, the 8257

(1) acquires control of system 80/10 bus through hold request (HRQ) output. HRQ, after inversion, is connected to BPRN/ line of the system 80/10 Bus.

(2) acknowledges the DMA request and DACK1 output goes LO (DMA request on channel 1 is assumed). The first negative going transition of DACK1 is used to start the frequency multiplication process. It is also used to enable the A/D output to the system bus.

(3) outputs the least significant eight bits of memory address on its output lines A_0 to A_7. These outputs are operated by the EXCLUSIVE-OR gates G37 to modify the address bits according to

$$B_i = A_i + A_{i+1} \qquad i = 0, 1, \ldots, 4$$
$$B_5 = A_5 \qquad (4.6)$$

These are connected to 8212 I/O port G39 in bit reversed order. This is exactly the operation involved in implementing Eqn. (3.45) in hardware. The inverters (G38) are used to make the address outputs compatible with the system 80/10 address bus. Note that bit DI_1 is held permanently at logic 0 and six output bits B_0 - B_5 are used to derive ADR1/-ADR6/. This means that alternative locations are skipped in depositing the A/D data in memory. The reason for doing this is to reformat the data as a 16 bit word through software, since computations are performed on 16-bit data. In the case of A/D with more than 8 bits, bit DI_1 of 8212 may be used to point to the next location to deposit the MSB_s of data, by complementing it before the 8257 issues the next DMA cycle. The most significant 8 bits are latched at I/O port 8212 by AEN and ADSTB signals. The sequence of events after a DMA request in DMA write mode is illustrated in Fig. 4.18. Note that A/DR/ output of 8257 enabling the A/D output to the data bus and write pulse MEMW are generated with a delay of one and two clock cycles from ADSTB pulse respectively. It is possible that a A/D conversion might not have been completed within this time. To avoid the transfer of incorrect data, memory write pulse MWTC/ is generated using the monostable multivibrator G44

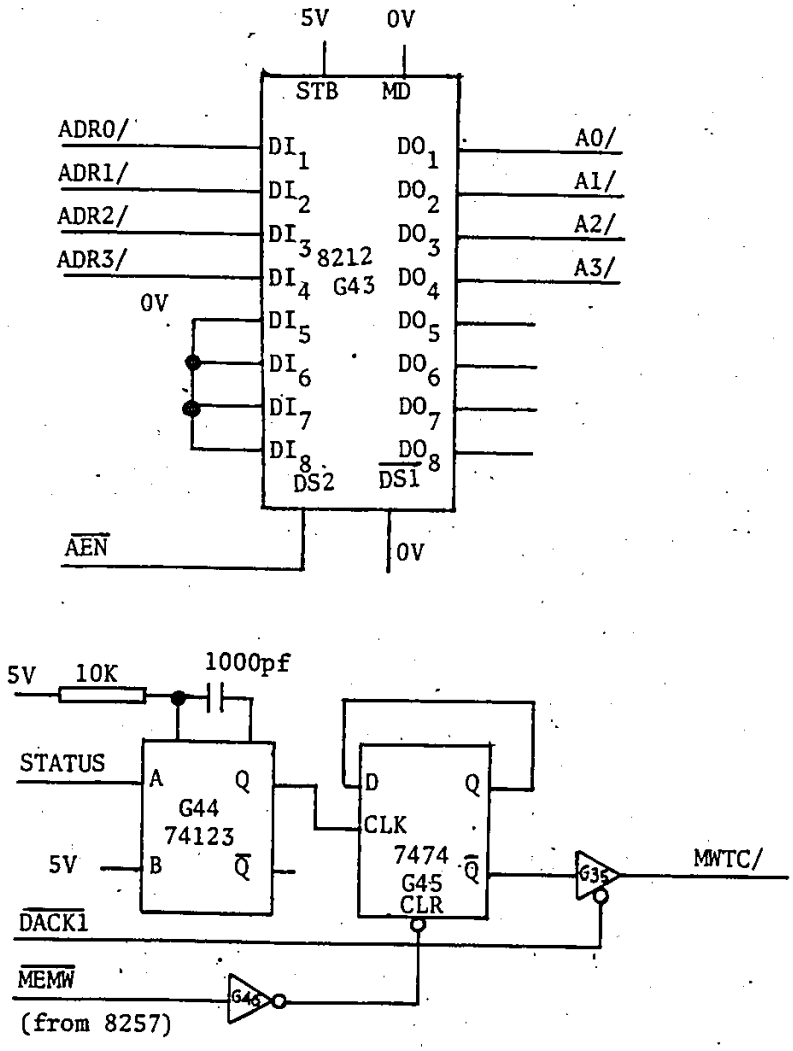


Fig. 4.17(b) Address data interface between 8257 and System 80/10 in programming mode

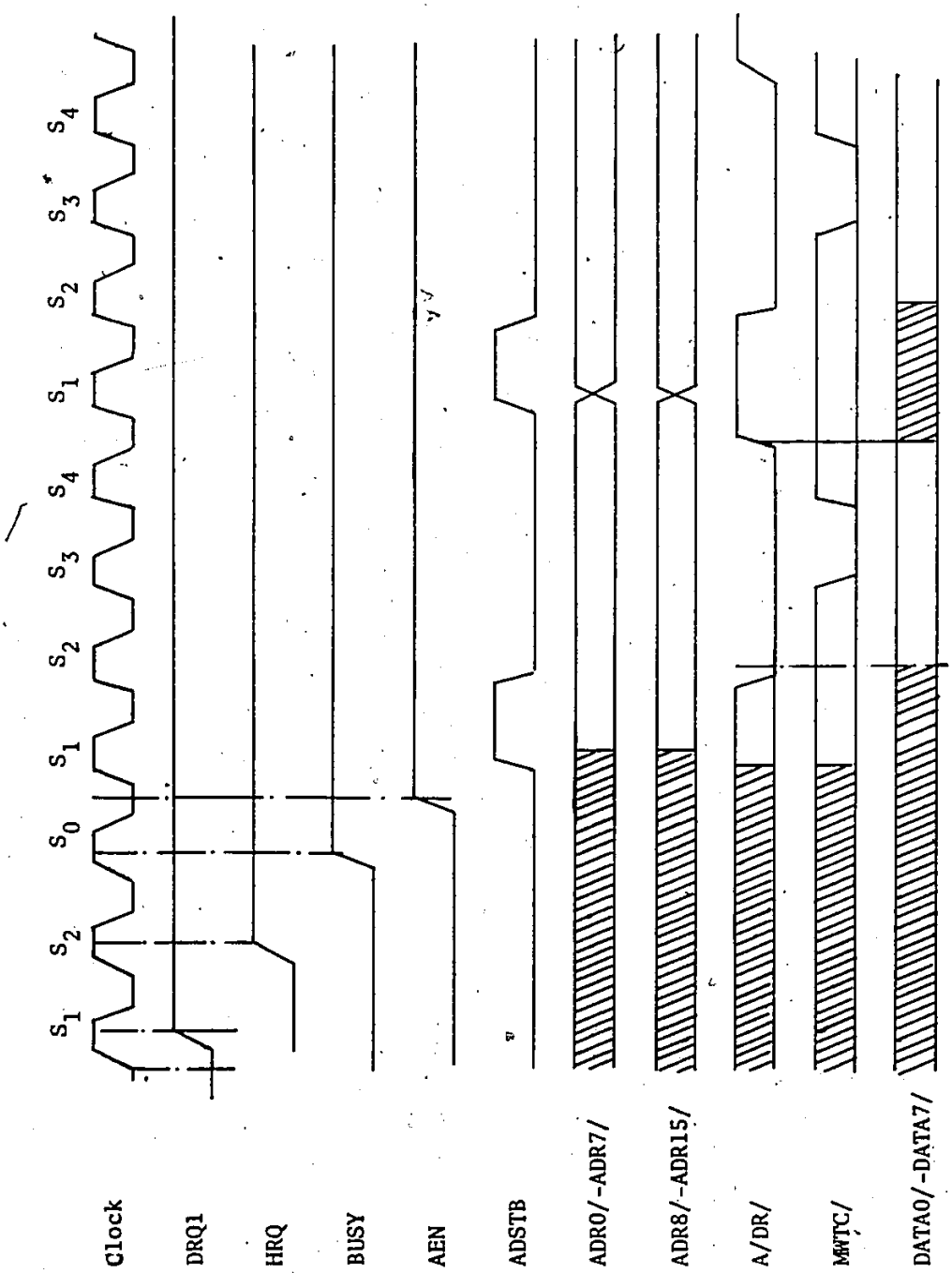


Fig. 4.18 Timing diagram of DMA operation

and flip-flop G45 triggered by the negative going edge of STATUS output of A/D as shown in Fig. 4.17(b). This prolongs the DMA write cycle by not returning an active READY signal by the system 80/10 memory.

During the programming mode, the 4 LSB address bits are transmitted from the system 80/10 bus to the 8257 A_0-A_3 inputs through 8212 I/O Port (G43) as shown in Fig. 4.17(b). I/O Ports G39 and G40 are in high impedence state while G43 is enabled during the programming mode of the 8257.

4.3.7 Control

The use of off-the-shelf general purpose peripheral ICs, tri-state I/O ports 8212, and programmable DMA controller 8257 results in a compact control structure. The control reduces to that of initiating and terminating a DMA request. This is implemented using a 7474 flip-flop (G48) as shown in Fig. 4.19. It is set by a $\overline{\text{BEGIN}}$ signal generated through software by the system 80/10. At the completion of required number of DMA cycles (64 samples in the system developed), TC output from the 8257 goes HI and resets G45, thereby terminating further DMA requests.

A provision is made through a two position switch to use either the input signal to be analysed or external marker pulses for frequency multiplication purposes. The latter mode permits the sequency analysis of non-periodic

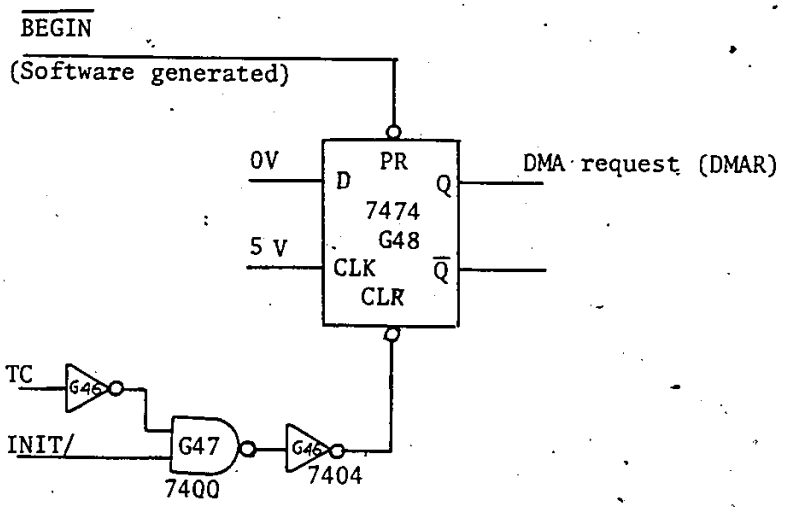


Fig. 4.19 Control circuit

signals by computing the sequency coefficients of a signal defined within the period set by two marker pulses.

4.3.8 Output Display

Attention to the display of final measured quantities is important for the effective use of any instrument. In the case of Walsh spectral measurements, the coefficient values and a plot of amplitude vs sequency are desired. In Fourier analysis, the outputs of interest are numerical values of Fourier components, a plot of Magnitude vs Frequency and Phase vs Frequency. These capabilities are added by interfacing to an intelligent BASIC graphic terminal 2647A. (The cost of a terminal of this capability is much higher than the instrument itself.) The TERMINAL BASIC contains most of the standard BASIC statements together with statements that will allow a program to monitor and control terminal operations. This allows a program running in the 2647A to interact with peripheral devices (the Walsh Spectral Analyser in the present case). Detailed operation of the terminal is given in references [59,69].

CHAPTER 5

ON WALSH TO FOURIER CONVERSION

5.1 Introduction

Walsh-to-Fourier Conversion was studied by Blachman [70,71], Abramson [72], Siemens and Kitai [22,73] and recently by Tadokoro and Higuchi [23]. Signals can be classified into four spectral categories;

1. infinite Walsh series with infinite Fourier Series
2. finite Walsh series with finite Fourier Series
3. finite Walsh series with infinite Fourier Series
4. infinite Walsh series with finite Fourier Series

The last category is of particular interest. It has been shown that for a bandlimited signal with the highest normalised frequency component (harmonic) F, all the F Fourier components can be determined from a finite number of Walsh coefficients with the sequency up to S without error, provided $S \geq F$ [22,73,74,75,76]. The Discrete Fourier Transform of a given sampled data sequence is widely employed to compute the Fourier coefficients of a signal. The DFT is implemented on general or special purpose computers using FFT techniques. Given N data samples, one

can obtain N Walsh coefficients using Fast Walsh transform algorithms as discussed earlier and N Fourier coefficients using FFT algorithms. Tadokoro and Higuchi [23] have formulated an algorithm to obtain N discrete Fourier coefficients from N discrete Walsh coefficients (a coefficient obtained using sampled values is referred to as a discrete coefficient) and it differs from the method used by Siemens and Kitai. Either way, obtaining the Fourier coefficients through the Walsh coefficients reduces the number of multiplications when compared to the Cooley-Tukey FFT algorithm for a data length up to 64. The fast nature of this process was pursued by Horton for digital impedence relaying of power lines [77]. Also when the desired number L of Fourier components is small compared to the input data length N, this conversion process is faster. (This feature is benificial in narrow band signal analysis.) A comparison of the Walsh to Fourier transformation process with the FFT method with respect to the number of multiplications, storage requirements and the finite word length effects is now considered.

5.2 Walsh to Fourier Conversion Process due to Siemens and Kitai [22]

This process is illustrated in Fig. 5.1. Using the symmetry properties of the sine/cosine functions and cal/sal

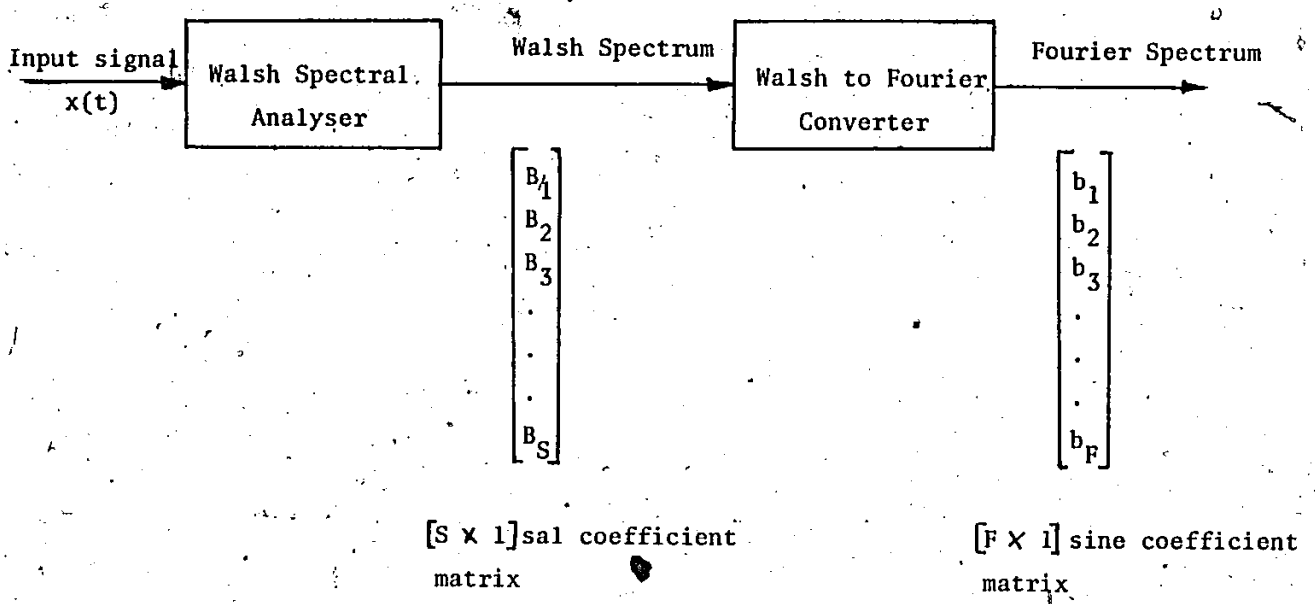


Fig. 5.1 Walsh to Fourier Transformation Process

functions, it can be shown that a_k terms of the Fourier expansion are functions of only A_s terms of the corresponding Walsh expansions. Similarly b_k terms depend only on B_s terms.

The conversion process to obtain b_k terms from B_s terms is discussed below. A similar method applies to a_k terms. Equating the odd parts of the Fourier and Walsh series representation (Eqns. (2.10) and 2.13)) we get,

$$\sum_{k=1}^{\infty} b_k \sin 2\pi kt = \sum_{s=1}^{\infty} B_s \, sal\,(s,t) \qquad (5.1)$$

T is taken as unity henceforth. Multiplying both sides by sin $2\pi ft$ and integrating over unit interval, we get

$$b_f = \sum_{s=1}^{\infty} b_{f,s} B_s; \; f = \{1, 2, \ldots, \infty\} \qquad (5.2)$$

where $b_{f,s}$ is the fth sine coefficient of the sth sal function and is given by

$$b_{f,s} = 2 \int_0^1 \sin 2\pi ft \; sal(s,t) \; dt$$

similary a_f terms are given by

$$a_f = \sum_{s=1}^{\infty} a_{f,s} A_s; \; f = \{1, 2, \ldots, \infty\} \qquad (5.3)$$

where $a_{f,s}$ is the fth cosine coefficient of the sth cal

function. In matrix form, Eqn. (5.2) can be written as,

$$[\underline{b}] = [F_b]\ [\underline{B}] \tag{5.4}$$

where $[\underline{b}]$ is the sine coefficient matrix

$[F_b]$ is the conversion matrix

$[\underline{B}]$ is the Walsh (sal) coefficient matrix.

For a band limited signal $x_b(t)$

$$x_b(t) = \sum_{k=1}^{F} b_k \sin 2\pi kt + \sum_{k=0}^{F} a_k \cos 2\pi kt \tag{5.5}$$

where F is the highest frequency component of the input signal.

The sth sal coefficent of x_b is then,

$$B_s = \int_0^1 x_b(t)\ \mathrm{sal}(s,t)\ dt \tag{5.6}$$

$$= \int_0^1 \left[\sum_{k=1}^{F} b_k \sin 2\pi kt\right] \mathrm{sal}(s,t)\ dt \tag{5.7}$$

$$= \sum_{k=1}^{F} d_{s,k}\ b_k \tag{5.8}$$

where $d_{s,k}$ is the sth sal coefficient of sin $2\pi kt$. In matrix form, Eqn. (5.8) can be written as,

$$[\underline{B}] = [D]\ [\underline{b}] \tag{5.9}$$

where $[\underline{B}]$ = (Sx1) sal coefficient matrix

(S = a finite number)

[D] = (SxF) conversion matrix

$[\underline{b}]$ = (Fx1) sine coefficient matrix.

One can solve for the set $\{b_f\}_{f=1}^{F}$ by a system of F linearly independent equations for $S \geq F$. Since

$$d_{s,k} = \frac{1}{2}\,[2 \int_0^1 \sin\,(2\pi kt)\ \mathrm{sal}(s,t)\ dt] = \frac{1}{2}\, b_{k,s}$$

$$[D] = \frac{1}{2}\,[F_b]^T \tag{5.10}$$

From Eqn. (5.9), the sine coefficient matrix $\underline{b}$ is given by

$$[\underline{b}] = [K^{-1}]\ [F_b]\ [\underline{B}] \tag{5.11}$$

where

$$[K] = \frac{1}{2}\,[F_b]\ [F_b]^T$$

Using the orthogonal property of Walsh functions, Siemens proved that $[K]^{-1}$ is a non-singular diagonal matrix [73] whose elements are given by

$$k_{f,f} = \mathrm{sinc}^{-2}\,(\frac{f}{2^m});\ f = \{1,\ 2,\ \ldots,\ 2^{m-1} - 1\}$$

$$= \frac{\mathrm{sinc}^{-2}}{2}\ (\quad = 2^{m-1} = F \tag{5.12}$$

As $N = 2^m \to \infty$, $k_{f,f}$ approaches unity. The rows of the matrix F_b represent the Walsh (sal) expansions of the corresponding sine functions. For example, the third row of F_b gives the Walsh expansion of sin $2\pi(3t)$:

$$\sin 6\pi t = \frac{4}{3\pi}\ \mathrm{sal}(1,t) + 1.025\ \mathrm{sal}(3,t) + \ldots \tag{5.13}$$

Similarly the columns of the matrix F_b represent the Fourier expansions of the sal functions $\{sal(s,t)\}_{s=1,2,\ldots,F}$. For example, the first column of F_b represents the Fourier coefficients of sal(1,t):

$$sal(1,t) = \frac{4}{\pi} \sin 2\pi t - \frac{4}{3\pi} \sin 2\pi(3t) + \ldots \qquad (5.14)$$

The elements $a_{f,s}$, $b_{f,s}$ can be computed using a non-recursive equation for the Fourier transform of a Walsh function [78]. The pattern of non-zero unique elements in F_b is important in the Walsh-Fourier conversion process and is discussed below. If s is odd, the sal function sal(s,t) have non-zero Fourier coefficients for odd-numbered harmonics only; that is $b_{1,s}$, $b_{3,s}$, ..., $b_{2k-1,s}$ only are non-zero. sal(2s,t) can be considered as a wave sal(s,t) with a time base that has been halved. Consequently sal(2s,t) has non-zero coefficients whose harmonics numbers are twice those of sal(s,t); that is $b_{2,2s}$, $b_{6,2s}$ ... $b_{2(2k-1),2s}$ are non-zero elements. In general, the non-zero coefficients of $sal(2^k s,t)$ are $b_{2^k,2^k s}$, $b_{3\cdot 2^k,2^k s}$, ..., $b_{2^k(2\ell-1),2^k s}$, where s is odd. In view of this property, the matrix F_b of dimension $2^{m-1} \times 2^{m-1}$, has 2^{m-2} independent columns elements; any other column of elements is a subset of some odd-numbered column elements.

The F_b matrix for F = 8 is given below

f \ s	1	2	3	4	5	6	7	8
1	1.27324	-	-0.52739	-	-0.10490	-	-0.2532	-
2	-	1.21324	-	-	-	-0.52739	-	-
3	0.42441	-	1.024624	-	-0.68463	-	.283584	-
4	-	-	-	1.27324	-	-	-	-
5	0.25464	-	0.614774	-	0.920075	-	-.38110	-
6	-	0.424413	-	-	-	1.02462	-	-
7	0.18189	-	-0.07534	-	0.37876	-	0.9144	-
8	-	-	-	-	-	-	-	1.27324

- denotes zero elements

5.3 Walsh to Fourier Conversion Method of Tadokoro and Higuchi [23]

The method described above starts with band limited signals. The Walsh and Fourier coefficients used are the analog inner products of the input signal with the basis functions according to Eqn. (2.11, 2.12, 2.14, 2.15).

The method due to Tadokoro and Higuchi can be considered as a discrete version of that due to Siemens and Kitai. Consider a data sequence X(0), ..., X(N-1) of length N obtained by uniform sampling of an analog signal. The discrete Fourier transform of the data sequence is given by

$$F(k) = \frac{1}{N} \sum_{i=0}^{N-1} X(i)\, w^{ik}, \quad k = 0, 1, \ldots, N-1 \qquad (5.15)$$

where

$$w = e^{-j2\pi/N}$$

Given the N DFT coefficients, the corresponding data sequence X(0) ... X(N-1) is obtianed by the inverse transform

$$X(i) = \sum_{k=0}^{N-1} F(k)\, w^{-ik}, \quad i = 0, 1, \ldots, N-1 \tag{5.16}$$

If the data sequence is real, then $F(k)$ satisfies the following symmetry conditions [24]

$$\mathrm{Re}[F(k)] = \hat{a}_k = \mathrm{Re}[F(N-k)] \tag{5.17a}$$

$$\mathrm{Im}[F(k)] = \hat{b}_k = -\mathrm{Im}[F(N-k)] \tag{5.17b}$$

(The notation $\hat{(\cdot)}$ is used to differentiate the coefficients of the DFT from those obtained as the analog inner product of the input signal and the basis functions used by Siemens and Kitai.)

Using the symmetry relations of Eqns. (5.17a) and (5.17b), the data sequence can be represented by,

$$X(i) = \sum_{k=0}^{(N/2)-1} 2\hat{a}_k \cos\frac{2\pi ki}{N} + \sum_{k=1}^{N/2} 2\hat{b}_k \sin\frac{2\pi ki}{N} \tag{5.18}$$

The discrete Walsh transform of the given data sequence is given by

$$B(k) = \frac{1}{N}\sum_{i=0}^{N-1} X(i)\, \mathrm{wal}(k,i), \quad k = 0, 1, \ldots, N-1 \tag{5.19}$$

and the inverse transform is given by

$$X(i) = \sum_{k=0}^{N-1} B(k)\, \mathrm{wal}(k,i), \quad i = 0, 1, \ldots, N-1 \tag{5.20}$$

In terms of cal and sal functions, Eqn. (5.20) can be written as

$$X(i) = \sum_{s=0}^{N/2-1} A_s \, cal(s,i) + \sum_{s=1}^{N/2} B_s \, sal(s,i) \qquad (5.21)$$

The Fourier component $\hat{b}_k$ is computed according to

$$\hat{b}_k = \frac{1}{N} \sum_{i=0}^{N-1} X(i) \sin \frac{2\pi ki}{N} \qquad (5.22)$$

Substituting for X(i) from Eqn. (5.21) in Eqn. (5.22) we get

$$\hat{b}_k = \frac{1}{N} \sum_{i=0}^{N-1} \left[\sum_{s=0}^{N/2-1} A_s \, cal(s,i) + \sum_{s=1}^{N/2} B_s \, sal(s,i) \right] \sin \frac{2\pi ki}{N} \qquad (5.23)$$

Interchanging the order of summation gives

$$\hat{b}_k = \sum_{s=0}^{N/2-1} A_s \frac{1}{N} \sum_{i=0}^{N-1} cal(s,i) \sin \frac{2\pi ki}{N} + \sum_{s=1}^{N/2} B_s \frac{1}{N} \sum_{i=0}^{N-1} sal(s,i) \sin \frac{2\pi ki}{N} \qquad (5.24)$$

The sum $\frac{1}{N} \sum_{i=0}^{N-1} cal(s,i) \sin \frac{2\pi ki}{N}$ is the sth cal coefficient of sin $2\pi ik/N$. Similarly

$\frac{1}{N} \sum_{i=0}^{N-1} sal(s,i) \sin \frac{2\pi ki}{N}$ is the sth sal coefficient of sin $2\pi ik/N$. In the case of analog integration, due to the odd and even symmetry of sal/cal functions, we have the following relations:

$$\int_0^1 \text{cal}(s,t) \sin 2\pi kt \, dt = 0 \qquad (5.25)$$

$$\int_0^1 \text{sal}(s,t) \cos 2\pi kt \, dt = 0 \qquad (5.26)$$

This simplifies the conversion process in that b_k terms depends only on B_s terms and a_k terms only on A_s terms. Similar relationships do not hold if the discrete orthonormal basis functions employed in the Fourier series expansion are $\{\sin 2\pi ki/N\}_{k=1}^{N/2}$, $\{\cos 2\pi ki/N\}_{k=1}^{N/2}$. This is illustrated for

$$\sum_{i=0}^{N-1} \text{cal}(1,i) \sin \frac{2\pi i}{N}$$

for N = 8 in Table 5.1.

Tadokoro and Higuchi simplified the conversion process by employing a displaced set of sinusoidal basis functions so that $\hat{a}_k$ terms depend only on A_s terms and $\hat{b}_k$ terms only on B_s terms. The sets of basis functions used are

$$\{\sin k [\frac{2\pi i}{N} + \frac{\pi}{N}]\}_{k=1}^{N/2} \qquad (5.27a)$$

$$\{\cos k [\frac{2\pi i}{N} + \frac{\pi}{N}]\}_{k=0}^{(N/2)-1} \qquad (5.27b)$$

With the above choice, Eqn. (5.24) reduces to

$$\hat{b}_k = \sum_{s=1}^{N/2} B_s \, \hat{b}_{k,s} \qquad (5.28)$$

i	0	1	2	3	4	5	6	7
cal(1,t)	1	1	-1	-1	-1	-1	1	1
$\sin \frac{2\pi i}{8}$	0	$1/\sqrt{2}$	+1	$+1/\sqrt{2}$	0	$-1/\sqrt{2}$	-1	$-1/\sqrt{2}$
cal(1,t) x $\sin \frac{2\pi i}{8}$	0	$1/\sqrt{2}$	-1	$-1/\sqrt{2}$	0	$+1/\sqrt{2}$	-1	$-1/\sqrt{2}$

$$\frac{1}{N} \sum_{i=0}^{N-1} \text{cal}(1,i) \sin \frac{2\pi i}{N} = (+1/\sqrt{2} - 1 - 1/\sqrt{2} + 1/\sqrt{2} - 1 - 1/\sqrt{2})/8$$

$$= -0.25$$

Table 5.1: First cal coefficient of $\sin \frac{2\pi i}{N}$

where $\hat{b}_{k,s}$ is the sth discrete sal coefficient of the kth discrete sine function. In matrix form, Eqn. (5.28) can be written

$$[\hat{\underline{b}}] = [\hat{F}_b]\ [\underline{B}] \qquad (5.29)$$

where $[\hat{\underline{b}}]$ = [N/2 x 1] sine coefficient matrix

$[\hat{F}_b]$ = [N/2 x N/2] conversion matrix

$[\underline{B}]$ = [N/2 x 1] sal coefficient matrix

The ith row in $[\hat{F}_b]$ is the discrete Walsh expansion of the ith sinusoid and can be obtained by performing a Walsh transform with the discrete sinusoid as the input data sequence.

Similar relations exist between the cal coefficients $[\underline{A}]$ and the cosine components $[\hat{\underline{a}}]$. The conversion matrix $[\hat{F}_b]$ for N = 16 is given in Table 5.2. The corresponding elements in the matrix $[\hat{F}_b]$ and F_b used in the previous method are not the same. The differences between them is due to the differences in the computation of the matrix elements and the basis sinusoidal functions used. Siemens and Kitai's method computes the Fourier coefficients as analog inner products of the signal and basis functions according to Eqns. (2.14) and (2.15) whereas Tadokoro and Higuchi's method computes the Discrete Fourier coefficients from the Discrete Walsh coefficients; they formulated the conversion process as the solution of N linear simultaneous equations. The pattern of non-zero elements in F_b and $\hat{F}_b$

Table 5.2: Conversion matrix $\hat{F}_b$ for N = 16

k\s	1	2	3	4	5	6	7	8
1	1.28145	0	-0.53079	0	-0.10558	0	-0.25488	0
2	0	1.30656	0	0	0	-0.54119	0	0
3	0.44998	0	1.08636	0	-0.72588	0	0.30066	0
4	0	0	0	1.41420	0	0	0	0
5	0.30066	0	0.72588	0	1.08636	0	-0.44998	0
6	0	.54118	0	0	0	1.30656	0	0
7	.25489	0	-0.10558	0	.53078	0	1.28145	0
8	0	0	0	0	0	0	0	1

are similar. Hence the number of multiplications needed in the conversion process using both the methods are equal. (The matrix F_b and K are multiplied and considered as a single matrix.)

The Discrete Fourier coefficients obtained through the second method are different from those obtained through conventional FFT methods as they assume a different set of sinusoidal functions. (The sets of basis functions assumed in the conventional DFT are

$$[\sin \tfrac{2\pi}{N} ki]_{k=1}^{N/2} \quad \text{and} \quad [\cos \tfrac{2\pi}{N} ki]_{k=0}^{(N/2)-1} .$$

Let c_k and d_k be respectively the real and imaginary parts of the kth conventional DFT coefficients. Then

$$c_k = \tfrac{1}{2} [a_k \cos \tfrac{k\pi}{N} + b_k \sin \tfrac{k\pi}{N}] \qquad (5.30)$$

$$d_k = \tfrac{1}{2} [a_k \sin \tfrac{k\pi}{N} - b_k \cos \tfrac{k\pi}{N}] \qquad (5.31)$$

In practice, the amplitude and the phase of the spectral components at different frequencies are of interest. To obtain the former, there is no need to transform the coefficients according to Eqns. (5.30) and (5.31). The latter can be obtained by adding a phase angle $k\pi/N$ to that obtained from the Fourier coefficients computed via the Walsh coefficients. Due to this, the additional transformation according to Eqns. (5.30, 5.31) which increases the number of multiplications by 2N may be of interest in

situations where only phase spectra are desired.

5.4 A Comparison of the Walsh-to-Fourier Conversion Process with the Cooley-Tukey FFT Algorithm of Computing the DFT

Over the years many fast algorithms [24,83,84,85] to compute the DFT have been developed. Some of the recent ones [87] are referred to as very fast Fourier transform algorithms. We limit ourselves to comparing the Walsh to Fourier Transformation with the commonly used Cooley and Tukey algorithm. The comparisons are made on the basis of the number of multiplications, memory requirements, and the effect of finite-word length in their implementation.

5.4.1 Speed

One of the factors normally considered in comparing the speeds of two signal processors is the number of multiplications required. The computation of the DFT coefficients using the Cooley-Tukey FFT algorithm according to Eqn. (5.15) requires $(N/2)\log_2 N$ complex multiplications i.e. $2N\log_2 N$ real multiplications (a complex multiplication requires four real multiplications).

The Walsh to Fourier conversion process described above is achieved by the multiplication of two matrices $\hat{F}_b$ and the Walsh coefficients matrix. The number of multiplications needed to obtain the sine components from

sal coefficients is equal to the number of non-zero elements in $\hat{F}_b$ matrix which is square and of dimension (N/2 x N/2) for a data sequence of length N. The odd-numbered rows of $\hat{F}_b$ contain N/4 non-zero elements. The kth sine coefficient is obtained by

$$\hat{b}_k = \sum_{i=1}^{z} \hat{b}_{k,i}\, B\{2^j(2i-1)\} \qquad (5.32)$$

where

$k = 2^j(2q-1)$, j,q are integers

$$z = (\frac{N}{4})\frac{1}{2^j} = \frac{N}{2^{j+2}}$$

Example: The values of k,z for N = 16 are

k	1	2	3	4	5	6	7	8
q	1	1	2	1	3	2	4	1
z	0	1	1	2	0	1	0	3

z determines the number of multiplications required for each $\hat{b}_k$. Summing over z for all $\hat{b}_k$s using Eqn. (5.32), the number of multiplications required is

$$M_s = \frac{N}{4} \times \frac{N}{4} + \frac{N}{8} \times \frac{N}{8} + \ldots + \frac{N}{16} \times \frac{1}{4^{k-1}} + \ldots + 4 + 1 \qquad (5.33a)$$

$$= \frac{(N^2-4)}{12} \qquad (5.33b)$$

An equal number of multiplications is required to obtain the Fourier cosine components, thereby making the total number of multiplications M_{tot} equal to $(N^2-4)/6$. M_{tot} for N = 8,

16, 32 is given below and is illustrated graphically in Fig. 5.2.

Number of Multiplications required for the Walsh to Fourier Conversion Process

Method \ N	8	16	32	64	128	256
FFT	48	128	320	768	1792	4096
Walsh to Fourier	10	42	170	682	2730	10,922

The Walsh to Fourier Conversion process is seen to be faster than the FFT method for data lengths up to 64. In certain applications [64,79], the number of frequency components of interest is relatively small compared to N. In the case of the FFT methods, the number of multiplications does not depend on the number of frequency components desired. But in the case of the Walsh to Fourier conversion, by limiting the conversion process to the selected number of components, one may reduce the number of multiplications. In that case, for $L = 2^r$ frequency components, the number of multiplications required is

$$M_L = 2\left[\frac{N}{4} \times \frac{L}{4} + \frac{NL}{64} + \ldots + \frac{N}{2^{r+1}} \frac{L}{2^{r+1}}\right] \qquad (5.34a)$$

$$= N(L^2-1)/64 \qquad (5.34b)$$

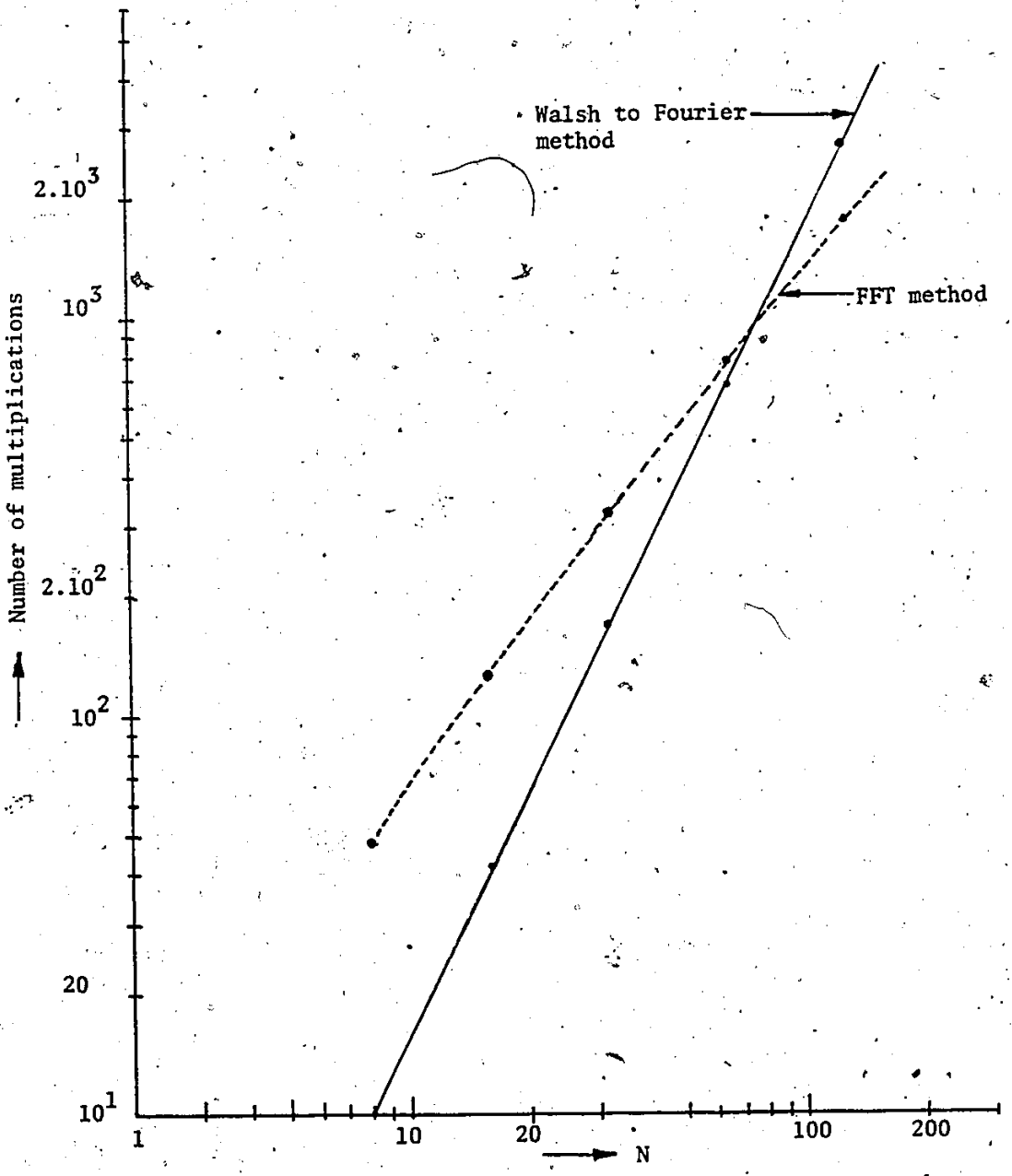


Fig. 5.2 Number of multiplications required to compute the Fourier coefficients by the FFT and Walsh to Fourier Conversion method

Given N, the number of multiplications required for different values of L is shown graphically in Fig. 5.3 where it is seen that the Walsh to Fourier conversion process requires fewer multiplications for all N, provided that L is relatively small compared to N.

5.4.2 Memory Requirements

The amount of memory capacity considered here represents only that required for the data and coefficients storage in a computer-based instrumentation for Fourier analysis. In the case of the FFT, N words are required for twiddle factors $\{w^k\}_{k=0}^{N/2}$ and 2N words for input data; a total of 3N words are required. (Since the FFT calculations are done on complex numbers, 2 words are assumed for one data point).

As the Fast Walsh Transform operates on real data yielding real-valued coefficients, the Walsh to Fourier conversion method requires N words for the data and $N^2/16$ words for the coefficient storage. The memory requirements for both methods for different values of N are:

N	8	16	32	64	128
FFT	24	48	96	192	384
Walsh to Fourier Conversion	12	32	96	320	1152

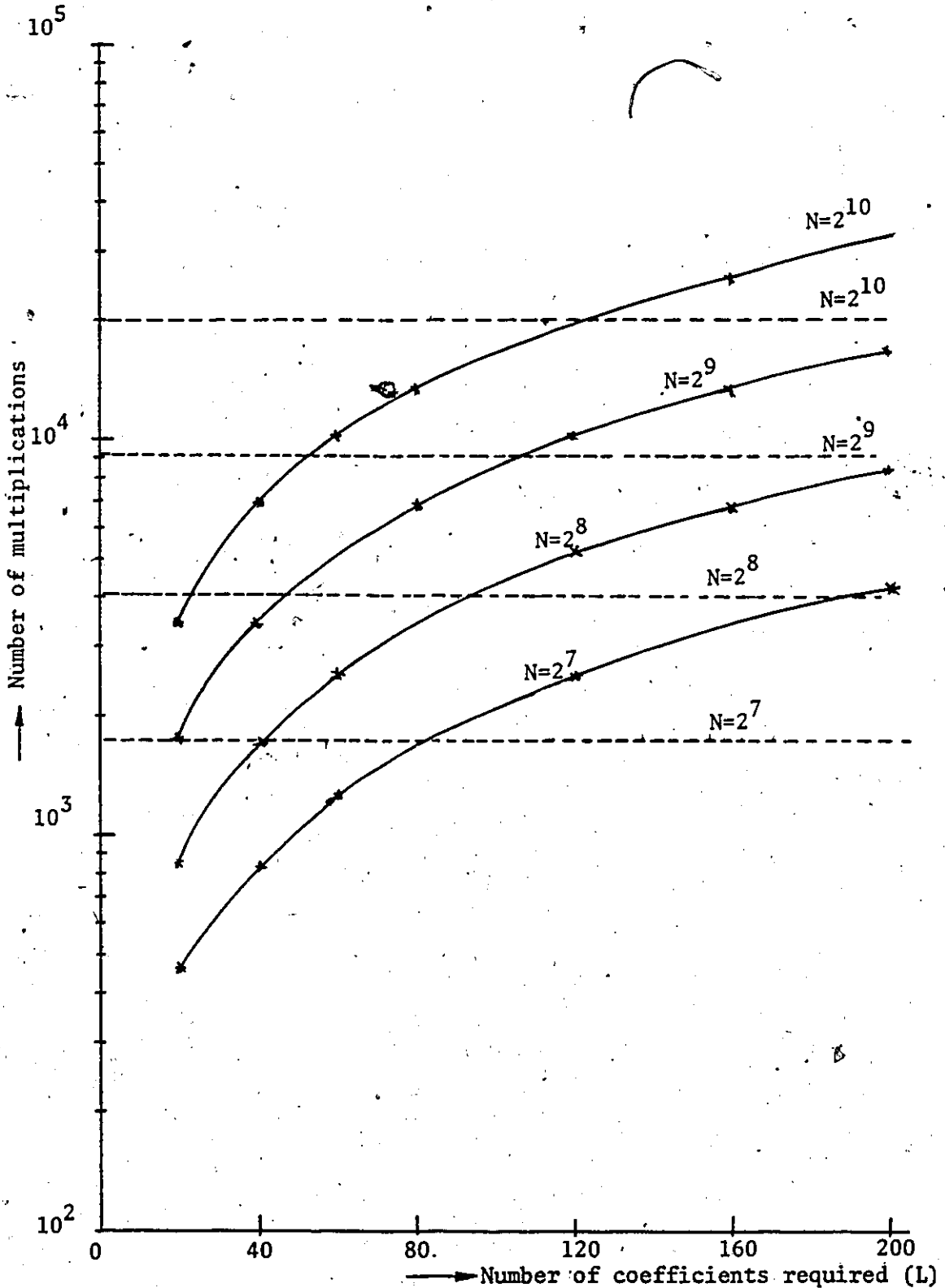


Fig. 5.3 Number of multiplications required in the Walsh to Fourier conversion method when the number L of desired components is less than N

This favors the Walsh to Fourier Method when compared to FFT for a data length less than or equal to 32. The memory requirements becomes excessive for large values of N for Walsh to Fourier conversion.

5.4.3 Effect of Finite Register Length

In practice, digital signal processing using general-purpose and special-purpose processors requires the representation of data in binary form with a finite number of bits. The results of processing may require additional bits for their representation. For example, a b-bit data sample multiplied by another b-bit data results in a product which is 2b bits long. The finite register length can be maintained by truncating or rounding the b least significant bits, the effects of which depend on such factors as whether we use fixed-point or floating-point arithmetic and on the type of representation for the data. In processing with fixed-point arithmetic it is natural, in a signal processing context, to consider a register as representing a fixed-point fraction. Then the products of two fractions remains a fraction and the finite word-length constraint can be maintained by truncating or rounding the least significant bits. The result of addition or subtraction need not be truncated or rounded but it can increase in magnitude so that eventually the sum is not a fraction. This overflow

can be handled by requiring that the input data be sufficiently small so that the possibility of overflow is avoided, or by scaling the results appropriately when overflow occurs.

In the following discussion we assume the binary data representation in two's complement form with (b+1) bits. Errors due to roundoff and truncation in the computation of the discrete Fourier transform were studied by Oppenheim and Weinstein [80] and Welsh [81] for the decimation in time (DIT) radix-2 fast Fourier transform algorithm. Recently Sundaramurthy and Reddy [82] obtained the error values for the DFT using the decimation in frequency (DIF) FFT algorithm. We describe briefly the statistical model used by them to obtain the error values, and we apply similar techniques to the Walsh to Fourier Transform algorithm described earlier to derive the computational errors. The results are compared with the FFT algorithms' error values [80,81,82].

The error due to a roundoff operation can be modelled as a random variable e_R uniformly distributed within the range $(-1/2)2^{-b}$ to $(1/2)2^{-b}$. It can be shown that the variance of this error is:

$$\sigma^2_{e_R} = \frac{1}{12} \times 2^{-2b} \qquad (5.35)$$

In a similar way the variance of the error due to scaling by a factor of 1/2 (shifting the number to the right by 1 bit and truncating the last bit shifted out) is [82],

$$\sigma^2_{e_s} = \frac{2^{-2b}}{8} \tag{5.36}$$

The signal flow-graphs depicting both the DIT and DIF FFT algorithms are shown in Fig. 5.4(a) and 5.4(b) respectively. At each stage, the algorithm passes through the entire array of N complex numbers, two at a time, generating a new N number array. Butterfly operation using the DIT FFT algorithm is

$$\begin{aligned} X_{k+1}(i) &= X_k(i) + \tilde{w}X_k(j) \\ X_{k+1}(j) &= X_k(i) - \tilde{w}X_k(j) \end{aligned} \tag{5.37}$$

The butterfly operation for the DIF FFT algorithm is

$$\begin{aligned} X_{k+1}(i) &= X_k(i) + X_k(j) \\ X_{k+1}(j) &= [X_k(i) - X_k(j)]\tilde{w} \end{aligned} \tag{5.38}$$

where $\tilde{w}$ is some appropriate power of $w = e^{-2\pi j/N}$.

In implementing the FFT algorithm with fixed-point arithmetic we must guard against overflow. From Eqns. (5.31) and (5.38), it follows that

$$\begin{aligned} &\max [|X_{k+1}(i)|, |X_{k+1}(j)|] \\ &\quad \leq 2 \max [|X_k(i)|, |X_k(j)|] \end{aligned} \tag{5.39}$$

Eqn. (5.39) suggests that the maximum modulus increases by no more than a factor of two from stage to stage. As a

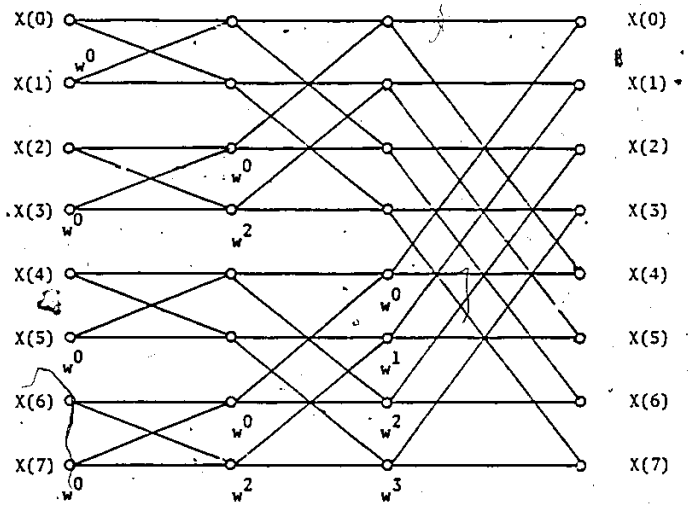


Fig. 5.4(a) Signal flow-graph of the decimation in time FFT algorithm

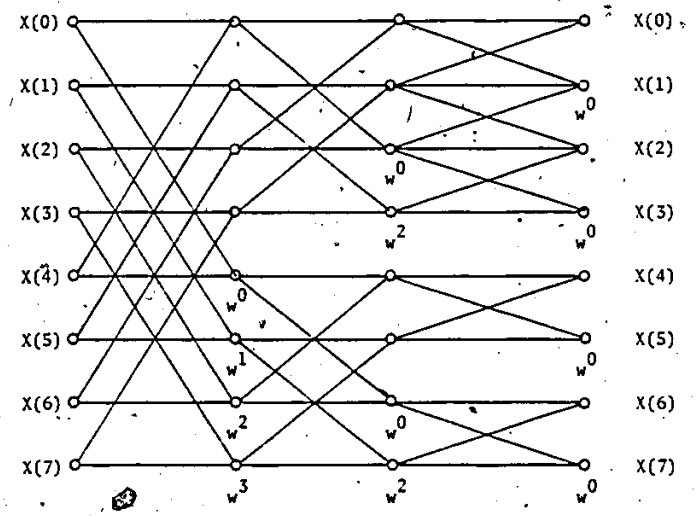


Fig. 5.4(b) Signal flow graph of the decimation in frequency FFT algorithm

result, overflow can be prevented by requiring that $|x(i)| < 1$ and incorporating an attenuation factor of 1/2 at each stage (right shift). Using this step by step scaling procedure, the final result is less than 1. The computation according to Eqn. (5.37) introduces one complex noise source due to rounding and another due to scaling at each node. In the case of Eqn. (5.38), one complex noise source due to scaling is introduced at each node but the roundoff error source is introduced at one of the two nodes in a butterfly operation. With this model for the noise sources due to computation, the variance of output errors for the DIT and DIF algorithms respectively are [82]

$$\sigma_{EF}^2\Big|_{DIT} = \frac{5}{3} 2^{-2b} \left[1 - \left(\frac{1}{2}\right)^m\right] \tag{5.40}$$

$$\sigma_{EF}^2\Big|_{DIF} = \frac{4}{3} 2^{-2b} \left[1 - \left(\frac{1}{2}\right)^m\right] \tag{5.41}$$

where $m = \log_2 N$.

A multiplication by factor 1 or j is noiseless. In Eqns. (5.40) and (5.41), it is assumed that such multiplications are noisy. The corresponding error values by subtracting the variance of the noise sources due to the noiseless multiplications are [82]

$$\sigma_{EF}^2\Big|_{DIT} = \frac{2^{-2b}}{3} \left(5 - \frac{4m}{N} - \frac{3}{N}\right) \tag{5.42}$$

$$\sigma^2_{EF}\Big|_{DIF} = \frac{2^{-2b}}{3}\left(\frac{19}{6} - \frac{4}{N} + \frac{4}{3N^2}\right) \qquad (5.43)$$

Walsh to Fourier Transformation Rounding and Scaling Errors

The transformation involves the computation of the Fast Walsh Transform followed by a matrix multiplication. As described in Chapter 3, the FWT performs a number of repeated butterfly operations analogus to the FFT algorithms. The butterfly operation in the case of the FWT is

$$X_{k+1}(i) = X_k(i) + X_k(j)$$
$$X_{k+1}(j) = X_k(i) - X_k(j) \qquad (5.44)$$

Assuming $|X_k(i)| < 1$, iteration by iteration scaling similar to the FFT algorithms is adopted to prevent overflow. It can be seen that the number of truncations errors propagating to any output node from the first, second, third and ith iteration are N/2, N/4, N/8 and $N/2^i$ respectively. The errors in the ith iteration undergo scalings by m-i times. Hence the total error due to scaling is

$$\sigma^2_T = \sigma^2_{e_s}\left[\frac{N}{2}\left(\frac{1}{4}\right)^{m-1} + \frac{N}{4}\left(\frac{1}{4}\right)^{m-2} + \ldots + \frac{N}{2^i}\left(\frac{1}{4}\right)^{m-i} + \ldots + 1\right] \qquad (5.45)$$

which reduces to

$$\sigma^2_T = 2\,\sigma^2_{e_s}\left[1 - \frac{1}{N}\right]$$
$$= \frac{2^{-2b}}{4}\left[1 - \frac{1}{N}\right] \qquad (5.46)$$

In obtaining the Fourier components, we perform a matrix multiplication. As discussed in section 5.4.1, a maximum of N/4 real multiplications are needed in the worst case. Note that the roundoff errors due to successive multiplication are scaled. Hence the maximum error due to roundoff in the computed sine/cosine Fourier components is

$$\sigma_R^2 = \sigma_{e_R}^2 \left[\left(\frac{1}{4}\right)^{(N/4)-1} + \left(\frac{1}{4}\right)^{(N/4)-2} + \ldots + \left(\frac{1}{4}\right)^{(N/4)-i} \ldots + \left(\frac{1}{4}\right) \right] \tag{5.47}$$

which reduces to

$$\sigma_R^2 = \sigma_{e_R}^2 \times \frac{4}{3} \left[1 - \frac{1}{(4)^{N/4}} \right] \tag{5.48}$$

The maximum variance of the error due to scaling is

$$\sigma_s^2 = \sigma_{e_s}^2 \left[\left(\frac{1}{4}\right)^{(N/4)-1} + \left(\frac{1}{4}\right)^{(N/4)-2} + \ldots + \left(\frac{1}{4}\right)^{(N/4)-i} + \ldots + \left(\frac{1}{4}\right) \right] \tag{5.49}$$

Hence the total error is

$$\sigma_{EW}^2\Big|_{WFT} = \left[\sigma_{e_s}^2 + \sigma_{e_R}^2\right] \frac{4}{3} \left[1 - \frac{1}{4^{N/4}}\right] + \frac{2^{-2b}}{4} \left[1 - \frac{1}{N}\right] \left[\frac{1}{4}\right]^{N/4} \tag{5.50}$$

The second term is due to the scaling errors introduced in the computation of the Walsh coefficients. Eqn. (5.50) after simplification becomes

$$\sigma_{EW}^2\Big|_{WFT} = \frac{5}{18} 2^{-2b} - \frac{2^{-2b}}{36} \left(\frac{1}{4}\right)^{N/4} - \frac{2^{-2b}}{4N} \left(\frac{1}{4}\right)^{N/4} \tag{5.51}$$

In general the kth sine or cosine components require a number of multiplications $z = N/2^{j+2}$ where j is defined in Eqn. (5.32). In that case the variance of the computational

error is given by

$$\sigma^2_{EW}\Big|_{WFT} = \frac{5}{18}\,2^{-2b} - \frac{2^{-2b}}{36}\,(\tfrac{1}{4})^{N/2^{j+2}} - \frac{2^{-2b}}{4N}\,(\tfrac{1}{4})^{N/2^{j+2}} \qquad (5.52)$$

Eqns. (5.51) and (5.52) give the variance of the error for either the sine or cosine components where as Eqns. (5.42) and (5.43) give the variances of the error for both the components combined (variance of a spectral component at a given frequency). When comparing the error performance of the FFT method with the Walsh to Fourier transformation, $2\sigma^2_T\big|_{WFT}$ should be used. From Eqns. (5.42), (5.43) and (5.51), the Walsh to Fourier transformation introduces a smaller computational error. The percentage improvement of over FFT algorithms is illustrated in Fig. 5.5.

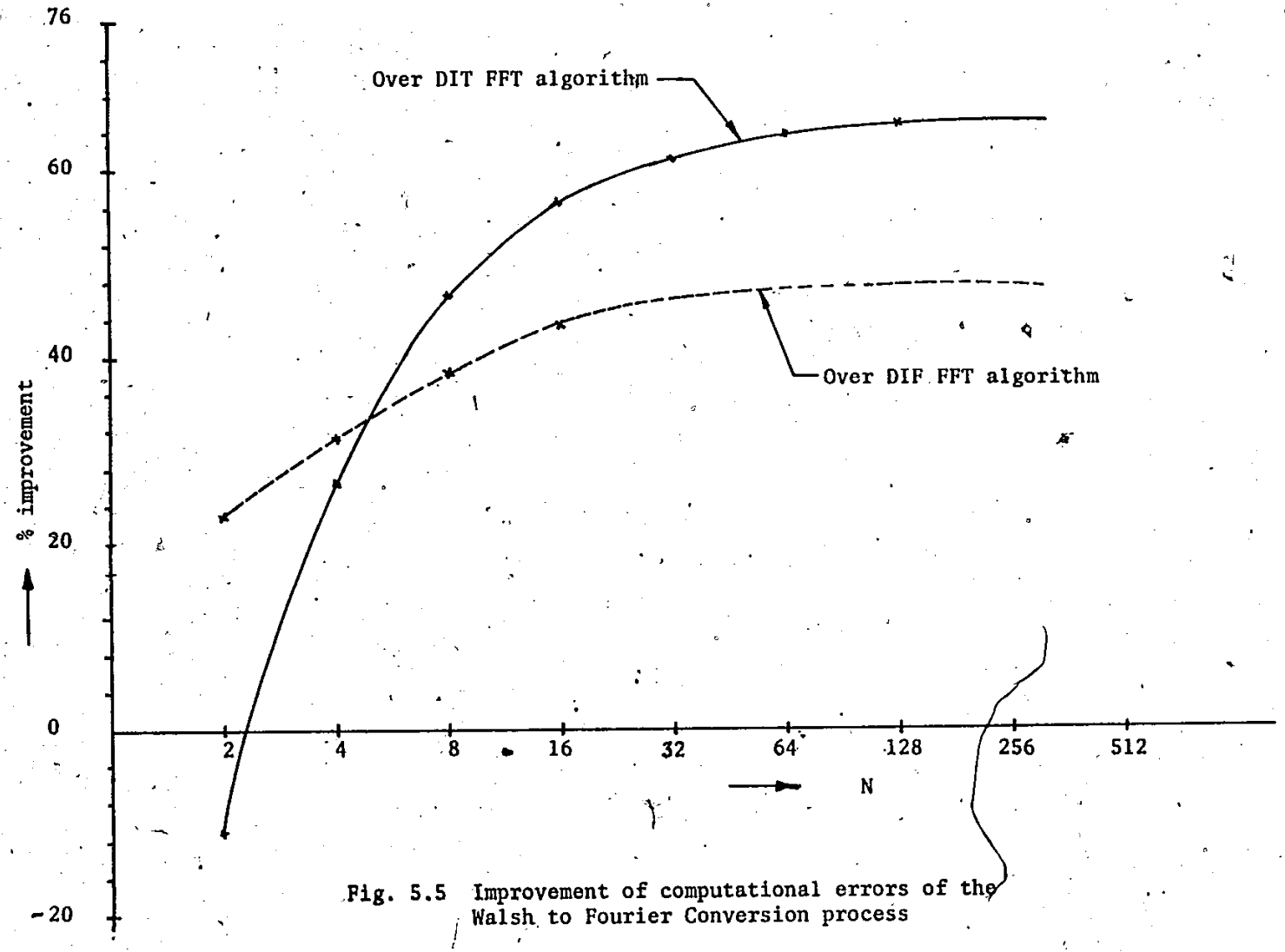


Fig. 5.5 Improvement of computational errors of the Walsh to Fourier Conversion process

CHAPTER 6

SOFTWARE DESIGN

6.1 Introduction

Software for the WSA is written in the form of functional modules or subroutines to be called by a main program. The list of subroutines is given later. Two of them, "WTFORM" and "WFCON" are special purpose modules for the WSA, and are described in detail below. Other subroutines used are general purpose in nature so that no descriptions are warranted. We also describe briefly the main program which links with a program written in BASIC in the HP2647A graphic terminal. Complete software details are available in a laboratory manual.

6.2 WTFORM (Fast Walsh Hadamard Transform) Subroutine

Chapter 4 shows that Walsh spectral analysis consists of two phases, viz, data acquisition and computation. The flow-chart of a subroutine developed for this purpose is shown in Fig. 6.1. It begins with the initialization of I/O ports of the system 80/10 (Port E8: output, E9: input, upper EA: input, lower EA: output), the address and the terminal count register of the DMA controller 8257 to BFBF and BFCO

(hexadecimal) respectively. The DMA controller is set to obtain 64 data samples with the starting address for the data storage as 4000 (Hexadecimal) and is in the "write" mode with the above parameter initialization. A "DMA request" signal is generated next in the form of a pulse via bit 5 of port EA and data acquisition commences. During this phase the system 80/10 is in the hold mode and resumes operation after 64 samples are deposited in the system memory.

The output of the A/D converter has 8 bits. The magnitude of a coefficient value may attain a maximum of $2^{14}-1$ for a 64 point discrete Walsh transform with the implied binary point after the 6th least significant bit; hence two word (16 bit) data representation is needed. The DMA hardware is designed so that the sampled A/D outputs are written with alternate memory locations skipped. In order to represent the data as two word numbers the data is reformatted so that the most significant byte is set to 11111111 or 00000000 depending on whether the least significant byte (sampled A/D output) is positive or negative.

The next step is to perform the FWHT (note that the FWHT on the permuted data gives the sequency ordered Walsh coefficients).

Based on the signal flow-graph (Fig. 3.3) for the FWHT, different variables are set up for the FWHT. Their

designations and assignments are given below.

Variable Name	Memory Location	Description
LOC1	3C 50 3C 51	16 bit (two word) variable containing the starting address of the data array i.e. X(0). (4000 is taken as the starting address.)
LOC2	3C 52 3C 53	16 bit (two word) variable containing the separation of two operands used in a butterfly operation. Initialized to 64 decimal and halved for each succeeding iteration.
LOC3	3C 54	8 bit (one word) variable containing the number of butterfly operations in a partition. Initialized to 32 decimal and is halved for the succeeding iteration.
LOC4	3C 55	8 bit (one word) variable containing the number of blocks in an iteration. Doubled for the succeeding iteration.
LOC5	3C 56	8 bit (one word) variable containing the number of iterations. Decremented by one at the completion of each iteration.

With the variables initialized to the values indicated above, the first iteration of FWHT commences. The register pair DE is loaded with the starting address of the data array and serves as a pointer to the address of the upper operand [X(i) in Eqn. (3.21)], for a butterfly operation. By adding the value of LOC2 loaded in register pair HL with the contents of register pair DE, the address of the second operand for a butterfly operation at the start

of an iteration is derived. Registers B and C are loaded with the contents of LOC3 and LOC4 respectively and serve as counters for the number of butterfly operations in a block and the number of blocks to be performed in an iteration.

To perform a butterfly operation, first $X(i) - X(i + N/2^j)$ is computed and is stored in the memory location occupied by $X(i + N/2^j)$ previously. Next $2X(i)$ is formed and the contents of the memory location occupied by $X(i + N/2^j)$ is subtracted from it, thereby obtaining $X(i) + X(i + N/2^j)$. The result is stored in memory locations occupied by $X(i)$. After each butterfly operation, register B is decremented and tested for zero (completion of a block). If not zero register pairs DE and HL are incremented to point to the next operands and a branch to the point 2 shown in the flow-chart (Fig. 6.1) is taken to perform the next butterfly operation.

When the contents of register B are zero, register C is decremented in an iteration and is tested for zero. In the case of a non-zero value, register pairs DE and HL are updated as shown in the flowchart (Fig. 6.1) to point to the operands in the next block. Also register B is loaded with the contents of LOC3. A branch to the point 2 in the flowchart is taken to continue the next block of butterfly operations.

A zero in the register C indicates that all the

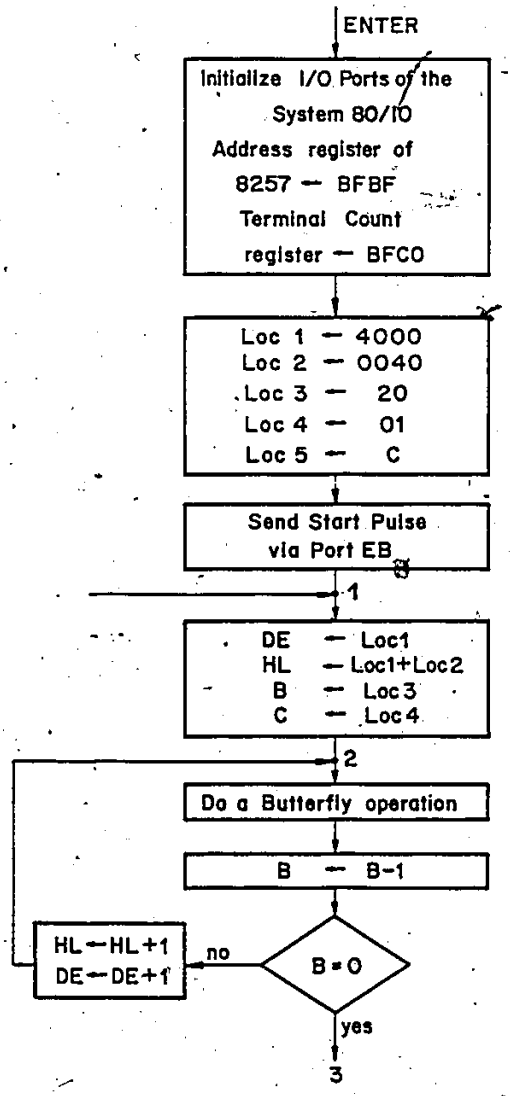


Fig. 6.1 Flowchart of WTFORM (FWHT) subroutine

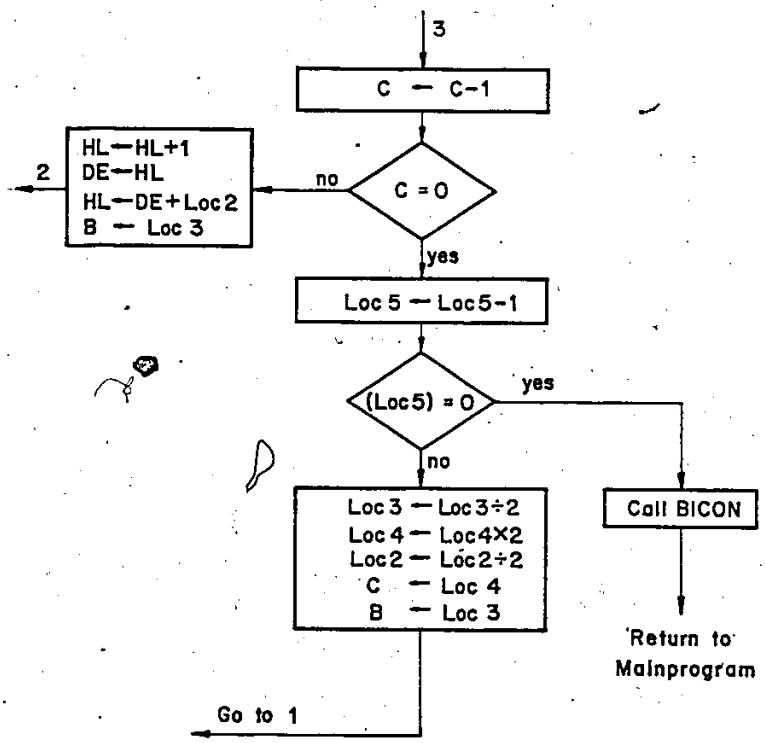


Fig. 6.1 Flowchart of WTFORM (FWHT) subroutine (continued)

butterfly operations in the current iteration are over. Now LOC5 is decremented and tested. If zero, the FWHT is completed and the subroutine "BICON" is called to convert the Walsh coefficients in binary form into signed BCD form. At the completion of BCD conversion, a return to the calling main program is taken.

If LOC5 is not zero, the contents of LOC3 (number of butterfly operations in a block) is halved, the contents of LOC4 (number of blocks in an iteration) is doubled and the contents of LOC2 (separation of two operands in a butterfly operation) is halved. With these updatings of the variables, a branch to point 1 in the flowchart (Fig. 6.1) is made to continue the next iteration. This completes the description of the flowchart.

The results are available both in binary and BCD form. The computation takes 23.844 msec for 64 Walsh coefficients. Binary to BCD conversion takes an additional 99.439 msec.

6.3 Walsh-to-Fourier Conversion Subroutine (WFCON)

Either of the two processes described in Chapter 5 can be considered for software implementation of Walsh to Fourier conversion in a microcomputer based WSA. The process due to Siemens and Kitai would yield Fourier coefficients without the numerical integration error

associated with the DFT coefficients provided the Walsh coefficients are computed without integration error; this is feasible in the instrument described in the thesis, if the signal is integrated over each interval and the integrator output is used as the data sequence for FWT [14]. The DFT components obtained by the process due to Tudakoro and Higuchi are different from those obtained from conventional FFT methods and requires correction in obtaining phase information of a spectral component as discussed in Section 5.3. In view of this, the process due to Siemens and Kitai was chosen.

The first microprocessor based Walsh to Fourier converter [75] used the first-generation 4004 microprocessor. The conversion time was 1.81 seconds for 32 coefficients. The slow speed can be attributed to the long cycle time of 4004. A later system [76] was based on the decimal PPS-25 microprocessor, the conversion time for 64 decimal coefficients being 2.5 seconds.

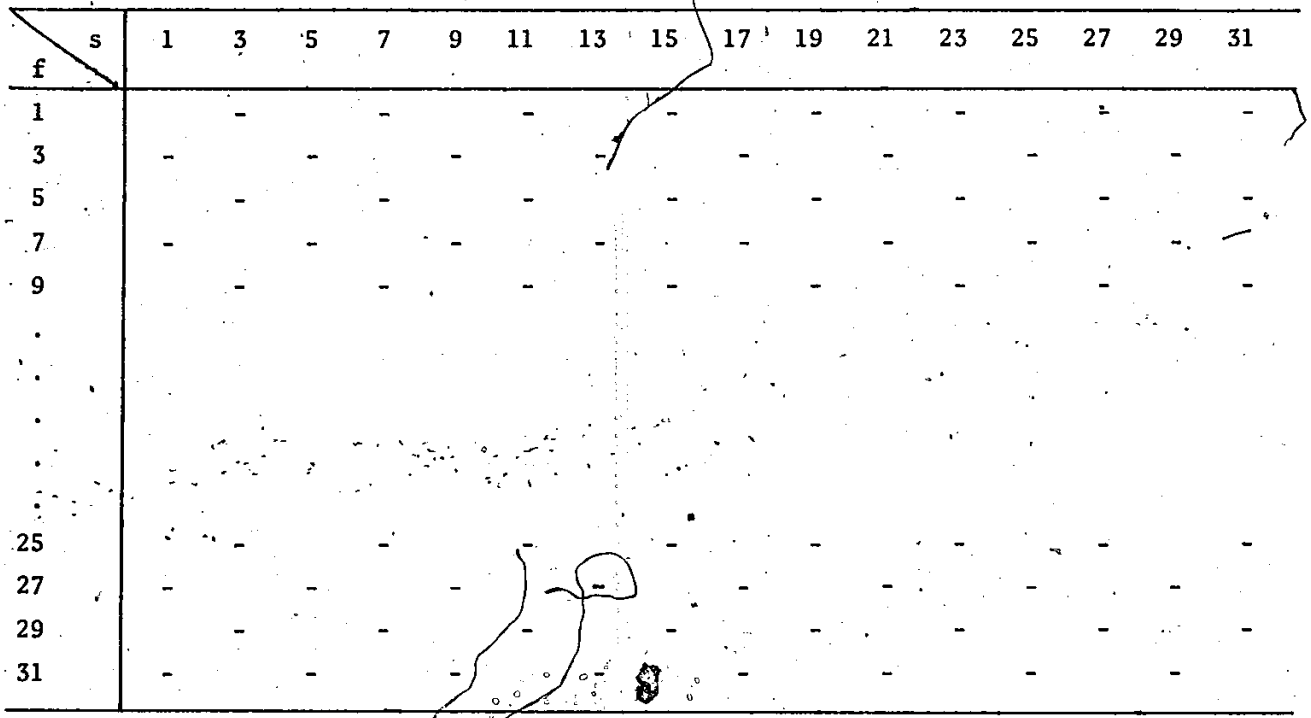
In the present work, the 64 binary-coded Fourier coefficients are obtained from 64 Walsh coefficients in about one second.

Data Representation and Storage

The Walsh coefficients computed are in 16 bit two's complement form with the implied binary point after the 6th lsb. Two memory locations are used for each coefficient.

As a result, two's complement representation for matrix elements is adopted in contrast to sign and magnitude representation in [75] and signed BCD form in [76]. Each conversion matrix element has a 9 bit fractional part and a 1 bit integer part. It is packed as two bytes with the remaining bits set to 1 or 0 depending on whether the coefficient is positive or negative. Booth's algorithm [88], which is independent of the signs of the operands for multiplication, is employed. The multiplication time is smaller, on the average, than conventional shift and add methods of multiplication. The Fourier coefficients computed are 32 bits long.

Observation of the conversion matrix elements for sine and cosine components shows that the magnitude of the corresponding elements in the two matrices are the same, but the signs may differ. Earlier implementations stored the magnitude and the signs of coefficients for sine and cosine components separately, resulting in additional software overheads to fetch the corresponding signs during the conversion process. It is seen that the signs of cosine conversion matrix elements either differ or do not differ from the signs of the corresponding sine conversion matrix elements alternately as shown in Fig. 6.2. Using this property, during accumulation, a partial result of a multiplication is either added to or subtracted from the



- denotes a sign change. No entry denotes no sign change

Fig. 6.2 Pattern of sign changes of the conversion matrix elements for cosine components from the corresponding conversion matrix elements for sine components

accumulated sum alternately for cosine components, assuming that the conversion matrix elements for sine components are stored.

The conversion process is achieved in two stages; in the first sal coefficient matrix $\underline{B}$ is multiplied with F_b and in the second, uncompensated Fourier components are multiplied by the appropriate compensation factors. This procedure requires the storage of only odd numbered rows of F_b and the diagonal elements of the matrix $[K]^{-1}$; i.e. $N^2/16 + N/2$ elements. If implemented as a one-stage procedure, the elements of a matrix obtained by the product of two matrices F_a and $[K]^{-1}$ need to be stored; i.e. $(N^2-4)/12$ elements are required and this is greater than that for the former scheme. If the input signal is sequency limited to the first N/2 cal or sal components, the corresponding N/2 sine or cosine components are given by $F_b\underline{B}$ or $F_a\underline{A}$; this can be realized by excluding the second stage in a two stage conversion process.

The number of multiplications increases by N in the two stage procedure over that of one stage procedure.

The matrix elements $b_{1,1}$, $b_{1,3}$ ... $b_{1,(N/2)-1}$, $b_{3,1}$, $b_{3,3}$, ..., $b_{3,(N/2)-1}$, ... $b_{2k-1,1}$, $b_{2k-1,3}$ ..., $b_{(N/2)-1,3}$ ... $b_{(N/2)-1,(N/2)-1}$ are stored in contiguous memory locations starting with the location OAOO (Hexadecimal). Zero-valued elements of F_b are not stored. The conversion

matrix F_b elements are given both in decimal and Hexadecimal form in Table 6.1.

Flowchart for Walsh-to-Fourier Conversion Subroutine

The main task of the conversion subroutine is to retrieve appropriate Walsh coefficients and conversion matrix elements for multiplication to obtain Fourier coefficients according to Eqn. (5.32). Siemens' scheme for software implementation [73] was adopted in the previous designs [74,75]. Only non-zero unique elements of conversion matrix F_b are stored as explained before, Based on this storage scheme, Siemens' procedure uses the following algorithm. Let f be the order of a component in the Fourier sine coefficient matrix b; it can be expressed as

$$f = 2^k(2q-1) \qquad (6.1)$$

where q defines the row wherein the conversion coefficients for the fth component are stored. Note that not all the coefficients in that row are used for a given f. k in Eqn. (6.1) is used in retrieving the Walsh coefficients in the order 2^k, $3(2^k)$, $5(2^k)$ ...

The above algorithm requires the computation of k, q for each f, in addition to the process of generating actual memory addresses based on data representation using k and q.

A different approach is adopted which avoids these computations. The flowchart of the subroutine is shown in

Table 6.1: F_b matrix elements

Element No.	Value		Element No.	Value	
	Decimal	Hex		Decimal	Hex
$b_{1,1}$	1.273240	02 8B	$b_{5,1}$	0.254648	00 82
$b_{1,3}$	-0.527393	FE F2	$b_{5,3}$	0.614774	01 3A
$b_{1,5}$	-0.104905	FF CB	$b_{5,5}$	0.920075	01 D7
$b_{1,7}$	-0.253263	FF 7F	$b_{5,7}$	-0.381108	FF 3D
$b_{1,9}$	-0.024944	FF F4	$b_{5,9}$	-0.102706	FF 98
$b_{1,11}$	0.010332	00 05	$b_{5,11}$	-0.491790	FF 05
$b_{1,13}$	-0.051944	FF EG	$b_{5,13}$	0.328604	00 A8
$b_{1,15}$	-0.125403	FF CO	$b_{5,15}$	-0.136112	FF BB
$b_{1,17}$	-0.006161	FF FD	$b_{5,17}$	-0.034094	FF EF
$b_{1,19}$	0.002552	00 01	$b_{5,19}$	-0.082311	FF DG
$b_{1,21}$	0.000508	00 00	$b_{5,21}$	-0.123187	FF C1
$b_{1,23}$	0.001225	00 00	$b_{5,23}$	0.051026	OO 1A
$b_{1,25}$	-0.012442	FF FA	$b_{5,25}$	-0.095462	FF DO
$b_{1,27}$	0.005154	00 02	$b_{5,27}$	-0.230467	FF 8B
$b_{1,29}$	-0.025909	FF F3	$b_{5,29}$	0.153993	00 4E
$b_{1,31}$	-0.062550	FF EO	$b_{5,31}$	-0.063786	FF EO
$b_{3,1}$	0.424413	00 D9	$b_{7,1}$	0.181891	00 5D
$b_{3,3}$	1.024624	02 0C	$b_{7,3}$	-0.075342	FF DA
$b_{3,5}$	-0.684632	FE A2	$b_{7,5}$	0.378769	00 C1
$b_{3,7}$	0.283584	00 91	$b_{7,7}$	0.914430	01 D4
$b_{3,9}$	-0.086024	FF D4	$b_{7,9}$	-0.750453	FE 80
$b_{3,11}$	-0.207681	FF 96	$b_{7,11}$	0.310848	00 9F
$b_{3,13}$	-0.310816	FF G1	$b_{7,13}$	0.061831	00 1F
$b_{3,15}$	0.128744	00 41	$b_{7,15}$	0.149274	00 4C
$b_{3,17}$	-0.019097	FF F7	$b_{7,17}$	-0.053411	FF E5
$b_{3,19}$	-0.046105	FF E9	$b_{7,19}$	0.022124	00 0B
$b_{3,21}$	0.030807	00 0F	$b_{7,21}$	-0.111223	FF C8
$b_{3,23}$	-0.012760	FF FA	$b_{7,23}$	-0.268516	FF 77
$b_{3,25}$	-0.042066	FF EB	$b_{7,25}$	-0.327188	FF 59
$b_{3,27}$	-0.101556	FF CD	$b_{7,27}$	0.135526	00 45
$b_{3,29}$	-0.151989	FF B3	$b_{7,29}$	0.026958	00 0D
$b_{3,31}$	0.062956	00 20	$b_{7,31}$	0.065082	00 21

Table 6.1 Continued

Element No.	Value		Element No.	Value	
	Decimal	Hex		Decimal	Hex
$b_{9,1}$	0.141471	00 48	$b_{13,1}$	0.097942	00 32
$b_{9,3}$	-0.058599	FF E2	$b_{13,3}$	0.236452	00 79
$b_{9,5}$	0.294598	00 96	$b_{13,5}$	-0.157992	FF BO
$b_{9,7}$	0.711223	01 6C	$b_{13,7}$	0.065442	00 21
$b_{9,9}$	0.866628	01 BB	$b_{13,9}$	0.215735	00 6E
$b_{9,11}$	-0.358960	FF 49	$b_{13,11}$	0.520830	01 0A
$b_{9,13}$	-0.071403	FF DC	$b_{13,13}$	0.779477	01 8F
$b_{9,15}$	-0.172383	FF A8	$b_{13,15}$	-0.322870	FF 5B
$b_{9,17}$	-0.081531	FF D7	$b_{13,17}$	-0.239457	FF 86
$b_{9,19}$	0.033771	00 11	$b_{13,19}$	-0.578099	FE D9
$b_{9,21}$	-0.169780	FF AA	$b_{13,21}$	0.386274	00 C5
$b_{9,23}$	-0.409884	FF 2F	$b_{13,23}$	-0.160000	FF AF
$b_{9,25}$	0.336383	00 AC	$b_{13,25}$	0.048535	00 18
$b_{9,27}$	-0.139335	FF B9	$b_{13,27}$	0.117175	00 3B
$b_{9,29}$	-0.027715	FF F2	$b_{13,29}$	0.175365	00 59
$b_{9,31}$	-0.066911	FF DE	$b_{13,31}$	-0.072638	FF DB
$b_{11,1}$	0.115749	00 3B	$b_{15,1}$	0.084883	00 2B
$b_{11,3}$	0.279443	00 8F	$b_{15,3}$	-0.035160	FF EE
$b_{11,5}$	0.418216	00 D6	$b_{15,5}$	-0.006994	FF FD
$b_{11,7}$	-0.173231	FF A8	$b_{15,7}$	-0.016884	FF F8
$b_{11,9}$	0.324092	00 A5	$b_{15,9}$	0.171428	00 57
$b_{11,11}$	0.782427	01 90	$b_{15,11}$	-0.071008	FF DC
$b_{11,13}$	-0.522801	FE F5	$b_{15,13}$	0.356981	00 B6
$b_{11,15}$	0.216551	00 6E	$b_{15,15}$	0.861828	01 B9
$b_{11,17}$	-0.129796	FF BE	$b_{15,17}$	-0.781115	FE 71
$b_{11,19}$	-0.313355	FF G0	$b_{15,19}$	0.323549	00 A5
$b_{11,21}$	-0.468969	FF 10	$b_{15,21}$	0.064358	00 20
$b_{11,23}$	0.194253	00 63	$b_{15,23}$	0.155373	00 4F
$b_{11,25}$	0.103830	00 35	$b_{15,25}$	0.015303	00 07
$b_{11,27}$	0.250669	00 80	$b_{15,27}$	-0.006339	FF FD
$b_{11,29}$	-0.167492	FF AB	$b_{15,29}$	0.031867	00 10
$b_{11,31}$	0.069377	00 23	$b_{15,31}$	0.076933	00 27

Table 6.1 continued

Element No.	Value		Element No.	Value	
	Decimal	Hex		Decimal	Hex
$b_{17,1}$	0.074896	00 26	$b_{21,1}$	0.060630	00 1F
$b_{17,3}$	-0.031023	FF F1	$b_{21,3}$	0.146375	00 4A
$b_{17,5}$	-0.006171	FF FD	$b_{21,5}$	0.219065	00 70
$b_{17,7}$	-0.014898	FF F9	$b_{21,7}$	-0.090740	FF D2
$b_{17,9}$	0.151260	00 4D	$b_{21,9}$	0.169762	00 56
$b_{17,11}$	-0.062654	FF E0	$b_{21,11}$	0.409843	00 D1
$b_{17,13}$	0.314983	00 A1	$b_{21,13}$	-0.273848	FF 74
$b_{17,15}$	0.760436	01 85	$b_{21,15}$	0.113432	00 3A
$b_{17,17}$	0.839012	01 AD	$b_{21,17}$	0.189249	00 60
$b_{17,19}$	-0.347530	FF 4F	$b_{21,19}$	0.456888	00 E9
$b_{17,21}$	-0.069128	FF DD	$b_{21,21}$	0.683781	01 5E
$b_{17,23}$	-0.166890	FF AB	$b_{21,23}$	-0.283231	FF 6F
$b_{17,25}$	-0.016437	FF F8	$b_{21,25}$	-0.151390	FF B3
$b_{17,27}$	0.006809	00 03	$b_{21,27}$	-0.365489	FF 45
$b_{17,29}$	-0.034229	FF EF	$b_{21,29}$	0.244212	00 7D
$b_{17,31}$	-0.082635	FF DG	$b_{21,31}$	-0.101156	FF CD
$b_{19,1}$	0.067013	00 22	$b_{23,1}$	0.055358	00 1C
$b_{19,3}$	0.161783	00 52	$b_{23,3}$	-0.022930	FF F5
$b_{19,5}$	-0.108100	FF C9	$b_{23,5}$	0.115278	00 3B
$b_{19,7}$	0.044776	00 16	$b_{23,7}$	0.278305	00 8E
$b_{19,9}$	0.147608	00 4B	$b_{23,9}$	0.339115	00 AD
$b_{19,11}$	0.356357	00 B6	$b_{23,11}$	-0.140446	FF B9
$b_{19,13}$	0.533326	01 11	$b_{23,13}$	-0.027940	FF F2
$b_{19,15}$	-0.220911	FF 8F	$b_{23,15}$	-0.067454	FF DE
$b_{19,17}$	0.297864	00 98	$b_{23,17}$	0.142620	00 49
$b_{19,19}$	0.719107	01 70	$b_{23,19}$	-0.059075	FF E2
$b_{19,21}$	-0.480492	FF 0A	$b_{23,21}$	0.296991	00 98
$b_{19,23}$	0.199026	00 65	$b_{23,23}$	0.716999	01 6F
$b_{19,25}$	-0.060374	FF E2	$b_{23,25}$	-0.588426	FE D3
$b_{19,27}$	-0.145756	FF B6	$b_{23,27}$	0.243734	00 7C
$b_{19,29}$	-0.218139	FF 91	$b_{23,29}$	0.048482	00 18
$b_{19,31}$	0.090356	00 2D	$b_{23,31}$	0.117045	00 3B

Table 6.1 continued

Element No.	Value		Element No.	Value	
	Decimal	Hex		Decimal	Hex
$b_{25,1}$	0.050930	00 1A	$b_{29,1}$	0.043905	00 16
$b_{25,3}$	-0.021096	FF F6	$b_{29,3}$	0.105996	00 36
$b_{25,5}$	0.106055	00 36	$b_{29,5}$	-0.070824	FF DC
$b_{25,7}$	0.256040	00 83	$b_{29,7}$	0.029336	00 0F
$b_{25,9}$	-0.210127	FF 95	$b_{29,9}$	-0.008899	FF FC
$b_{25,11}$	0.087037	00 2C	$b_{29,11}$	-0.021484	FF F6
$b_{25,13}$	0.017313	00 08	$b_{29,13}$	-0.032153	FF F0
$b_{25,15}$	0.041797	00 15	$b_{29,15}$	0.013318	00 06
$b_{25,17}$	0.116814	00 3B	$b_{29,17}$	0.089785	00 2D
$b_{25,19}$	-0.048386	FF E8	$b_{29,19}$	0.216761	00 6E
$b_{25,21}$	0.243253	00 7C	$b_{29,21}$	-0.144835	FF B6
$b_{25,23}$	0.587265	01 2C	$b_{29,23}$	0.059993	00 1E
$b_{25,25}$	0.715585	01 6E	$b_{29,25}$	0.197769	00 65
$b_{25,27}$	-0.296405	FF 69	$b_{29,27}$	0.477457	00 F4
$b_{25,29}$	-0.058959	FF E2	$b_{29,29}$	0.714564	01 6D
$b_{25,31}$	-0.142339	FF B8	$b_{29,31}$	-0.295982	FF 69
$b_{27,1}$	0.047157	00 18	$b_{31,1}$	0.041072	00 15
$b_{27,3}$	0.113847	00 3A	$b_{31,3}$	-0.017013	FF F8
$b_{27,5}$	0.170384	00 57	$b_{31,5}$	-0.003384	FF FF
$b_{27,7}$	-0.070575	FF DC	$b_{31,7}$	-0.008170	FF FC
$b_{27,9}$	-0.037723	FF ED	$b_{31,9}$	-0.000805	00 00
$b_{27,11}$	-0.091072	FF D2	$b_{31,11}$	0.000333	00 00
$b_{27,13}$	0.060853	00 1F	$b_{31,13}$	-0.001676	00 00
$b_{27,15}$	-0.025206	FF F4	$b_{31,15}$	-0.004045	FF FE
$b_{27,17}$	0.100628	00 33	$b_{31,17}$	0.082343	00 2A
$b_{27,19}$	0.242937	00 7C	$b_{31,19}$	-0.034108	FF EF
$b_{27,21}$	0.363581	00 BA	$b_{31,21}$	-0.006784	FF ED
$b_{27,23}$	-0.150600	FFF B3	$b_{31,23}$	-0.016379	FF F8
$b_{27,25}$	0.281753	00 90	$b_{31,25}$	0.166300	00 55
$b_{27,27}$	0.680212	01 5C	$b_{31,27}$	-0.068884	FF DD
$b_{27,29}$	-0.454503	FF 18	$b_{31,29}$	0.346301	00 B1
$b_{27,31}$	0.188261	00 60	$b_{31,31}$	0.836045	01 AC

Fig. 6.3. Three address pointers are used for (1) the conversion matrix elements (2) the Walsh coefficients and (3) the Fourier coefficients (each treated as an one-dimensional array) and are denoted MPRA, MPDA and STRA respectively. In computations, these pointers are updated using the properties of the conversion matrix. They are initialized to the addresses of the first elements of the respective arrays. The matrix multiplication is carried out as follows:

Step 1: The conversion process begins with the first matrix element of the first row in F_b and all of the multiplications associated with it are completed. The multiplications associated with a conversion element - say $b_{1,1}$ - can be written as $(b_{1,1})\ (B_1)$, $(b_{1,1})\ (B_2)$, $(b_{1,1})\ (B_4)$ ... $(b_{1,1})\ (B_{2^k})$; $2^k \leq N/2$. This shows that the index of Walsh coefficients associated with $b_{1,1}$ is doubled for sucessive multiplications starting with 1. Each successive product is assigned to a Fourier component whose index is also twice the previous value. To update the pointers MPDA, STRA, four variables denoted SPAN1, SPAN2, SPAN3 and SPAN4 are set up. SPAN1 and SPAN2 are initialized to 1; SPAN3 and SPAN4 are initialized to 2 (for 32 bits results). SPAN2 and SPAN4 are doubled by a left shift operation and are added to the current values of the pointers MPDA and STRA respectively after each multiplication. The uses of SPAN1

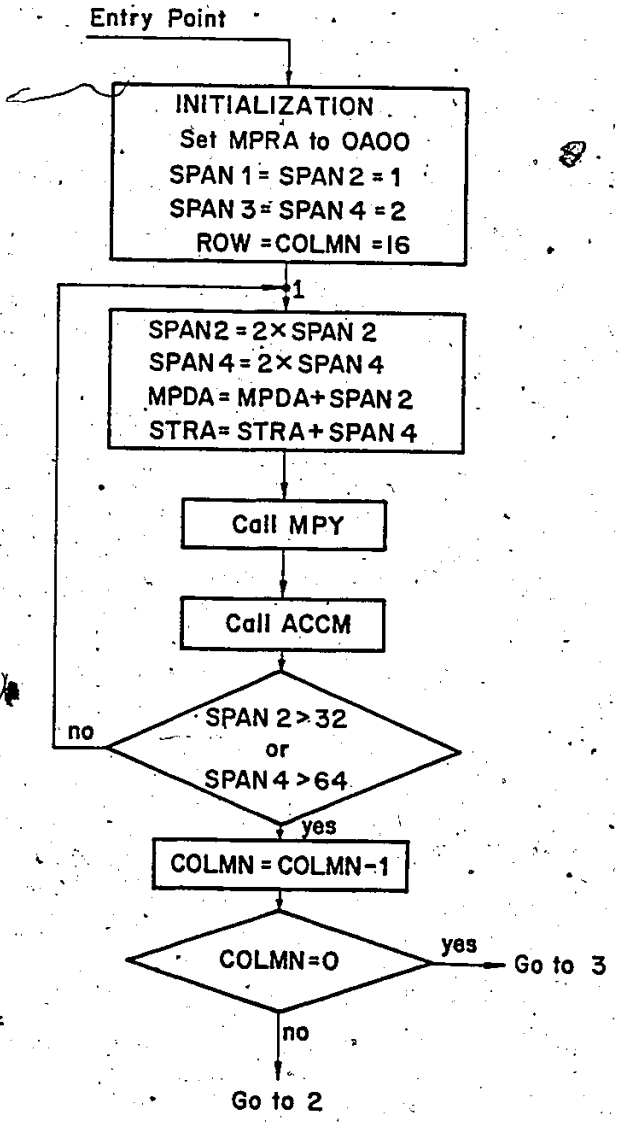


Fig. 6.3 Flowchart of Walsh to Fourier conversion subroutine

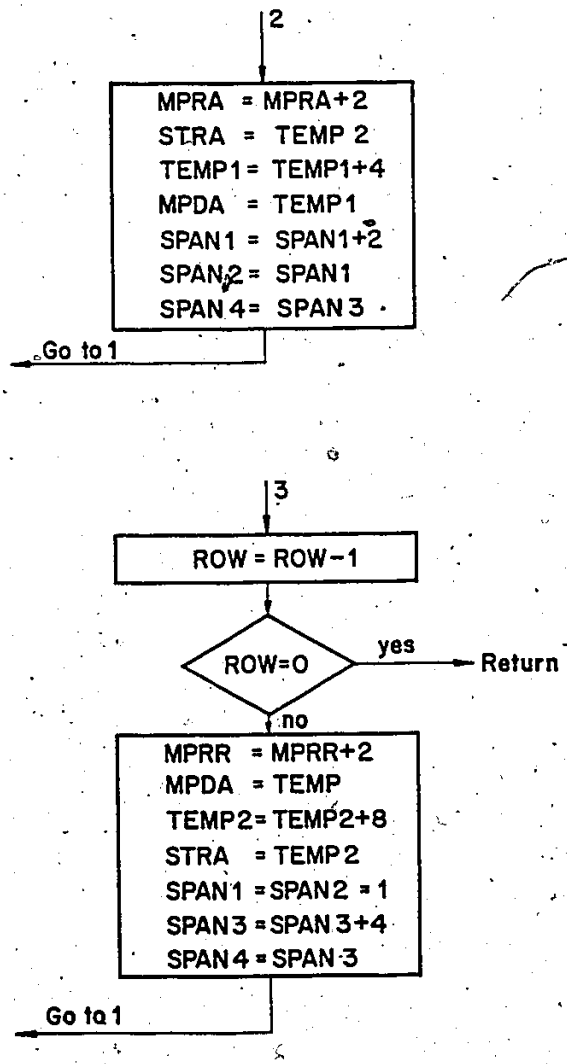


Fig. 6.3 Flowchart of Walsh to Fourier conversion subroutine (continued)

and SPAN3 will be explained later.

The multiplications associated with a conversion element is terminated when either SPAN2 exceeds 32 or SPAN4 exceeds 64. Then all multiplications associated with the conversion element are completed.

Step 2: The next matrix element in the same row as in step 1 is used for multiplications; this element is retrieved with the pointer MPRA incremented by 2. The index of the first Walsh coefficient associated with this element would be higher than the index of the initial Walsh coefficient for the previous element by 2. SPAN1 which stores the later is incremented by 2; SPAN2 is reinitialized to the value of SPAN1. Since the products are contributions to Fourier components with the same starting index as for the previous element, SPAN4 is reinitialized to the value of SPAN3. Then step 1 is followed.

When all of the multiplications corresponding to one row is complete, step 3 is followed. This can be accomplished by way of a counter which is preset to 16.

Step 3: The starting index of the Fourier component to which the products of multiplications associated with the new row of elements are assigned is greater than that of the previous row by 2. For example, the products associated with the elements in the third row are assigned to F_3, F_6, F_{12} ... $F_{2^{k_{(3)}}}$. As a result STRA is initialized to the

previous starting value plus 8. SPAN3 which stores the index of the first Fourier component corresponding to a row is incremented by 4; SPAN4 is also set equal to SPAN3. The index of the Walsh coefficients associated with the first element of the new row is 1 and hence SPAN1 and SPAN2 are set to 1. Also MPDA is set to point to the beginning of Walsh coefficient array. Then step 1 is followed.

If all of the rows of matrix elements are used, the conversion process is complete and is returned to the main program.

The program is written as a subroutine with the entry point OD42. The subroutine assumes a data structure for Walsh coefficients with cal and sal coefficients interleaved as would be obtained by the FWHT algorithm.

The conversion time for 32 sine or cosine components is 493 msec maximum.

This subroutine computes uncompensated sine or cosine components. To compute both, it has to be executed twice with the pointers initialized properly. Then they are compensated by multiplying the Fourier components with compensation factors which are listed in decimal and hexadecimal form in Table 6.2.

Table 6.2: Compensation matrix elements

No	Value	
	Decimal	Hex
$K_{1,1}$	1.00080	0100
$K_{2,2}$	1.00322	0101
$K_{3,3}$	1.00726	0102
$K_{4,4}$	1.01295	0103
$K_{5,5}$	1.02032	0105
$K_{6,6}$	1.02942	0107
$K_{7,7}$	1.04030	010A
$K_{8,8}$	1.05303	010E
$K_{9,9}$	1.06767	0111
$K_{10,10}$	1.08434	0116
$K_{11,11}$	1.10312	011A
$K_{12,12}$	1.12415	0120
$K_{13,13}$	1.14755	0126
$K_{14,14}$	1.17349	012C
$K_{15,15}$	1.20214	0134
$K_{16,16}$	1.23370	013C
$K_{17,17}$	1.26841	0145
$K_{18,18}$	1.30651	014E
$K_{19,19}$	1.34831	0159
$K_{20,20}$	1.39414	0165
$K_{21,21}$	1.44437	0172
$K_{22,22}$	1.49942	0180
$K_{23,23}$	1.55979	018F
$K_{24,24}$	1.62604	01A0
$K_{25,25}$	1.69879	01B3
$K_{26,26}$	1.77876	01C7
$K_{27,27}$	1.86670	01DE
$K_{28,28}$	1.96385	01F7
$K_{29,29}$	2.07104	0212
$K_{30,30}$	2.18965	0231
$K_{31,31}$	2.32119	0252
$K_{32,32}$	1.23370	015C

6.4 List of Other Subroutines Used in the WSA

1. SINCON: Initializes the variables to obtain the Fourier sine components from sal coefficients and computes sine components by calling subroutine WFCON. The entry point is 0E60.
2. COSCON: Initializes the variables to obtain the Fourier cosine components from cal coefficients and computes cosine components by calling the subroutine WFCON. The entry point is 0E80.
3. MPY: Multiplies two 16 bit binary numbers in two's complement form using Booth's algorithm. The entry point is 0F60.
4. ACCM: Used to either add or subtract a 32 bit word in registers BCDE with a 32 bit word stored in the system memory. Used by WFCON subroutine. The entry point is 0EA3.
5. BICON: Converts an array of 64 16-bit binary numbers into an array of 64 signed BCD numbers. The entry point is 0F00.
6. CLEAR: Clears a segment of 252 memory locations. The entry point is 0EE0.
7. COMPEN: Multiplies the cosine and sine Fourier components obtained by executing the WFCON subroutine with the compensation factors of the matrix $[K]^{-1}$. The entry point is 0800.

6.5 Main Program

The main program consists two programs, viz, 1) program A written in 8080 ASSEMBLY language in the System 80/10, 2) program B written in BASIC in the graphics terminal hp2647A. The information transfer between the two programs is through a RS232C serial I/O interface. The flowcharts of the programs are given in Figs. 6.4(a) and 6.4(b). Program A is started by the system 80/10 Reset switch (this starts the program execution at location 0000) and B by the "RUN" command of hp2647A. The start of program B precedes the Reset operation i.e. the start of program A. The flowcharts are self explanatory. The features of program B are:

1. An output message "SYSTEM READY. TYPE 1 TO Start, 0 To TERMINATE" is displayed.
2. A message in the form of one ASCII character is typed on the keyboard of 2647A as the respone to the query in (1).
3. After computing the Walsh coefficients, an output message "SEEECTION OF SPECTRUM. TYPE 'F' FOR FOURIER SPECTRUM. 'W' FOR WALSH SPECTRUM" is displayed. Depending on the response, either Walsh or Fourier coefficients are outputted.
4. In the case of Walsh coefficients, a table of cal and sal coefficients values in volts with the associated sequency numbers are given. Also plots of cal coefficient

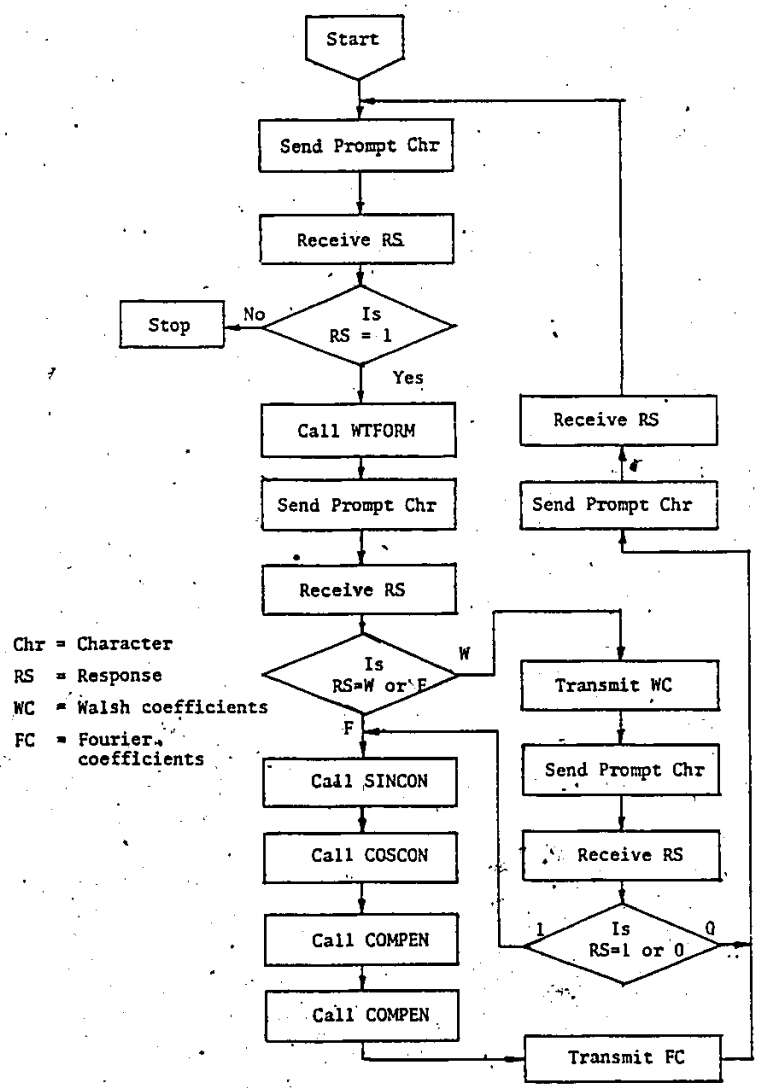
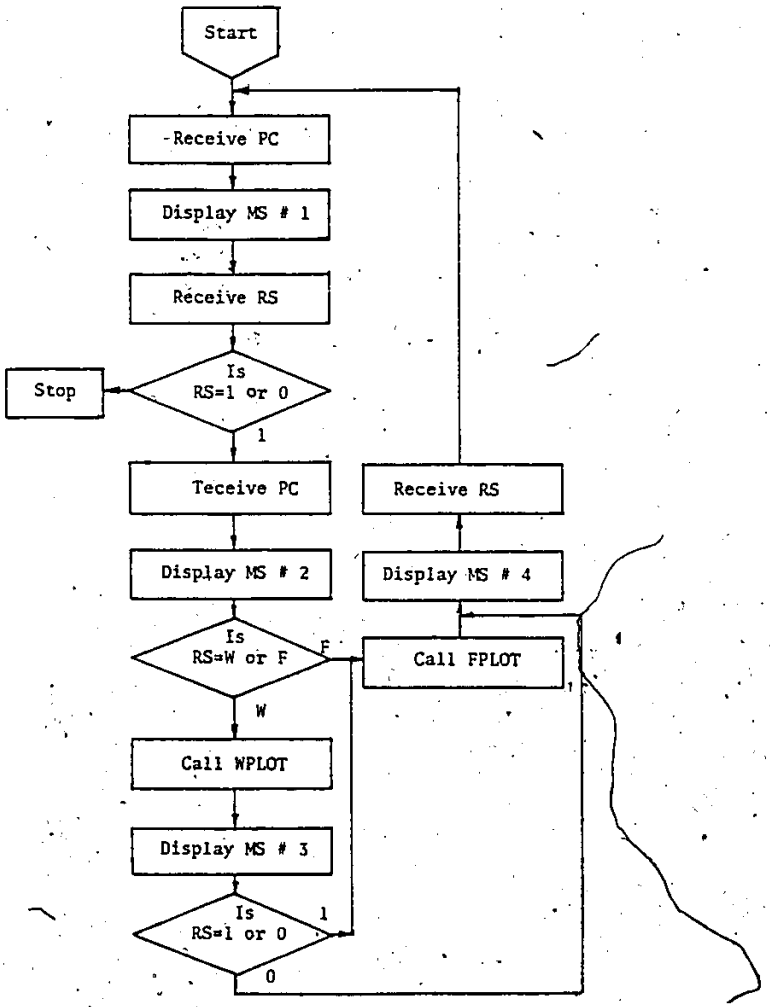


Fig. 6.4(a) Flowchart of main Walsh spectral analyser program in Intel 8080 Assembly language



PC = Promt Character MS = Measage
RS = Response
MS # 1: "SYSTEM READY. TYPE 1 TO START 0 TO TERMINATE"
MS # 2: "SELECTION OF SPECTRUM. TYPE W FOR WALSH COEFFICIENTS.
F FOR FOURIER COEFFICIENTS"
MS # 3: "TYPE 1 IF FOURIER COEFFICIENTS ARE NEEDED"
MS # 4: "TYPE 1 TO CONTINUE"

Fig. 6.4(b) Flowchart of BASIC program for hp 2647A graphic terminal

values vs sequency and sal coefficients values vs sequency are displayed. The results obtained for a sinusoidal voltage with a peak to peak voltage of 4 volts are illustrated in Table 6.3.

In the case of Fourier components, a table of magnitude in volts, phase in degrees and the frequency (Harmonic number) are given along with a plot of magnitude vs frequency. The results of Fourier analysis for the same sinusoidal signals are illustrated in Table 6.4.

5. After outputting the results as described above, a message "Type 1 to Continue" is displayed. Upon receipt of a response, the system is ready for the analysis of the next input signal.

6.6 Computational Errors

6.6.1 Errors on Walsh Coefficients

The computational errors in the following discussions are expressed in terms of the LSB of the A/D converter.

The 8-bit input data obtained from the A/D converter is in 2's complement form with an accuracy of ±1 lsb. (This error can be considered to be uniformly distributed.) In computing Walsh coefficient B(i) [Eqn. (3.3)], 64 sampled signal values are used (either added or subtracted) and the final result is divided by 64. Hence a total error in the computed error is ±1 lsb. (The fractional part due to

Table 6.3: Sal and cal coefficients of a sinusoidal signal 4 volts peak to peak

sequency	coefficients		sequency	coefficients	
	sal	cal		sal	cal
1	-2.4396	-0.6720	2	+0.0262	+0.0128
3	+1.0236	-0.2753	4	+0.0177	+0.0031
5	+0.2032	+0.0568	6	-0.0067	+0.0092
7	+0.4974	-0.1276	8	-0.0006	+0.0128
9	+0.0446	+0.0153	10	-0.0067	-0.0006
11	-0.0226	+0.0116	12	-0.0055	-0.0006
13	+0.0995	+0.0214	14	-0.0055	-0.0055
15	+0.2472	-0.0604	16	+0.0055	+0.0031
17	+0.0153	+0.0104	18	+0.0031	+0.0043
19	+0.0018	+0.0018	20	-0.0006	-0.0031
21	-0.0006	+0.0043	22	-0.0079	-0.0018
23	-0.0043	-0.0043	24	+0.0079	+0.0067
25	+0.0262	+0.0067	26	-0.0031	-0.0018
27	-0.0067	+0.0031	28	-0.0018	+0.0006
29	+0.0446	+0.0153	30	+0.0055	+0.0006
31	+0.1239	-0.0323	32	+0.0018	

The level crossing detector was adjusted so that the synchronizing pulse for frequency multiplication was displaced by 13 degrees.

Table 6.4: Fourier amplitude and phase spectrum of the signal of Table 6.3

Frequency	Amplitude	Phase	Frequency	Amplitude	Phase
1	+3.9702	+13.0789	2	-	-
3	-	-	4	-	-
5	-	-	6	-	-
7	-	-	8	-	-
9	-	-	10	-	-
11	-	-	12	-	-
13	-	-	14	-	-
15	-	-	16	-	-
17	-	-	18	-	-
19	-	-	20	-	-
21	-	-	22	-	-
23	-	-	24	-	-
25	-	-	26	-	-
27	-	-	28	-	-
29	-	-	30	-	-
31	-	-	32	-	-

- denotes zero

division by 64 is not neglected.)

6.6.2 Errors in Fourier Coefficients

Each Fourier coefficient is obtained by multiplying a maximum of 16 Walsh coefficients with the appropriate conversion matrix elements see Eqn. [(5.32)]. The conversion matrix elements are represented with a precision of nine fractional bits. After the conversion process, the fractional part is discarded, resulting in a maximum error of -1 lsb, in addition to errors due to finite precision of the conversion elements. The latter errors are negligible when compared with the truncation error. Finally the computed Fourier coefficient is multiplied with the compensation element and the fractional part is ignored, introducing an additional error of -1 lsb.

Hence the combined maximum error is -2 lsb in a computed Fourier coefficient. This error can be made $\pm$1 lsb by rounding off the fractional part rather than truncation.

CHAPTER 7

SUMMARY AND CONCLUSIONS

Walsh Spectral Analysers are classified into direct and transform types. Direct type instruments require N^2 arithmetic operations and transform type use $N\log_2 N$ operations to obtain N coefficients from a given data length N. The former is costlier in terms of hardware and can be justified in situations where all the N coefficients are needed immediately after the Nth sample.

A serial processor using long shift registers implementing a Fast Walsh Hadamard Transform is suggested. The processor is faster than an earlier design used by Geadah and Corinthios [17] by 1.5 times. An improved method to adapt the above design to obtain coefficients in dyadic and sequency orderings is given. The method consists of storing the incoming data in a permuted manner in an auxiliary memory and retrieving the data sequentially from the memory. The data is sent to the serial processor for further processing. The permutation operation involves a binary to Gray code conversion followed by bit-reversal operation for sequency ordered coefficients and a bit-reversal operation for dyadic ordered coefficients.

This scheme avoids the different permutation operations between the iterations of the FWHT used in the design of Geadah and Corinthios [17]. It also results in identical stages, when a pipeline structure is used for higher input data rates. These processors can accept input data rates of the order of 1 MHz for digital devices available at present.

Microprocessors provide an alternative in realizing a Walsh Spectral Analyser. This is lower in cost and Walsh to Fourier conversion can be implemented through software. An instrument based on an off-the-shelf Intel System 80/10 is developed. It consists of a cabinet containing an 8080-based single board computer SBC 80/10, a card cage with ready-wired bus and power supply lines for up to three additional boards and a power supply. Only one additional specially-designed board is used. It contains circuits for trigger level adjustment, A/D conversion, 1K of read-write memory, a frequency multiplier module (FMM) and a direct memory access (DMA) controller.

The FMM circuit generates 64 equally-spaced pulses within one period of the input signal, so that the data window spans one cycle. The leakage error in the Fourier spectrum measurement is made negligibly small. Another feature of the circuit is that it does not require a high-speed clock for its proper functioning. The signal frequency may lie anywhere within the range 0.2 Hz to 10 Hz

without any range switching. A measurement uses two cycles of input signal, the first being used for frequency determination and the second cycle for data acquisition. This is followed by processing of 24 msec duration, to yield the Walsh spectrum.

Given the Walsh coefficients of the signal, the corresponding Fourier coefficients are obtained by way of a matrix multiplication so that both spectra are available. The conversion process takes about 1 sec, most of the time being devoted to software multiplication.

The computed results (Walsh and Fourier components) are transmitted to a Hewlett Packard graphical terminal type 2647A with processing capability for displaying numerical and graphic data.

The present instrument is based on an eight bit microprocessor, accepting digital data in 8 bit, two's complement representation. This limits the dynamic range of the input signal to 42 dB. For better dynamic range, an A/D converter of 12 bits should be used and the use of one of the very recent 16 bit microprocessors is desirable to simplify the software and to reduce the computation time.

The FMM circuit developed in the present work may be extended to a higher speed of operation by incorporating multiple D/A converters for fine tuning adjustment, each D/A to be used in a round robin fashion.

REFERENCES

[1] H.C. Andrews, A.G. Tescher and R.P. Kruger, "Image Processing by Digital computer"; IEEE Spectrum, Vol. 9, pp. 20-32, July 1972.

[2] T. Fukinuki and M. Miyata, "Interframe Image Coding by Cascaded Hadamard Transforms", IEEE Trans. Communications, Vol. COM-21, pp. 175-180, March 1973.

[3] S.J. Campanella and G.S. Robinson, "A Comparison of Orthogonal Transformations for Digital Speech Processing", IEEE Trans. Communications, Vol. COM-19, pp. 1045-1050, June 1973.

[4] Proc. 1970-1974 Sym. Applications of Walsh Functions, Washington, D.C.

[5] H.C. Andrews, Introduction to Mathematical Techniques in Pattern Recognition, John Wiley and Sons Inc., 1972.

[6] H.F. Harmuth, Transmission of Information by Orthogonal Functions, Springer-Verlag, New York, 1972.

[7] N. Ahmed and K.R. Rao, Orthogonal Transforms for Digital Signal Processing, Springer-Verlag, New York, 1975.

[8] L. Franks, Signal Theory, Prentice-Hall Inc., Englewood Cliffs, N.J., 1969.

[9] K.H. Seimens and R. Kitai, "Digital Walsh-Fourier Analysis of Periodic waveforms", IEEE Trans. Instrum. and Measur., Vol. IM-18, pp. 316-321, Dec. 1969.

[10] B.G. Batchelor, "Circular Pictures, Their Digitization and Processing", IEE Journal on Computer and Digital Techniques, Vol. 1, No. 4, pp. 179-189, October 1978.

[11] B. Szabados and E.F. Hill, "On-line Phase Measurements of Power System Harmonics", IEEE Trans. Instrum. and Meas., Vol. IM-26, pp. 170-176, June 1977.

[12] R. Kitai, "On-line Measurement of Power System Harmonic Magnitude and Phase", IEEE Trans. Instrum. and Meas., Vol. IM-27, pp. 79-81, March 1978.

[13] R. Kitai, "Real-Time Walsh Spectrum Analysis", Proc. of the IEEE Conf. on Decision and Control, New Orleans, Louisiana, pp. 503-505, Dec. 1972.

[14] Y. Tanda and H. Sano, "A Hybrid Walsh Transform Analyser", Systems, Computers and Controls, Vol. 7, No. 1, pp. 80-89, Jan.-Feb. 1976.

[15] M.R. Ashouri and A.G. Constantinides, "A Pipeline Fast Walsh Fourier Transform", Proc. IEEE Inter. Conf. on ASSP, Hartford, pp. 515-518, May 1977.

[16] A.R. Elliot and Y.F. Shum, "A Parallel Array Hardware Implementation of the Fast Hadamard and Walsh Transform", Proc. Sym. Walsh Functions, pp. 181-183, March 1972.

[17] Y.A. Geadah and M.J.G. Corinthios, "Natural, Dyadic and Sequency order Algorithms and Processors for Walsh-Hadamard Transform", IEEE Trans. Computers, Vol. C-26, pp. 435-442, May 1977.

[18] K. Muniappan and R. Kitai, "A Micro-Computer based Walsh-Fourier Spectral Analyser for Periodic Signals", presented at 1979 Electrical and Electronic measurement and Test Instrument Conf. Ottawa, May 15-17, 1979. Accepted for publication in IEEE Trans. Instrum. and Meas., Dec. 1979.

[19] S.D. Stearns, Digital Signal Analysis, Hayden Inc., New Jersey, 1975, Chapter 7.

[20] B.A. Gimmel, "A Digital Method of Frequency Multiplication and its Application to Numeric Spectrum Analysis of Periodic Signals", IEEE Trans. on Instrum. and Meas., Vol. IM-26, pp. 181-183, June 1977.

[21] K. Muniappan and R. Kitai, "Improved Frequency Multiplier", Digest of Conf. on Precision Electromagnetic Measurements, pp. 81-83, June 1978.

[22] R. Kitai, "Walsh-to-Fourier Spectral Conversion for Periodic Waves", IEEE Trans. Electromagnetic Compatibility, pp. 266-269, Nov. 1975.

[23] Y. Tadokoro and T. Higuchi, "Discrete Fourier Transform computation via the Walsh Transform", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 3, pp. 236-240, June 1978.

[24] L.R. Rabiner and B. Gold, Theory and Applications of Digital Signal Processing, Prentice-Hall, New Jersey, 1975.

[25] H.L. Gorginsky and G.A. Works, "A Pipeline Fast Fourier Transform", IEEE Trans. Computers, Vol. C-19, No. 11, pp. 1015-1019, Nov. 1970.

[26] M.J. Corinthios, "The Design of a Class of Fast Fourier Transform Computers", IEEE Trans. Computers, Vol. C-20, pp. 617-623, June 1971.

[27] Spectrum Analysis - Theory, Implementations and Applications, Rockland System Corporation, 1977.

[28] N.A. Pendergrass and J.S. Farnhoch, "A High-resolution, Low Frequency Spectrum Analyser", Hewlett-Packard Journal, pp. 2-13, Sept. 1978.

[29] J.L. Walsh, "A Closed Set of Normal Orthogonal Functions", Amer. J. Math., Vol. 45, pp. 5-24, 1923.

[30] H.F. Harmuth, "Generalised Concepts of Frequency and some Applications", IEEE Trans. Inf. Theory, Vol. IT-14, pp. 375-382, May 1968.

[31] W.K. Pratt, J. Kane and H.C. Andrews, "Hadamard Transform Image Coding", Proc. of the IEEE, Vol. 57, pp. 58-68, January 1969.

[32] C.K. Yuen, "Walsh Functions and Gray Code", Proc. Sym. Appl. Walsh Func., pp. 68-73, 1971.

[33] C.K. Yuen, "Walsh Function Generation using Gray Code", Proc. Symp. Appl. Walsh Func., pp. 284-289, 1973.

[34] P.M. Besslich, "Walsh Function Generator for Minimum Orthogonality Error", IEEE Trans. EMC-15, pp. 177-180, Nov. 1973.

[35] T.S. Durrani and Nightingale, J.M., "Sequential Generation of Orthogonal Functions", Electronics Letters, Vol. 7, pp. 377-380, July 1971.

[36] R. Kitai and K.H. Siemens, "A Hazard Free Walsh Function Generator", IEEE Trans. Instrum. and Meas., Vol. IM-21, pp. 80-83, Feb. 1972.

[37] H.S. Stone, Discrete Mathematical Structures and Their Applications, Science Research Associates Inc., Chicago, 1973.

[38] R.B. Lackey and D. Meltzer, "A Simplified Definition of Wlash Functions", IEEE Trans. Computers, Vol. C-20, pp. 211-213, Feb. 1971.

[39] J.M. Carl, "Comments on a Simplified Definition of Walsh Functions", IEEE Trans. Computers, Vol. C-20, pp. 1617, Dec. 1971.

[40] A.C. Davies, "On the Definition and Generation of Walsh Functions", IEEE Trans. Computers, Vol. C-21, pp. 187-189, Feb. 1972.

[41] R. Kitai and K.H. Siemens, "Comment on a Simplified Definition of Walsh Functions", IEEE Trans. Computers, Vol. C-21, pp. 512, May 1972.

[42] H.F. Harmuth, "Application of Walsh Functions in Communication", IEEE Spectrum, pp. 82-91, Nov. 1969.

[43] H.C. Andrews, Computer Techniques in Image Processing, Academic Press, New York, 1970.

[44] C.K. Yuen, "Remarks on the Orderings of Walsh Functions", IEEE Trans. Computers, Vol. C-21, pp. 1452, Dec. 1972.

[45] C.K. Yuen, "Upper Bounds on Walsh Transforms", IEEE Trans. Computers, Vol. C-21, pp. 1273-1280, Dec. 1972.

[46] H. Gethoffer, "Sequency Analysis Using Correlation and Convolution", Proc. Symp. Appl. Walsh Func., pp. 118-123, April 1971.

[47] S.G. Knauer, "Real-time Video Compression Algorithm for Hadamard Transform Processing", IEEE Trans. Electromagnetic Compatibility, Vol. EMC-18, pp. 28-36, Feb. 1976.

[48] D.A. Gaubatz and R. Kitai, "Progammable Walsh Function Generator", IEEE Trans. Electromagnetic Compatibility, Vol. EMC-15, pp. 177-180, Nov. 1973.

[50] H. Schreiber and G.F. Sandy, Applications of Walsh Functions and Sequency Theory, Chapter 15, IEEE Press, New York, 1974.

[51] D. Hein and N. Ahmed, "On a Real-Time Walsh Hadamard Cosine Transform Image Processor", IEEE Trans. Electromagnetic Compatibility, Vol. EMC-18, pp. 28-36, Feb. 1976.

[52] Y.Y. Shum and A.R. Elliott, "Computation of the Fast Hadamard Transform", Proc. Symp. Appl. Walsh Func., pp. 177-180, March 1972, Washington, D.C.

[53] J.W. Manz, "A Sequency-ordered Fast Walsh Transform", IEEE Trans. Audio and Electroacoustics, AU-20, pp. 204-205, August 1972.

[54] R.J. Polge, B.K. Bhagavan and J.M. Carswell, "Fast Computational Algorithms for Bit Reversal", IEEE Trans. Computers, Vol. C-23, pp. 1-8, Jan. 1974.

[55] D. Bhagt and H.W. Mergler, "A High Performance Microprocessor-Based FFT Processor", IEEE Trans. Indus. Electronics and Control Instrum., Vol. IECI-25, pp. 102-107, May 1978.

[56] G. Berauer, "Fast 'In Place' Computation of the Discrete Walsh Transform in Sequency Order", Proc. Symp. Appl. Walsh Func., pp. 272-279, 1972, Washington, D.C.

[57] System 80/10 Microcomputer Hardware Reference Manual, Intel Corp., California, 1976.

[58] Intel 8080 Microcomputer Systems User's Manual, Intel Corp., California, Sept. 1976.

[59] User's Manual - Intelligent Graphics 2647, Hewlett Packard, USA, 1978.

[60] K.A. Krishnamurthy, et al., "A Simple Frequency Multiplier", Int. J. Electronics, Vol. 43, No. 2, pp. 201-205, Aug. 1977.

[61] S. Seth, et al., "A Low Coast Pulse Frequency Multiplier", Proc. IEEE, Vol. 66, No. 11, pp. 1578-1580, Nov. 1978.

[62] I. Hulley, "Versatile Rate Multiplier", Applied Ideas, Electronic Engineering, pp. 28, Nov. 1978.

[63] D. Horelick and B. Bertolucci, "A New Type of Digital Frequency Multiplier", Proc. IEEE, Vol. 63, pp. 1365-1366, Sept. 1975.

[64] S.K. Sarna, R. Kitai, K. Muniappan, et al., "Gastro-duodenal Coordination: A Computer Analysis", Proc. 6th International Symp. on Gastric Motility, Edinburgh, Sept. 1977.

[65] B.A. Gimmel, "An Improved Type of Digital Frequency Multiplier", Proc. IEEE, Vol. 65, No. 5, pp. 815, May 1977.

[66] "Model 3505J and 3507J Fast Slewing Operational Amplifier", Specification Notes, Burr Brown Corp., Arizona, June 1974.

[67] "Dual Power Supply for Computer Interface Model 546", Specification Notes, Burr Brown Corp., Arizona, Nov. 1971.

[68] MC8-85™ User's Manual, Section 4, pp. 4.51-4.67, Intel Corp., 1978.

[69] User's Manual - Intelligent Graphics 2647, Hewlett Packard Company, USA, 1978.

[70] N.M. Blachman, "Spectral Analysis with Sinusoids and Walsh Functions", IEEE Trans. Aerosp. Electron. Sys., Vol. AES-7, pp. 900-905, Sept. 1971.

[71] N.M. Blachman, "Sinusoids versus Walsh Functions", Proc. IEEE, Vol. 62, pp. 346-354, March 1974.

[72] R.F. Abramson, "The Sinc and Cosinc Transform", IEEE Trans. Electromagnetic Compatibility, Vol. EMC-19, pp. 88-94, May 1977.

[73] K.H. Siemens, Walsh Spectral Analysis, Ph.D. Thesis, McMaster University, Hamilton, Ontario, Canada, Chapter 5, June 1972.

[74] R. Kitai, I. Renyi and F. Vajda, "Using a Micro-processor in a Walsh-Fourier Spectral Analyser", Computer, pp. 27-31, April 1976.

[75] P.N. Bansod and R. Kitai, "Microprocessor Application in Walsh-Fourier Spectral Conversion", Proc. IEEE International Symposium on Electromagnetic Compatibility, Washington D.C., pp. 272-277, 1977.

[76] W.M. Doran, The Design of Digital Walsh-Fourier Converter, M.Eng. Thesis, McMaster University, Hamilton, Ontario, Canada, 1972.

[77] J.W. Horton, "Walsh Functions for Digital Impedence Relaying of Power Lines", IBM J. Res. Develop., pp. 530-547, Nov. 1976.

[78] K.H. Siemens and R. Kitai, "A Non-recursive Equation for Fourier Transform of Walsh Function", IEEE Trans. on Electromagnetic Compatibility, Vol. EMC-15, No. 2, pp. 81-83, May 1973.

[79] K. Muniappan, Computer Analysis of Gastroduodenal Electrical and Mechanical Signals, M.Eng. Thesis, McMaster University, Hamilton, Canada.

[80] A.V. Oppenheim and C.J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform", Proc. IEEE, Vol. 60, No. 8, pp. 957-976, August 1972.

[81] P.D. Welsh, "A Fixed-point Fast Fourier Transform Error Analysis", IEEE Trans. Audio Electroacoust., Vol. AU-17, pp. 151-157, June 1969.

[82] M. Sundaramurthy and V. Umapathi Reddy, "Some Results in Fixed Point FFT Error Analysis", IEEE Trans. on Computer, Vol. C-26, No. 3, pp. 305-308, March 1977.

[83] R.C. Singleton, "An Algorithm for Computing the Mixed Radix Fast Fourier Transform", IEEE Trans. Audio Electroacoust., Vol. AU-17, pp. 93-105, June 1969.

[84] H. Silverman, "An Introduction to Programming the Winograd Fourier Transform Algorithm", IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-25, pp. 152-165, April 1977.

[85] L.R. Morris, "A Comparative Study of Time Efficient FFT and WFTA Programs for General Purpose Computers", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-26, No. 2, pp. 141-150, April 1978.

[86] R.W. Patterson and J.H. McClellan, "Fixed-Point Error Analysis of Winograd Fourier Transform Algorithms", IEEE Trans. on Acoust. Speech, Signal Processing, Vol. ASSP-26, No. 5, pp. 447-455, 1978.

[87] A.M. Despain, "Very Fast Fourier Transform Algorithms Hardware for Implementation", IEEE Trans. on Computer, Vol. C-28, No. 5, pp. 333-341, May 1979.

[88] E.L. Braun, Digital Computer Design, Academic Press, New York, Chapter 6, 1973.

[89] K.W. Henderson, "Some Notes on the Walsh Functions", IEEE Trans. Computers, Vol. EC-13, pp. 50-51, Feb. 1964.