A new algorithm for calculation of the discrete Hartley transform

Mukesh Sharma
Lehigh University

A New Algorithm for calculation of the Discrete Hartley Transform

by

Mukesh Sharma

A Thesis

Presented to the Graduate Commitee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

1989

This Thesis is accepted and approved in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering.

SEPT 18, 1989
(date)

[illegible]
Professor in Charge

L. J. Varvein
CSEE Department Chairperson

Acknowledgements

I am deeply indebted to Dr. M. D. Wagh for both his involved guidance and the numerous suggestions he made, without which parts of this thesis would have been incomplete. In addition, valuable insight was provided by friends and fellow students, at Lehigh University and I would like to thank them all. I would also like to thank the CSEE department at Lehigh University for the financial Support that was provided during my stay here.

Table of Contents

List of Figures

Abstract

Group Theory has played a very significant role in designing fast algorithms for digital signal processing. This thesis develops a new algorithm to compute the Hartley Transform based on group theoretic concepts. The proposed algorithm identifies multidimensional cyclic structures within the Hartley transform kernel by using the properties of groups formed by the row and column indices. It then uses the readily available convolution algorithms to evaluate the products of such submatrices and vectors properly prepared from the input sequences. A combination of such product vectors gives the desired transform. Thus the algorithm is inherantly divided into three stages, the preprocessing stage to massage the input sequence, the convolution stage to carry out the actual product of submatrices and the postprocessing stage to combine the product vectors. The proposes algorithm has a better computational complexity than other algorithms available in the literature, is more universal (i.e., applicable to all lengths), and has a modular structure, implying its suitability for parallel implementation.

Chapter 1
INTRODUCTION

1.1. The Central Goal.

Because of the increasing availability of digital computational engines, much of the signal processing today, is done in discrete digital form. Linear transformations such as the Fourier, Hartley and Hadamard transforms have often been used successfully in various digital signal processing applications. However, in order that the transform technique satisfy todays demands for real time performance, one needs to decrease the computational complexity of the transform algorithm. This requirement, for example, forced researchers to improve the algorithm for discrete Fourier transform through Goertzel algorithm [1], fast Fourier transform algorithm [2], the chirp-z transform [3] and Winograd Fourier transform algorithm [4].

The Hartley transform defined in 1942 by R. V. L. Hartley [5] has many properties superior to Fourier transform. It is symmetric with respect to the transformation and kernel, the transformation matrix is real and it performs better than Fourier transform in many applications such as the spectral analysis. However, its discrete version, the discrete Hartley transform was derived only in 1983 [6]. Consequently, the fast algorithms for discrete Hartley transforms have begun appearing in literature only recently [7-11]. This thesis derives a new algorithm for Hartley transform based on fast cyclic convolutions. The performance of our algorithm is superior in terms of number of operations to other algorithms available in literature. In addition, our algorithm can be easily partitioned for parallel execution and is therefore well suited either to general purpose parallel architectures or to specially designed digital systems.

1.2. Algorithms for Hartley Transform.

The Discrete Hartley Transform (DHT) of an N-point sequence is defined as

$$X(s) = \sum_{r=0}^{N-1} M(s, r)\, x(r), \qquad s = 0, 1, 2, \dots, N-1,$$

where $M(s, r) = \mathrm{cas}(\frac{2\pi rs}{N}) = \cos(\frac{2\pi rs}{N}) + \sin(\frac{2\pi rs}{N})$.

From the definition, one can calculate this N-point Hartley transform in N^2 real multiplications and $N(N-1)$ additions. When N is a power of two, one can use techniques similar to the fast Fourier transform to calculate the Hartley transform.[1] The decimation in time algorithm given by Bracewell [7] is based on recursively expressing a N point DHT in two (N/2) point DHTs. In particular, he uses the relation

$$X(i) = \frac{1}{2} [X_e(i \bmod (N/2)) + X_o(i \bmod (N/2)) \cos (2\pi i/N)$$
$$+ X_o(i \bmod (N/2))\sin(2\pi i/N)], \qquad i = 0, 1, \dots, N-1,$$

where, X_e and X_o are the DHTs of the (N/2) point sequences formed by the even and odd components of x(i) respectively. Applying this same relation to these smaller length DHTs, and continuing down to DHT's of 2 point sequences, one can obtain a fast DHT algorithm. The algorithm can thus be expressed in vector notation as

$$X = (1/N)\, L_{\log_2 N} \cdots L_3 L_2 L_1 P\, x,$$

where, each L_t matrix represents a butterfly stage, P is a permutation matrix and x is the given vector. The computational complexity of the resultant algorithm is $N\log_2 N - 2N + 2$ multiplications and $2N\log_2 N - 2N + 2$ additions. If instead one uses the decimation in frequency [9], one can base the computation on recursive equations:

$$X(2i) = X_1(i), \quad X(2i+1) = X_2(i) \cos(2\pi i/N) + X_2(N/2 - i) \sin(2\pi i/N),$$

where, X_1 and X_2 are the DHTs of sequences whose j-th components are $x(j) + x(N/2 + j)$ and $x(j) - x(N/2 - j)$ respectively. Thus a N point DHT is again calculated through two (N/2) point DHTs. Recursive application of these relations gives the total computational complexity to be $N\log_2 N - 3N + 4$ multiplications and $1.5N\log_2 N - 1.5N$

[1]These algorithms are referred to as the fast Hartley transform algorithms.

+ 2 additions. Thus in both the versions of the fast Hartley transform, one finds that the computational complexity is $O(N\log_2 N)$.

Hsu and Wu [8] have observed that the product of the Hartley transform matrix and the Walsh-Hadamard transform matrix has a simple block diagonal structure. They call this resultant matrix by the H-transform and use that in the computing of Hartley transform. In particular, a sequence could be first transformed by the (inverse) Walsh-Hadamard transform and then through the H-transorm to yield the desired Hartley transform. The block diagonal structure of the H-transform makes this method attractive for parallel and VLSI implementations.

Several other methods of calculating the discrete Hartley transform have also been proposed [10] by Sorensen et. al. These methods are adapted from various Fourier transform algorithms to suit Hartley transform. They include such wide variety as the split radix, prime factor, radix 4 and Winograd algorithms and are generally computationally superior to the fast Hartley transforms [7, 9].

1.3. Principle and Structure of a New algorithm.

This thesis proposes a new algorithm for computing the Discrete Hartley transform by exploiting the structure of its kernel matrix. In particular, it has been shown that the matrix can be partitioned such that each partition submatrix looks like a multidimensional cyclic convolution matrix[2]. The efficient algorithms for cyclic convolutions [12] can then be used to calculate these smaller submatrices. Thus the new algorithm is a three stage procedure. The most important of these is the second stage in

[2]This statement is not exactly true. In case of certain effective convolution lengths, the submatrix is a Kronecker product of a Toeplitz matrix and a multidimentional cyclic matrix.

which the convolutions are performed. The entire algorithm structure depends upon the kinds of convolutions that are to be carried out in this stage. The first stage sets up the inputs to the various convolutions. This first stage, referred to as the preprocessing stage in this work, combines the input data points using additions and subtractions only. The last stage of the algorithm combines the appropriate convolution outputs to create the final transform vector. This stage, called the post-processing stage, is also purely additive. Thus the only multiplications involved in the entire algorithm are performed in the convolution stage.

The algorithm for discrete Hartley transform presented in this thesis is unique in many respects. It is the first time cyclic structures have been identified within the Hartley transform kernel. As has been amply demonstrated in this thesis, this has been possible only through extensive use of the ideas borrowed from group theory. All the three stages of the algorithm have regular and modular structures and thus lend themselves well to parallelisation. The complexity of the algorithm presented here is lower than other algorithms available in the literature [7-11].

1.4. Organization of the Thesis.

Chapter 2 of the thesis states and proves the underlying principles of the new algorithm. This chapter also presents the mathematical ideas, including the principles of group theory and the relation between groups and convolutions, which are crutial to understanding this work. As stated in Sec. 1.3, the algorithm has three stages. These are treated in Chapters 3, 4 and 5. Wherever necessary, these chapters include examples to illustrate the concepts and figures to explain the structure in the algorithm. Finally, Chapter 6 contains the conclusion and general discussion of the work presented.

Chapter 2
CYCLIC COMPONENTS OF HARTLEY TRANSFORM

2.1. Group table and convolution.

This chapter provides the mathematical background necessary to comprehend the following chapters as well as the description of the basic structure of the Hartley transformation. The mathematical foundation of understanding a linear transformation through group algebra are taken from an earlier work by Wagh and Ganesh [5], and interested readers may refer to it for further details. The description of the Hartley transform algorithm in terms of group algebra is described in this chapter has not appeared in earlier litrature.

This entire thesis is concerned with exploiting the structure in the linear transformation, the Hartley transform. In particular, we are interested in linear transformation matrices that mimic a group table.

Consider an $N \times N$ matrix defined by

$$M(i, j) = f((i.j) \bmod N), \qquad 1 \le i, j \le N-1. \qquad (2.1)$$

for arbitrary (complex) function $f : \mathbb{Z}_N \to \mathbb{C}$. An example of such a matrix is the Fourier matrix defined by

$$M(i, j) = e^{\sqrt{-1}\frac{2\pi ij}{N}} = \cos\left(\frac{2\pi ij}{N}\right) + \sqrt{-1}\,\sin\left(\frac{2\pi ij}{N}\right). \qquad (2.2)$$

Note that in (2.2) M(i, j) is a function of i.j. Futher since both the sine and cosine functions have periods of 2π, M(i, j) is a function of ij mod N. If the indices i and j form a cyclic group with the group operation of multiplication modulo N, one can represent the matrix as a group table. The operation of the matrix multiplying with a vector turns out in this case to be a convolution of the first row of the matrix and the vector. Following example demonstrates the structure of such a matrix which is clearly cyclic.

Example 2.1 : Let i, j $\in$ A(5) = { 1, 2, 3, 4 }. A(5) forms a cyclic group under the operation of multiplication modulo 5. If f(ij mod 5) describes the matrix entries, the matrix looks like

$$M = \begin{bmatrix} f(1) & f(2) & f(3) & f(4) \\ f(2) & f(4) & f(1) & f(3) \\ f(3) & f(1) & f(4) & f(2) \\ f(4) & f(3) & f(2) & f(1) \end{bmatrix}.$$

Notice that if the row and column numbers are placed in an ascending order, M does not resemble a cyclic matrix. However, if one orders the rows and columns using a genrator of the group, the matrix does become cyclic. In the above example 2 $\in$ A(5) has order 4. One can use the generator 2 to order the rows and columns. The new index order is 2^0, 2^1, 2^2, 2^3 (1, 2, 4, 3, since the operation is still modulo 5). M can now be seen to be transformed into

$$M' = \begin{bmatrix} f(1) & f(2) & f(4) & f(3) \\ f(2) & f(4) & f(3) & f(1) \\ f(4) & f(3) & f(1) & f(2) \\ f(3) & f(1) & f(2) & f(4) \end{bmatrix},$$

which is a 4×4 cyclic matrix and its product with a vector can be computed using cyclic convolution algorithms.

Two comments are in order at this point. First, the cyclic convolution is now computing the product $y' = M'.x'$ rather than the required product $y = Mx$. However, the row and column shufflings transforming M into M$'$ can be compensated by equivalent shufflings of the components of x and y to create x′ and y′ respectively. To illustrate this in case of the previous example, let the required computation be

$$\begin{bmatrix} f(1) & f(2) & f(3) & f(4) \\ f(2) & f(4) & f(1) & f(3) \\ f(3) & f(1) & f(4) & f(2) \\ f(4) & f(3) & f(2) & f(1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

By shuffling the columns of M (to the required order) and appropriately shuffling components of x, one gets

$$\begin{bmatrix} f(1) & f(2) & f(4) & f(3) \\ f(2) & f(4) & f(3) & f(1) \\ f(3) & f(1) & f(2) & f(4) \\ f(4) & f(3) & f(1) & f(2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_4 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

Finally, by shuffling the rows of the matrix and the components of y, one gets

$$\begin{bmatrix} f(1) & f(2) & f(4) & f(3) \\ f(2) & f(4) & f(3) & f(1) \\ f(4) & f(3) & f(1) & f(2) \\ f(3) & f(1) & f(2) & f(4) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_4 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_4 \\ y_3 \end{bmatrix},$$

the required cyclic convolution. Note that the shufflings of x and y vectors do not add to the computational complexity. Thus the ordering of row and column indices as per some group characteristics plays an important part in transforming the matrix to a cyclic matrix.

Secondly, the group formed by the row and column indices may not form a cyclic group. However, note that it is still an Abelian group (integer elements and the group operation of multiplication modulo N guarantes that). The fundamental theorem of groups says that it is isomorphic to the direct product of cyclic groups. By appropriate shuffling of rows and columns, the matrix could therefore be made to look like Kronecker product of cyclic matrices. Such matrix vector product can be evaluated through multidimensional cyclic convolutions. This concept is illustrated by example 2.2.

Example 2.2 : Let i, j $\in$ A(15). A(15) = { 1, 2, 4, 7, 8, 11, 13, 14 } = $C_4 \times C_2$. The matrix M(i, j) = f(ij mod 15) = cas(2πij/15) is shown in Fig. 2.1.

	1	2	4	7	8	11	13	14
1	1.32	1.41	0.89	-0.77	-1.19	-1.10	-0.07	0.51
2	-1.41	0.89	-1.19	0.51	1.32	-0.77	-1.10	-0.07
4	0.89	-1.19	1.32	-0.07	1.41	0.51	-0.77	-1.10
7	-0.77	0.51	-0.07	0.89	-1.10	1.41	1.32	-1.19
8	-1.19	1.32	1.41	-1.10	0.89	-0.07	0.51	-0.77
11	-1.10	-0.77	0.51	1.41	-0.07	1.32	-1.19	0.89
13	-0.07	-1.10	-0.77	1.32	0.51	-1.19	0.89	1.41
14	0.51	-0.07	-1.10	-1.19	-0.77	0.89	1.41	1.32

Fig 2.1. Matrix (M) for i, j $\in$ A(15).

Using generators 2 and 11 of C_4 and C_2 respectively, one gets C_4 = { 1, 2, 4, 8 } and C_2 = { 1, 11 }. Their direct product thus gives A(15) with the desired order of rows and columns of M as {1, 11, 2, 7, 4, 14, 8, 13 }. Note that this order transforms the matrix M to desired M$'$.

$$\begin{array}{c|cccccccc}
 & 1 & 11 & 2 & 7 & 4 & 14 & 8 & 13 \\ \hline
1 & 1.32 & -1.10 & 1.41 & -0.77 & 0.89 & 0.51 & -1.19 & -0.07 \\
11 & -1.10 & 1.32 & -0.77 & 1.41 & 0.51 & 0.89 & -0.07 & -1.19 \\
2 & 1.41 & -0.77 & 0.89 & 0.51 & -1.19 & -0.07 & 1.32 & -1.10 \\
7 & -0.77 & 1.41 & 0.51 & 0.89 & -0.07 & -1.19 & -1.10 & 1.32 \\
4 & 0.89 & 0.51 & -1.19 & -0.07 & 1.32 & -1.10 & 1.41 & -0.77 \\
14 & 0.51 & 0.89 & -0.07 & -1.19 & -1.10 & 1.32 & -0.77 & 1.41 \\
8 & -1.19 & -0.07 & 1.32 & -1.10 & 1.41 & -0.77 & 0.89 & 0.51 \\
13 & -0.07 & -1.19 & -1.10 & 1.32 & -0.77 & 1.41 & 0.51 & 0.89
\end{array}$$

Fig 2.2. Matrix M' obtained from M by row and column shuffling.

Finally, the cyclic structure in a matrix may have to be brought out by changing signs of appropriate rows and columns and correspondingly negating components of x and y vectors. This is illustrated in example 2.3.

Example 2.3 : Consider the following computation

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a & -b & c & d \\ -d & a & -b & -c \\ c & -d & a & b \\ b & -c & d & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

This product cannot be computed as stated using the techniques of cyclic convolution, since M is not a cyclic matrix. However, if one inputs $-x_2$ instead of x_2 and computes $-y_2$ instead of y_2, the computation can be made to look like

/

$$\begin{bmatrix} y_1 \\ -y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{bmatrix} \begin{bmatrix} x_1 \\ -x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

The matrix in this computation now has a cyclic structure and the product can be computed using numerous algorithms used for cyclic convolution.

2.2. Paritioning Hartley Transform in convolutions.

In most practical situtations, the indices defining the entire transformation matrix may not form a group. Instead, these indices may be partitioned into subsets such that each subset forms a group. thus the origanal matrix structure could be viewed as a containing many smaller convolutions. The procedure for computing the matrix-vector product would then consist of evaluating many smaller convolutions and the combining the their results. This procedure of partitioning the index sets into appropriate groups is at the heart of converting Hartley transform and is described in this section.

The Discrete Hartley Transform (DHT) of an N-point sequence is defined as

$$X(s) = \sum_{r=0}^{N-1} M(s, r)\, x(r), \qquad s = 0, 1, 2, \dots, N-1.$$

where $M(s, r) = \mathrm{cas}(\frac{2\pi rs}{N}) = \cos(\frac{2\pi rs}{N}) + \sin(\frac{2\pi rs}{N})$.

We calculate X(0) seperately because X(0) is mearly a sum of all x components. To compute other X(s) components the nonzero indices may be partitioned as follows.

Define *the primary partition index*, $d \in \mathbb{Z}$, such that $d \mid N$ and $1 \leq d < N$. Then one can form sets S_d such that

$$S_d = \{ i \in \mathbb{Z} \mid \gcd(i, N) = d \},\ 1 \le i \le N \}.$$

We now concentrate on computing X(s), $s \in S_d$. For $i \in S_d$, $\gcd(N, i) = d$ and $1 \le i <$ N gives

$$\gcd(N/d, i/d) = 1 \text{ and } 1 \le i/d < N/d.$$

Clearly, set of such integers (i/d), form the group A(N/d), the group of automorphisms of cyclic group of order N/d. Let the elements in A(N/d) be denoted by g_j, $0 \le j \le |A(N/d)|-1$ using the ordering of group elements as described in section 2.1. One then gets

$$S_d = \{ d.\, g_j \mid g_j \in A(N/d) \}.$$

Also $M(d.g_j, r)$ can be written after proper shuffling corresponding to a group ordering as

$$M(d.g_j, r) = \cos(\frac{2\pi r g_j}{N/d}) + \sin(\frac{2\pi r g_j}{N/d}).$$

Such $M(d.g_j, r)$ is periodic in r with period (N/d) because

$$\begin{aligned} M(d.g_j, r + l.N/d) &= \cos(\frac{2\pi(r + l.\, N/d)g_j}{N/d}) + \sin(\frac{2\pi(r + l.\, N/d)g_j}{N/d}). \\ &= \cos(2\pi l g_j + \frac{2\pi r g_j}{N/d}) + \sin(2\pi l g_j + \frac{2\pi r g_j}{N/d}). \\ &= \cos(\frac{2\pi r g_j}{N/d}) + \sin(\frac{2\pi r g_j}{N/d}). \\ &= M(dg_j, r)\ . \end{aligned} \tag{2.3}$$

Now from the definition of the Hartley Transform one gets

$$X(dg_j) = \sum_{r=0}^{N/d-1} \sum_{l=0}^{d-1} M(dg_j, r + l.\, N/d).\, x(r + l.\, N/d), \qquad g_j \in A(N/d).$$

Using eqn (2.3) one get

$$X(dg_j) = \sum_{r=0}^{N/d-1} \sum_{l=0}^{d-1} M(dg_j, r). \, x(r + l. \, N/d), \qquad g_j \in A(N/d).$$

Since $M(dg_j, r)$ is independent of l we can simplify the above as

$$X(dg_j) = \sum_{r=0}^{N/d-1} M(dg_j, r) \sum_{l=0}^{d-1} x(r + l. \, N/d), \qquad g_j \in A(N/d).$$

Let $\quad y_d(r) = \sum_{l=0}^{d-1} x(r + l. \, N/d), \qquad 0 \leq r \leq \frac{N}{d} - 1.$ (2.4)

then, $X(dg_j) = \sum_{r=0}^{N/d-1} M(dg_j, r) \; y_d(r), \qquad g_j \in A(N/d).$

Now define c such that $c \mid (N/d)$ as the *secondary partition index.* Define a set T_c as

$$T_c = \{ \, r \mid \gcd(r, N/d) = c \, \}, \;\; 1 \leq r < N/d.$$

Notice however that $\gcd(r/c, N/cd) = 1$ and $1 \leq (r/c) < (N/cd)$. The r/c values form A(N/cd) under the group elemrnt ordering of Sec 2.1. Thus T_c can be expressed as

$$T_c = \; \{ \, c.h_i \mid \; h_i \in A(N/cd) \, \}.$$

Therefore one can rewrite (2.4) as

$$X(dg_j) = \sum_{c \mid N/d} X_{d,c}\,(j), \tag{2.5}$$

where, $X_{d,c}(j) = \sum_{h_i \varepsilon \; A(N/cd)} M(dg_j, ch_i) \; y_d(h_i), \quad g_j \in A(N/d), \;\; 0 \leq j \leq \mid A(N/d) \mid - 1.$

We now show that only a few of $X_{d,c}(j)$ in (2.5) are distinct.

Since $M(dg_j, ch_i) = \; \text{cas}\,(\frac{g_j. \, h_i.2\pi}{N/cd}),$

$$M(d(x + m. \, N/cd), c.h_i) = \cos(\frac{2\pi(x+ m. \, N/cd)h_i}{N/cd}) + \sin(\frac{2\pi(x+ m. \, N/cd)h_i}{N/cd}).$$

$$= \cos(2\pi m h_i + \frac{2\pi x h_i}{N/cd}) + \sin(2\pi m h_i + \frac{2\pi x h_i}{N/cd}).$$

$$= \cos(\frac{2\pi x h_i}{N/cd}) + \sin(\frac{2\pi x h_i}{N/cd}).$$

$$= M(x.d,\ c.h_i).$$

Thus $M(d.g_j, c.h_i) = M(d(g_j \bmod \frac{N}{cd}), c.h_i)$.

Thus one needs to calculate $X_{d,c}(j)$ only at those j's for whom $g_j < N/cd$. Further, $\gcd(g_j, N/cd) = 1$ implies that the g_j's we are interested in belong to $A(N/cd)$ and the computation in (2.5) can be rewritten as

$$X_{d,c}(j) = \sum_{h_i \varepsilon A(N/cd)} \operatorname{cas}\,(\frac{g_j.\ h_i.2\pi}{N/cd}).y_d(h_i.c), \qquad g_j \in A(\frac{N}{cd})$$

It is now easy to show that the function $\operatorname{cas}\,(\frac{g_j.h_i.2\pi}{N/cd})$ can be expressed as $f(g_j \oplus h_i)$ where

$$g_j \oplus h_i = g_j.\ h_i \bmod \frac{N}{cd}$$

and $f(u) = \operatorname{cas}(\frac{u.2\pi}{N/cd})$. Note that $\oplus$ denotes the multiplication modulo N/cd, the group operation of $A(N/cd)$.

$$X_{d,c}(j) = \sum_{h_i \varepsilon A(N/cd)} f\,(g_j \oplus h_i)\ .\ y_d(h_i.\ c), \qquad g_j \in A(\frac{N}{cd}). \qquad (2.6)$$

As has been shown in sec 2.1, $X_{d,c}$ vector thus can be computed through a convolution over group $A(N/cd)$.

If $2 \mid (N/d)$, the structure in M can be further exploited because one can write

$$M(dg_j, r) = \operatorname{cas}\,(\frac{r.\ g_j\ \pi}{N/2d}) \text{ where } g_j \in A(\frac{N}{d}).$$

The function $M(dg_j, r)$ is now periodic with a smaller period N/2d because

$$M(dg_j, r + l.\ N/2d) = \cos(\frac{\pi(r + l.\ N/2d)g_j}{N/2d}) + \sin(\frac{\pi(r + l.\ N/2d)g_j}{N/2d}).$$

$$M(dg_j, r + l.\ N/2d) = \cos(\pi l g_j + \frac{\pi r g_j}{N/2d}) + \sin(\pi l g_j + \frac{\pi r g_j}{N/2d}).$$

$$M(dg_j, r + l.\ N/2d) = (-1)^{lg_j} [\cos(\frac{\pi r g_j}{N/2d}) + \sin(\frac{\pi r g_j}{N/2d})].$$

$$M(dg_j, r + l.\ N/2d) = (-1)^{lg_j} M(dg_j, r)\ .$$

This last step is due to equation $\cos(k\pi + \theta) = (-1)^k \cos(\theta)$ and $\sin(k\pi + \theta) = (-1)^k \sin(\theta)$. Since $g_j \in A(N/d)$ and N/d is even, g_j must be odd. Therfore

$$M(dg_j, r + l\,N/2d) = (-1)^l M(dg_j, r). \tag{2.7}$$

The expression for $X(dg_j)$ can now be written as

$$X(dg_j) = \sum_{r=0}^{N/2d-1} \sum_{l=0}^{2d-1} M(dg_j, r + l.\ \frac{N}{2d}).\ x(r + l.\ \frac{N}{2d}), \qquad g_j \in A(N/d),\ N/d \text{ even}.$$

Using (2.7) this can be simplified as

$$X(dg_j) = \sum_{r=0}^{N/2d-1} \sum_{l=0}^{2d-1} (-1)^l M(dg_j, r).\ x(r + l.\ \frac{N}{2d}), \qquad g_j \in A(N/d).$$

$$X(dg_j) = \sum_{r=0}^{N/2d-1} M(dg_j, r) \sum_{l=0}^{2d-1} (-1)^l\ x(r + l.\ \frac{N}{2d}), \qquad g_j \in A(N/d).$$

Or alternately

$$y_d(r) = \sum_{l=0}^{2d-1} (-1)^l\ x(r + l.\ \frac{N}{2d})$$

and $$X(dg_j) = \sum_{r=0}^{N/2d-1} M(dg_j, r)\ .\ y_d(r), \qquad g_j \in A(N/d). \tag{2.8}$$

As before, one can define the *secondary partition index* c such that $c \mid (N/2d)$ and form sets T_c as

$$T_c = \{ i \mid \gcd(i, N/2d) = c \},\ 1 \leq i <(N/2d) \}.$$

Clearly, in this case for $i \in T_c$, $(i/c) \in A(N/2cd)$ because $\gcd(i/c, N/2cd) = 1$ and $1 \leq (i/c) < N/2cd$. Therefore set T_c can also be described as

$$T_c = \{ c.h_i \mid h_i \in A(\frac{N}{2cd}) \},$$

where h_i is the i-th element of group $A(N/2cd)$ ordered as in Sec 2.1. The output vector can thus be written as

$$X(dg_j) = \sum_{c \mid N/2d} X_{d,c}(j),$$

where, $X_{d,c}(j) = \sum_{h_i \in T_c} M(dg_j, ch_i)\, y_d(h_i), \qquad g_j \in A(N/d). \qquad (2.9)$

In (2.8), $M(dg_j, ch_i) = \mathrm{cas}(\frac{g_j.h_i.\pi}{N/2cd})$. Depending upon the even or odd nature of $N/2cd$, this function has differnt periodicities. These are treated as the following two cases

<u>Case I : $2 \mid (N/2cd)$.</u> In this case one has

$$M(d(x + m.\frac{N}{2cd}), h_i.c) = \cos(\frac{\pi(x+ m.(N/2cd))h_i}{N/2cd}) + \sin(\frac{\pi(x+ m.(N/2cd))h_i}{N/2cd}).$$

$$M(d(x + m.\frac{N}{2cd}), h_i.c) = \cos(\pi m h_i + \frac{\pi x h_i}{N/2cd}) + \sin(\pi m h_i + \frac{\pi x h_i}{N/2cd}).$$

$$M(d(x + m.\frac{N}{2cd}), h_i.c) = (-1)^{mh_i} [\cos(\frac{\pi x h_i}{N/2cd}) + \sin(\frac{\pi x h_i}{N/2cd})].$$

$$= (-1)^{mh_i} M(x.d, h_i.c).$$

Now $h_i \in A(\frac{N}{2cd})$ and $\frac{N}{2cd}$ is even, h_i must be odd. Therfore one can write the above equation as

$$M(d(x + m.\frac{N}{2cd}), h_i.\, c) = (-1)^m\, M(d.x, h_i.c)$$

$$M((g_j + m.\frac{N}{2cd})d, h_i.\, c) = (-1)^m\, M(d(g_j \bmod \frac{N}{2cd}), h_i.c)$$

Thus one needs to calculate $X_{d,c}(j)$ for those j's for whom $g_j < N/2cd$. Further gcd $(g_j, \frac{N}{2cd}) = 1$ implies that $g_j \in A(\frac{N}{2cd})$. Therefore in this case,

$$X_{d,c}(j) = \sum_{h_i \varepsilon\ A(N/2cd)} \text{cas}\,(\frac{g_j.\, h_i.\pi}{N/2cd}) \,.\, y_d(h_i.\, c), \qquad g_j \in A(\frac{N}{2cd})$$

Note that the function $\text{cas}\,(\frac{g_j.h_j.\pi}{N/2cd})$ can be expressed as $f(g_j \oplus h_i)$ where $f(u) = \text{cas}(\frac{u.\pi}{N/2cd})$ and $h_i \oplus g_j = h_i.\, g_j \bmod (N/cd)$, the operation of the group $A(N/cd)$. However, this is a differnt group than the one containing h_i and g_j.

Using this interpretation, one can conclude that

$$X_{d,c}(j) = \sum_{h_i \varepsilon\ A(N/2cd)} f\,(g_j \oplus h_i) \,.\, y_d(h_i.c), \quad g_j \in A(\frac{N}{2cd}) \quad 2 \mid (N/2cd). \qquad (2.10)$$

Since h_i, $g_j \in A(N/2cd)$ and $\oplus$ is the operation of group $A(N/cd)$, this computation cannot be carried out as a convolution over a group. It will be shown in Chapter 4 that this is a Toeplitz computation.

<u>Case II : 2 does not divide N/2cd , but 2 | (N/cd).</u> In this case, as before, one can show

that

$$M(dg_j, ch_i) = M((g_j + m\, \frac{N}{cd}).d, h_i.c)$$

Thus one needs to calculate $X_{d,c}(j)$ only at those j's for whom $g_j < N/cd$. Also gcd$(g_j, \frac{N}{cd}) = 1$. This implies that $g_j \in A(N/cd)$. Therefore

$$X_{d,c}(j) = \sum_{h_i \varepsilon\ A(N/2cd)} \text{cas}\,(\frac{g_j.\, h_i.\, 2\pi}{N/cd}).\, y_d(h_i.c), \quad g_j \in A(\frac{N}{cd}). \qquad (2.11)$$

Note that now $h_i \in A(N/2cd)$ while $g_j \in A(N/cd)$. Since 2 does not divide N/2cd, let Q represent the odd integer N/2cd. Then, $h_i \in A(Q)$ and $g_j \in A(2Q)$. However $A(2Q) \equiv A(2) \times A(Q)$, or $A(2Q) \equiv A(Q)$. Thus A(2Q) and A(Q) are isomorphic and g_j has an image in A(Q) under this isomorphism. However, even though h_i, $g_j \in A(Q)$, the cas function can be expressed as $f(g_j \oplus h_i)$ where $\oplus$ is the operation of multiplication modulo 2Q (=N/cd). Note that in this case, the submatrix calculating $X_{d,c}(j)$ cannot be viewed as a convolution matrix. It will be shown later in chapter 4 that it is part of a larger group table and in fact, is a Toeplitz matrix.

2.3. Convolutions involved in DHT.

In order to illustrate the ideas presented in section 2.2, consider the calculation of DHT of order 12 shown in Fig 2.3.

$$
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{bmatrix}
=
\begin{bmatrix}
1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
1.00 & 1.37 & 1.37 & 1.00 & 0.37 & -0.37 & -1.00 & -1.37 & -1.37 & -1.00 & -0.37 & 0.37 \\
1.00 & 1.37 & 0.37 & -1.00 & -1.37 & -0.37 & 1.00 & 1.37 & 0.37 & -1.00 & -1.37 & -0.37 \\
1.00 & 1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 \\
1.00 & 0.37 & -1.37 & 1.00 & 0.37 & -1.37 & 1.00 & 0.37 & -1.37 & 1.00 & 0.37 & -1.37 \\
1.00 & -0.37 & -0.37 & 1.00 & -1.37 & 1.37 & -1.00 & 0.37 & 0.37 & -1.00 & 1.37 & -1.37 \\
1.00 & -1.00 & 1.00 & -1.00 & 1.00 & -1.00 & 1.00 & -1.00 & 1.00 & -1.00 & 1.00 & -1.00 \\
1.00 & -1.37 & 1.37 & -1.00 & 0.37 & 0.37 & -1.00 & 1.37 & -1.37 & 1.00 & -0.37 & -0.37 \\
1.00 & -1.37 & 0.37 & 1.00 & -1.37 & 0.37 & 1.00 & -1.37 & 0.37 & 1.00 & -1.37 & 0.37 \\
1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 \\
1.00 & -0.37 & -1.37 & -1.00 & 0.37 & 1.37 & 1.00 & -0.37 & -1.37 & -1.00 & 0.37 & 1.37 \\
1.00 & 0.37 & -0.37 & -1.00 & -1.37 & -1.37 & -1.00 & -0.37 & 0.37 & 1.00 & 1.37 & 1.37
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix}
$$

Fig 2.3. Kernel matrix of the DHT of order 12.

If appropriate grouping and shuffling is applied to the DHT of Fig. 2.3, the convolution structure in the matrix becomes clear. Fig 2.4 shows the structure which helps one evaluate the y vector through convolutions. Note that the computation consists of a single convolution for each possible (d, c) pair. The convolution is over the group A(N/2cd) or over A(N/cd) depending upon whether N/cd is even or odd, as discussed in Sec 2.2.

$$
\begin{bmatrix} y_0 \\ y_6 \\ y_1 \\ y_5 \\ y_7 \\ y_{11} \\ y_2 \\ y_{10} \\ y_3 \\ y_9 \\ y_4 \\ y_8 \end{bmatrix}
=
\begin{bmatrix}
1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
1.00 & 1.00 & -1.00 & -1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 & 1.00 \\
1.00 & -1.00 & 1.37 & -0.37 & -1.37 & 0.37 & 1.37 & -0.37 & 1.00 & -1.00 & 0.37 & -1.37 \\
1.00 & -1.00 & -0.37 & 1.37 & 0.37 & -1.37 & -0.37 & 1.37 & 1.00 & -1.00 & -1.37 & 0.37 \\
1.00 & -1.00 & -1.37 & 0.37 & 1.37 & -0.37 & 1.37 & -0.37 & -1.00 & 1.00 & 0.37 & -1.37 \\
1.00 & -1.00 & 0.37 & -1.37 & -0.37 & 1.37 & -0.37 & 1.37 & -1.00 & 1.00 & -1.37 & 0.37 \\
1.00 & 1.00 & 1.37 & -0.37 & 1.37 & -0.37 & 0.37 & -1.37 & -1.00 & -1.00 & -1.37 & 0.37 \\
1.00 & 1.00 & -0.37 & 1.37 & -0.37 & 1.37 & -1.37 & 0.37 & -1.00 & -1.00 & 0.37 & -1.37 \\
1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & -1.00 & -1.00 & -1.00 & 1.00 & 1.00 & 1.00 \\
1.00 & -1.00 & -1.00 & -1.00 & 1.00 & 1.00 & -1.00 & -1.00 & 1.00 & -1.00 & 1.00 & 1.00 \\
1.00 & 1.00 & 0.37 & -1.37 & 0.37 & -1.37 & -1.37 & 0.37 & 1.00 & 1.00 & 0.37 & -1.37 \\
1.00 & 1.00 & -1.37 & 0.37 & -1.37 & 0.37 & 0.37 & -1.37 & 1.00 & 1.00 & -1.37 & 0.37
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_6 \\ x_1 \\ x_5 \\ x_7 \\ x_{11} \\ x_2 \\ x_{10} \\ x_3 \\ x_9 \\ x_4 \\ x_8 \end{bmatrix}
$$

Fig 2.4. Shuffled Kernel of the DHT of order 12

In order to find the total number of different convolutions involved in the computation, one can define a parameter q = c.d. Since c | N/d, q is always an integer, and q | N. In fact, foe every q, q | N, there are $\chi(q)$ ways of choosing distinct (d, c) pairs satisfying our requirments. Here $\chi(q)$ denotes the number of ways q can be factored in two numbers. If N/q is even, then all these $\chi(q)$ convolutions are over the group A(N/2q) and if N/q is odd, then the $\chi(q)$ convolutions are over the group A(N/q). (The dimensionality of the convolution depends upon the structure, and in particular the direct factors of the group A(N/2q) or A(N/q), as might be the case). Table 2.1 lists the convolutions as they occur in the calculation of the DHT of length 12. The convolutions specified in this table can be identified with the structure of the DHT matrix of Fig 2.4.

Table 2.1. Convolutions involved in the DHT of N=12.

q	d	c	N/d or N/2d	convolution group	convolution size
1	1	1	6	A(6)	2×2
2	1	2	6	A(3)	2×2
	2	1	3	A(3)	2×2
3	1	3	6	A(2)	1×1
	3	1	2	A(2)	1×1
4	4	1	3	A(3)	2×2
6	6	1	1	-	-

Chapter 3
PREPROCESSING OF INPUT DATA

3.1. Introduction.

It was established in Chapter 2 that the discrete Hartley transform matrix can be partitioned into submatrices by proper grouping and ordering the row and column indices. Each of these submatrices may be computed using cyclic convolution techniques. By examining (2.6), (2.10) and (2.11), one can see that variable sequences, y_d, taking part in in the convolution are obtained by systematic addition (or subtraction) of selected input data components. This addition stage is called the preprocessing of input data and is dealt with in this chapter.

The Hartley transform length N can be factored as $N = 2^n Q$, for $n \geq 0$ and Q is an odd integer. The *primary partition index*, d, which has to divide N is therefore either a power of 2 or a product of two integers one of which is an odd integer. These two cases cover all the values of d since if d is not a power of 2 then it has to have an odd divisor. Section 3.2 and 3.3 deal with these two cases respectively.

3.2. Primary partition index, a power of 2.

Equations (2.4) and (2.7) show the y_d computations that need to be computed before one can do any convolution processing. These two equations represent the preprocessing stage of the algorithm and need to be evaluated efficiently.

Let $N = 2^n Q$, where Q is an odd integer and $n \geq 0$, and d, the *primary partition index*, 2^s, $0 \leq s < n$. We calculate y_{2^s} for all $s = 0, 1, 2, \cdots, n$ as follows :

1. (Initialization). Set $N_0 = N$ and $t_0(i) = x(i), 0 \leq i < N_0$.

2. (Reccursion). For $s = 0, 1, \cdots, n\text{-}1$, set $N_{s+1} = N_s/2$ (3.1)

and perform step 3.

3. (Repeatition). For i = 0, 1, ..., $(N_s/2)$ evaluate

$$t_{s+1}(i) = t_s(i) + t_s(i + \frac{N_s}{2}), \text{ and} \tag{3.2}$$

$$y_{2^s}(i) = t_s(i) - t_s(i + \frac{N_s}{2}), \text{ if } N_s \text{ is even} \tag{3.3}$$

$$= t_s\,(i), \text{ if } N_s \text{ is odd.} \tag{3.4}$$

The above procedure may be proved as follows. $t_s(i)$ evaluated according to (3.1) has the expression

$$t_s(i) = t_{s-1}(i) + t_{s-1}(i + \frac{N_{s-1}}{2}), \qquad 0 \leq i < \frac{N_{s-1}}{2}. \tag{3.5}$$

Equation (3.1) can be solved recursively to give N_{s-1} as

$$N_{s-1} = \frac{N}{2^{s-1}}.$$

Using this (3.5) can be rewritten as

$$t_s(i) = \sum_{l=0}^{1} t_{s-1}(i + l.\frac{N}{2^s}). \tag{3.6}$$

Use of (3.6) repeatedly gives

$$t_s(i) = \sum_{l=0}^{1} \sum_{m=0}^{1} \sum_{n=0}^{1} \cdots \sum_{w=0}^{1} t_0(\, i +(\, l + 2m + 4n + \ldots + 2^s w).\frac{N}{2^s}).$$

In the above summation one can replace t_0 with x (using eqn (3.2)). Now notice that as the summation variables go through all their values, the quantity $(l + 2m + 4n + \ldots + 2^s w)$ takes all possible values from 0 to $2^{s+1} - 1$.[1]

[1]In fact the expression $(l + 2m + 4n + \ldots + 2^s w)$ can be considered to be an expression of a number between 0 and $2^{s+1}-1$ in binary number system; the number being $(w, \ldots, n, m, l)_2$.

Thus $t_s(i) = \sum_{l=0}^{2^{s+1}-1} x(i + l.\frac{N}{2^s})$.

Finally, using the value in (3.3) gives

$$y_{2^s}(i) = \sum_{l=0}^{2^{s+1}-1} x(i + 2l.\frac{N}{2^{s+1}}) - \sum_{l=0}^{2^{s+1}-1} x(i + (2l+1).\frac{N}{2^{s+1}})$$
$$= \sum_{l=0}^{2^{s+1}-1} (-1)^l\, x(i + l.\frac{N}{2^{s+1}}),$$

which is identical to (2.4).

The structure of the preprocessing addition stage is very regular. It can be represented as a butterfly diagram. Following example illustrates the fact.

Example 3.1 : Let $N = 12 = 2^2.3$. Using (3.1)-(3.5) y_1, y_2 and y_4 may be written as

$$y_1(i) = x(i) - x(i + 6), \qquad 0 \le i < 6.$$

$$y_2(i) = x(i) - x(i + 3) + x(i + 6) - x(i + 9), \qquad 0 \le i < 3,$$

$$y_2(i) = t_1(i) - t_1(i + 3).$$

Also since N_2 is odd we get y_4 as

$$y_4(i) = x(i) + x(i + 3) + x(i + 6) + x(i + 9), \qquad 0 \le i < 3,$$

$$= t_0(i) + t_0(i + 3) + t_0(i + 6) + t_0(i + 9),$$

$$= t_1(i) + t_1(i + 3),$$

$$= t_2(i).$$

Note that (3.2) gives the following expression for t_0, t_1 and t_2 sequences for $N = 12$.

$t_0(i) = x(i), \ 0 \leq i < 12,$

$t_1(i) = t_0(i) + t_0(i + 6), \ 0 \leq i < 6,$

and $t_2(i) = t_1(i) + t_1(i + 3), \ 0 \leq i < 3.$

The algorithm for evaluating y_1, y_2 and y_4 for $N = 12$ is shown in Fig 3.1 and mimics a truncated butterfly diagram used in fast transforms. Note that this same computation can also be arranged as shown in Fig 3.2. This new arrangement is very suitable for a parallel archirecture implementation.

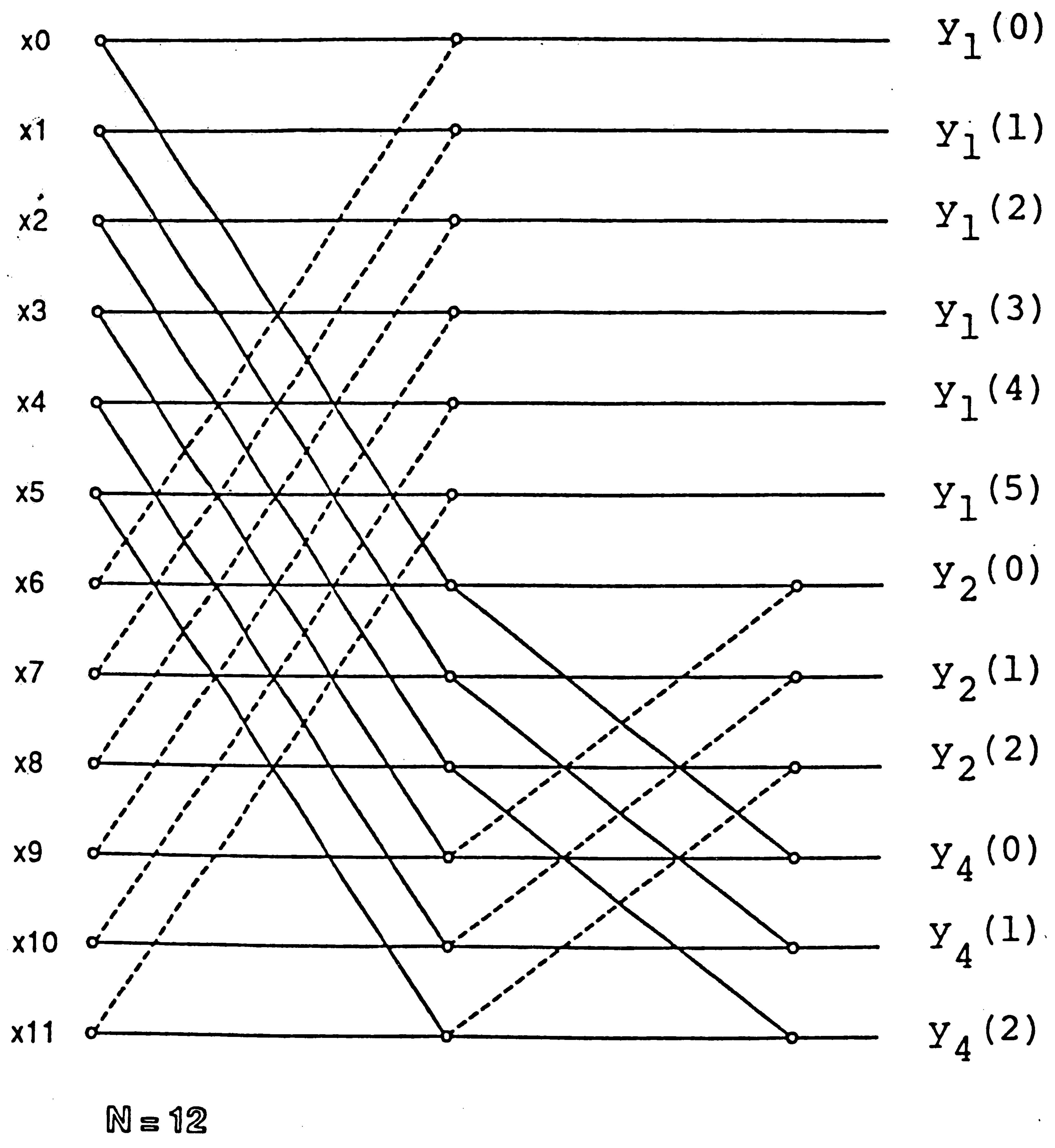


Fig. 3.1. Evaluation of y_1, y_2 and y_4 for $N = 12$.

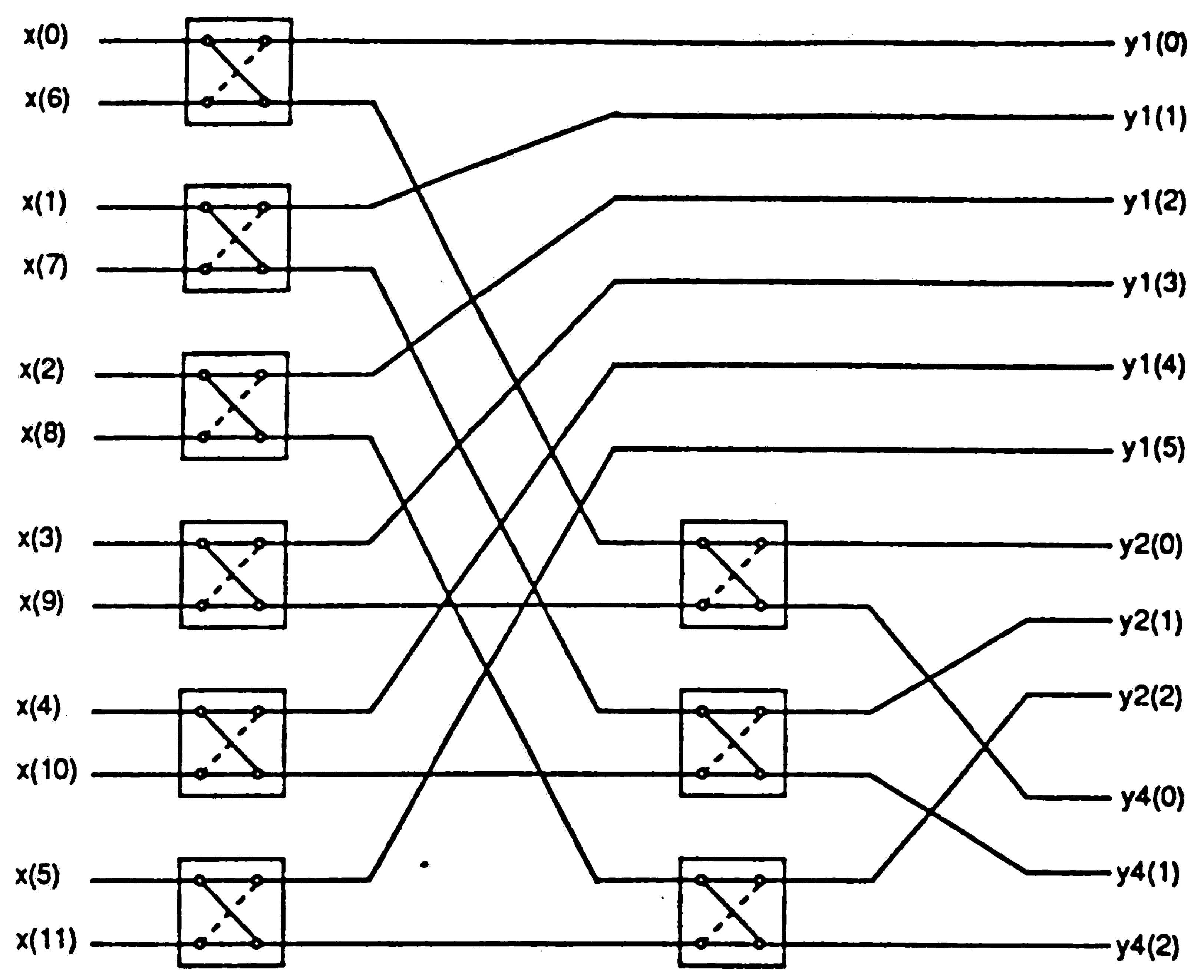


Fig. 3.2. Parallel architecture implementation of Fig. 3.1.

One can compute the additive complexity of this preprocessing stage from (3.2) and (3.3). We see that to calculate y_{2^s} and t_{s+1} from t_s we need $N_s (= N/2^s)$ additions. Therefore the total number of additions to calculate all y_{2^s}, $0 \leq s \leq n\text{-}1$ can be written as

$$\# \text{ of additions to calculate } y_{2^s} = \sum_{s=0}^{n-1} \frac{N}{2^s} = N \sum_{s=0}^{n-1} \frac{1}{2^s}.$$

$$= 2N(1 - \frac{1}{2^n}),$$

$$= 2(N - Q). \qquad (3.7)$$

3.3. Factorizable primary partition index.

Suppose now that the *primary partition index*, d, is not a power of 2. Then d can be represented as r.s, with some odd s. Evaluation of y_{rs} from y_r requires consideration of whether N/d is even or odd. Accordingly, we deal with it in the following two cases.

Case 1 : This case concerns expressing y_{rs} in terms of y_r, given the fact that $2 \mid (N/rs)$ and s is odd. Since r.s is a divisor of N and also 2 divides N/rs, one can use (2.4) and express y_{rs} as follows

$$y_{rs}(i) = \sum_{l=0}^{2rs-1} (-1)^l \, x(i + l.\tfrac{N}{2rs}),$$

Let $l = sl_1 + l_2, \quad 0 \le l_2 < s \ , \ 0 \le l_1 < 2r$

$$\text{Then}, y_{rs}(i) = \sum_{l_1=0}^{2r-1} \sum_{l_2=0}^{s-1} (-1)^{sl_1}(-1)^{l_2} \ x(i + l_1.\tfrac{N}{2r} + l_2.\tfrac{N}{2rs}).$$

We can express $(-1)^{sl_1}$ as $(-1)^{l_1}$ since s is odd. Thus

$$y_{rs}(i) = \sum_{l_1=0}^{2r-1} \sum_{l_2=0}^{s-1} (-1)^{l_1} (-1)^{l_2} x(i + l_1.\tfrac{N}{2r} + l_2.\tfrac{N}{2rs}), \qquad 0 \le i < \tfrac{N}{2rs}.$$

$$= \sum_{l_2=0}^{s-1} (-1)^{l_2} \sum_{l_1=0}^{2r-1} (-1)^{l_1} x(i + l_1.\tfrac{N}{2r} + l_2.\tfrac{N}{2rs}), \qquad 0 \le i < \tfrac{N}{2rs}.$$

Now $y_r(i)$ can be expressed from (2.7) since $2 \mid \frac{N}{r}$ as

$$y_r(i) = \sum_{l=0}^{2r-1} (-1)^l \, x(i + l.\tfrac{N}{2r}), \qquad 0 \le i < \tfrac{N}{2r}.$$

or $$y_r(i + l_2.\tfrac{N}{2rs}) = \sum_{l_1=0}^{2r-1} (-1)^{l_1} x(i + l_2.\tfrac{N}{2rs} + l_1.\tfrac{N}{2r}), \qquad 0 \le i < \tfrac{N}{2r}.$$

Substituting this value of y_r in the expression for y_{rs} one gets

$$y_{rs}(i) = \sum_{l_2=0}^{s-1} (-1)^{l_2} \ y_r(i + l_2.\tfrac{N}{2rs}), \qquad 0 \le i < \tfrac{N}{2rs}. \qquad (3.8)$$

Equation (3.8) expresses y_{rs} in terms of y_r when s is odd and $2 \mid (N/rs)$.

In order to express the computational complexity of evaluating y_{rs} from y_r, following notation may be used :

$A(d_1, d_2)$ = denotes the number of additions required to compute y_{d_2} from y_{d_1}.

$A(d_1, d_2, \cdots, d_t)$ = denotes the number of additions required to compute y_{d_t} begining from y_{d_1} and going through $y_{d_2}, \cdots, y_{d_{t-1}}$.

Computation (3.7) clearly yields following expression for A(r, rs) :

$$A(r, rs) = \frac{N}{2rs}(s-1), \quad \text{when } 2 \mid \frac{N}{rs} \text{ and s is odd.} \tag{3.9}$$

Fig. 3.3 illustrates equation (3.8) as y_3 is calculated from y_1 for N = 12.

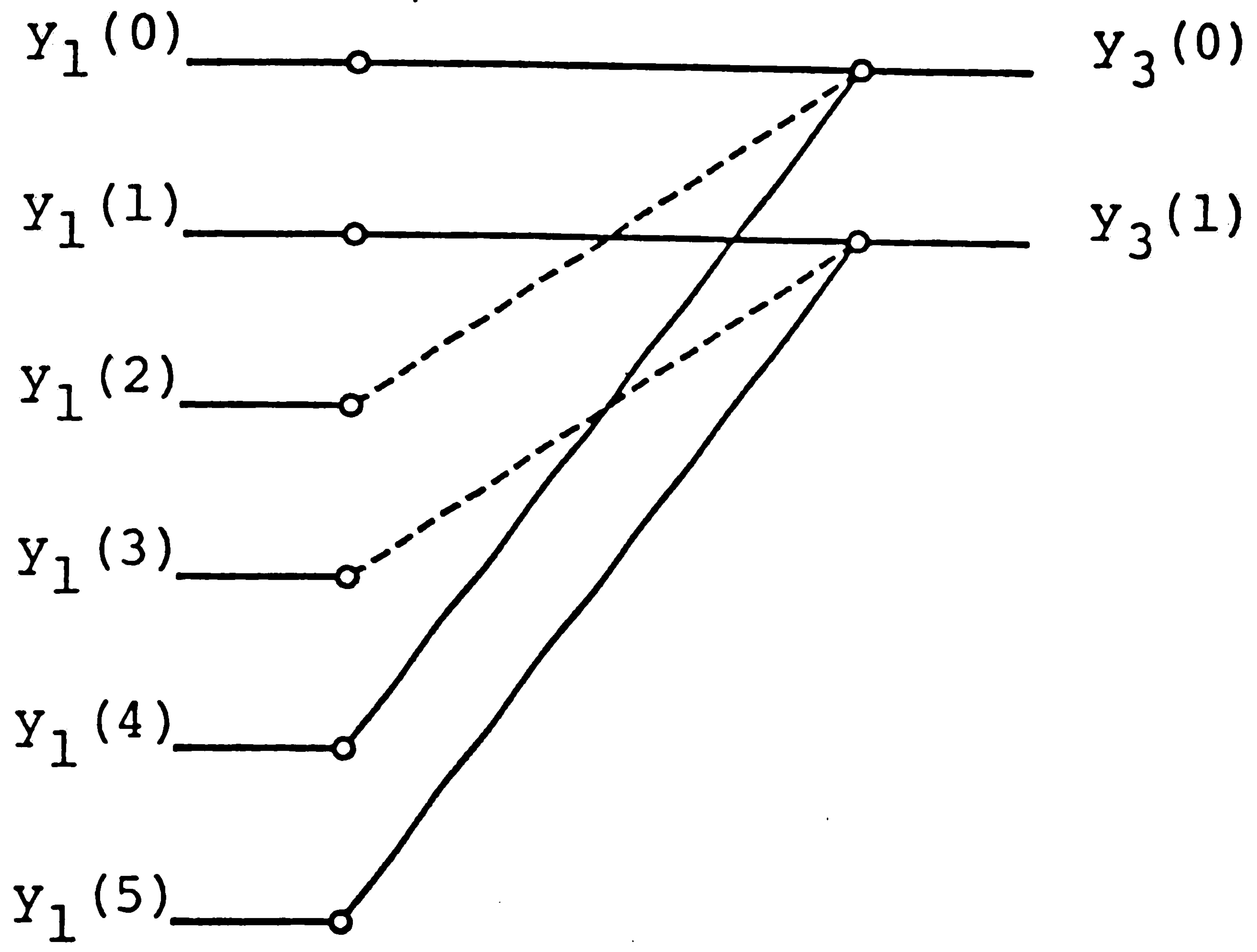


Fig 3.3. Calculation of y_3 from y_1 for N = 12.

Case 2 : When s is odd but 2 does not divide N/rs, one can y_{rs} in terms of y_r from (2.2) as follows :

$$y_{rs}(i) = \sum_{l=0}^{rs-1} x(i + l.\, \tfrac{N}{rs}),$$

As before, let $s = sl_1 + l_2, \quad 0 \le l_2 < s, \quad 0 \le l_1 < r.$

$$\text{Then, } y_{rs}(i) = \sum_{l_2=0}^{s-1} \sum_{l_1=0}^{r-1} x(i + l_1 . \tfrac{N}{r} + l_2 . \tfrac{N}{rs}), \qquad 0 \le i < \tfrac{N}{rs}.$$

Now, since 2 does not divide $\frac{N}{rs}$, and s is odd, 2 also does not divide $\frac{N}{r}$.

Using (2.2), $y_r(i)$ can be expressed as

$$y_r(i) = \sum_{l=0}^{r-1} x(i + l.\, \tfrac{N}{r}), \qquad 0 \le i < \tfrac{N}{r}.$$

Thus, $y_r(i + l_2.\frac{N}{rs}) = \sum_{l_1=0}^{r-1} x(i + l_2.\frac{N}{rs} + l_1.\frac{N}{r})$,

Finally using this value of y_r in the expression for y_{rs} one gets

$$y_{rs}(i) = \sum_{l_2=0}^{s-1} y_r(i + l_2.\frac{N}{rs}), \qquad 0 \le i < \frac{N}{rs}. \tag{3.10}$$

Relation (3.10) allows one to evaluate y_{rs} from y_r when s is odd and 2 does not divide N/rs. The complexity of this computation can be easily seen to be

$$A(r, rs) = \frac{N}{rs}(s-1) \quad \text{when s is odd and 2 does not divide } \frac{N}{rs}. \tag{3.11}$$

(3.10) is clearly illustrated in Fig, 3.4, in which y_6 is calculated from y_2 for $N = 30$. Both (3.9) and (3.11) suggest that one should choose the smallest possible value of s for a given rs for minimum complexity of the preprocessing stage. For example, given y_3 and y_5, the evaluation of y_{15} should be done from y_5 rather that y_3 because it would involve 2N/15 additions instead of 4N/15.

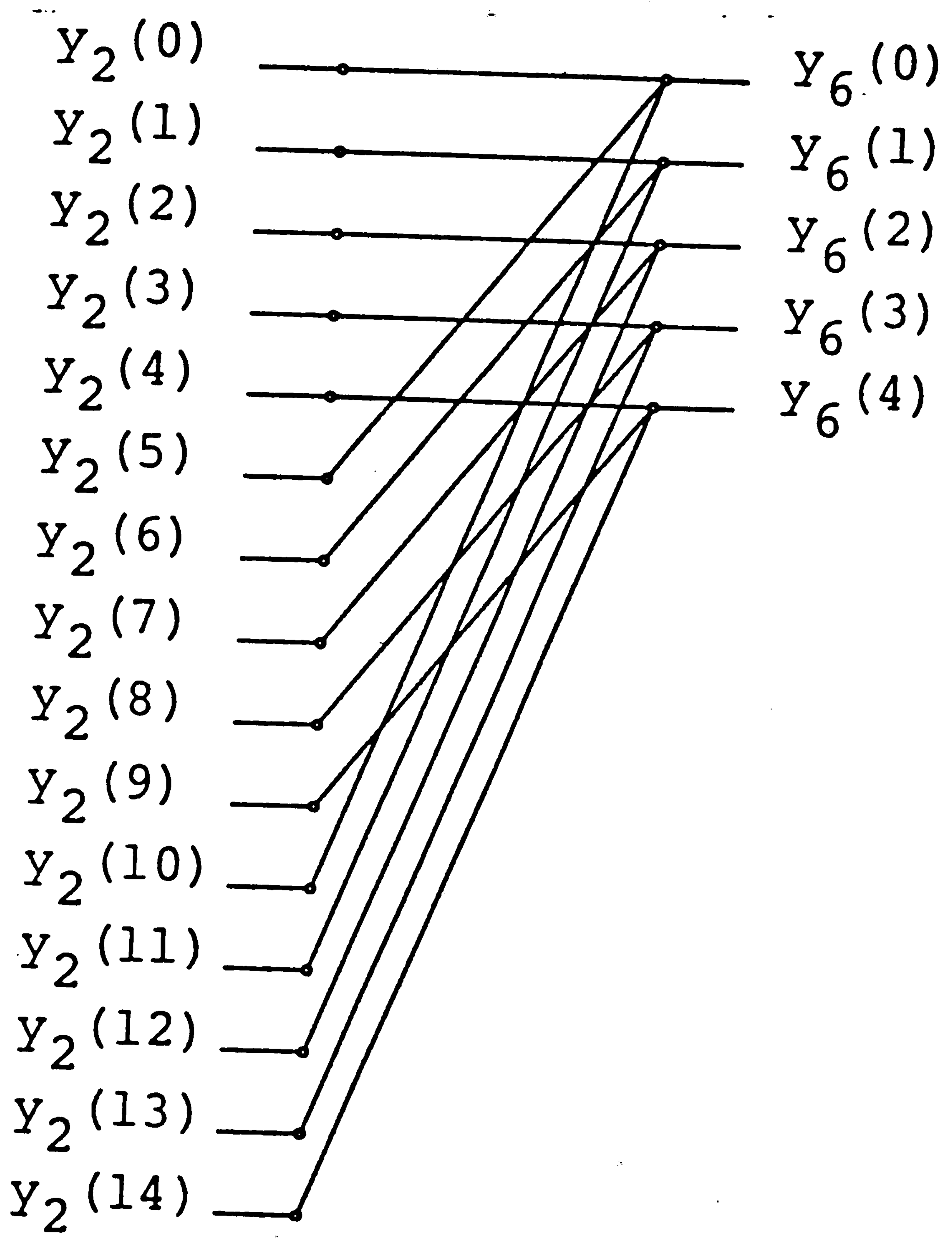


Fig 3.4. Calculation of y_6 from y_2 for $N = 30$.

3.4. Special Cases.

This section deals with the total complexity of the preprocessing stage of four commonly occuring cases of N which has only one or two prime devisors.

Case 1 : $N = 2^n$. In this case, the only possible d's are of type 2^s. From Sec. 3.1 it is obvious that $Q = 1$ and

$$A(2^n) = 2(N-1). \tag{3.12}$$

Case 2 : $N = p^n$, p prime.

Here d can have values p^s, where $0 \leq s \leq n$. It is clear that one must compute y_p from y_1, y_{p^2} from y_p, and so on till y_{p^n}. Therefore the total number of additions to complete the preprocessing is

$$= \sum_{i=1}^{n} A(p^{i-1}, p^i).$$

$$= \sum_{i=1}^{n} \frac{N(p-1)}{p^i}, \qquad \text{from (3.11).}$$

$$= \sum_{i=1}^{n} (p-1)\, p^{n-i}.$$

$$= p^n - 1.$$

$$= N - 1.$$

Fig 3.5 illustrates the computation when $N = 3^3$.

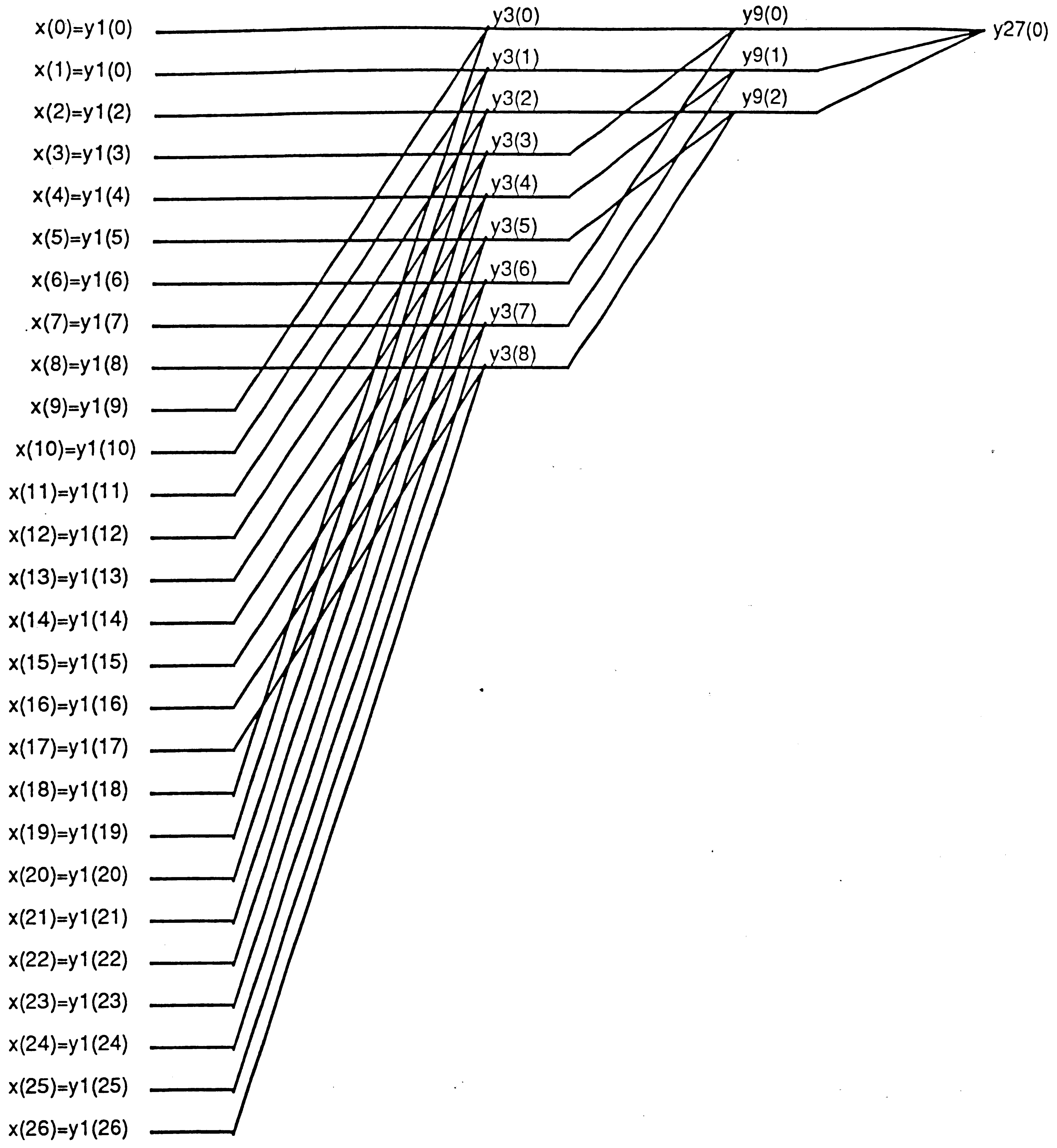


Fig 3.5. Computation of y_1, y_3, y_9 and y_{27} for $N = 3^3$.

Case 3 : $N = 2^{n_1} p^{n_2}$, p prime.

This complexity for this case is computed in three seperate parts. The firse part consists of the d's which are powers of 2. Since these are independent they have to be calculated specifically. Therefore the total number of additions to compute these d's is $= 2(N -$

p^{n_2})(from eqn. 3.7). The second part are the d's which can have the form $2^{l_1}p^{l_2}(l_1 \neq n_1)$.

If this y_d is calculated from y_r where $r = 2^{t_1}p^{t_2}$, then the complexity in evaluating y_d is

$$A(2^{t_1}p^{t_2}, 2^{l_1}p^{l_2}) = \frac{N(2^{l_1-t_1}p^{l_1-t_1} - 1)}{2^{l_1+1}p^{l_2}}.$$

To minimize the expression above one must chose values of $t_1 = l_1$ and $t_2 = l_2-1$. If this is done, one gets

$$A(2^{t_1}p^{t_2}, 2^{l_1}p^{l_2}) = \frac{N(p-1)}{2^{l_1+1}p^{l_2}}.$$

The third part is where d's have the form $2^{n_1}p^{l}$. The complexity for this case is minimized if it is calculated from the y_r where r has the form $2^{n_1}p^{l-1}$. Now one gets

$$A(2^{n_1}p^{l-1}, 2^{n_1}p^{l}) = \frac{N(p-1)}{2^{n_1}p^{l}}.$$

Thus complexity of the preprocessing stage, $A(2^{n_1}p^{n_2})$, is

$$A(2^{n_1}p^{n_2}) = 2(N - p^{n_2}) + \sum_{j=0}^{n_1-1}\sum_{i=1}^{n_2} \frac{N(p-1)}{2^{j+1}p^{i}} + \sum_{l=1}^{n_2} \frac{N(p-1)}{2^{n_1}p^{l}}.$$

$$= 2(N - p^{n_2}) + (N - p^{n_2} - 2^{n_1} + 1) + (p^{n_2+1} - 1)$$

$$= 3N - (p-3)p^{n_2} - 2^{n_1}.$$

<u>Case 4</u> : $N = p_1^{n_1}p_2^{n_2}$, p_1 and p_2 prime, $p_1 > p_2$.

In this case, d would have the form $p_1^{l_1}p_2^{l_2}$. If this y_d is evaluated from y_r where $r = p_1^{t_1}p_2^{t_2}$, then the complexity involved in the evaluation of this y_d is

$$A(p_1^{t_1}p_2^{t_2}, p_1^{l_1}p_2^{l_2}) = \frac{N(p_1^{l_1-t_1}p_2^{l_1-t_1} - 1)}{p_1^{l_1}p_2^{l_2}}$$

To minimize the expression above when $p_1 > p_2$, one must chose values of $t_1 = l_1$ and

$t_2 = l_2 - 1$. If this is done, one gets

$$A(p_1^{t_1} p_2^{t_2}, p_1^{l_1} p_2^{l_2}) = \frac{N(p_2 - 1)}{p_1^{l_1} p_2^{l_2}}$$

Clearly, one will also have to compute $y_{p^{t_1}}$, $0 \le t_1 \le n_1$. Thus complexity of the preprocessing stage, $A(p_1^{n_1} p_2^{n_2})$, is

$$\begin{aligned} A(p_1^{n_1} p_2^{n_2}) &= \sum_{j=1}^{n_1} A(p_1^{j-1}, p_1^{j}) + \sum_{j=0}^{n_1} \sum_{i=1}^{n_2} A(p_1^{j} p_2^{i-1}, p_1^{j} p_2^{i}), \\ &= \sum_{j=1}^{n_1} \frac{N(p_1 - 1)}{p_1^{j}} + \sum_{j=0}^{n_1} \sum_{i=1}^{n_2} \frac{N(p_2 - 1)}{p_1^{j} p_2^{i}}, \\ &= \frac{N(p_1 - 1)}{p_1^{n_1}} \frac{p_1^{n_1} - 1}{(p_1 - 1)} + \frac{N(p_2 - 1)}{p_1^{n_1} p_2^{n_2}} \frac{p_1^{n_1+1} - 1}{(p_1 - 1)} \frac{p_2^{n_2} - 1}{(p_2 - 1)}, \\ &= (N - 1) + \frac{(Np_1 - p_1^{n_1+1} - p_2^{n_2} + 1)}{(p_1 - 1)}. \end{aligned}$$

Chapter 4
CONVOLUTION PROCESSING

4.1. Introduction.

It was established in Chapter 2 that the discrete Hartley transform matrix can be partitioned into smaller cyclic submatrices which may be computed using multidimentional cyclic convolution techniques. In particular as (2.6) shows, when N/d is odd, the submatrix corresponds to a group table of $A(N/cd)$. However, when N/d is even, (2.10) and (2.11) show that while h_i and g_j belong to the group $A(N/2cd)$, the operation $\oplus$ is multiplication modulo N/cd. This implies that the corresponding submatrices do not form group tables. This chapter will establis the structure of the matrices as Toeplitz. In order to simplify the notation let the effective convolution length, N' denote N/cd.

The case of $N' = 2^n$, $n > 2$ is dealt with in Section 4.2, $N' = 2^nQ$, $n = 2$ or 3, odd Q in Section 4.3 and $N' = 2^nQ$, $n > 3$, odd Q in Section 4.4.

4.2. Effective convolution length, a power of 2.

If $N' = 2^n$ and $n = 1$ or 2, then the group to which h_i and g_j belong, $A(N'/2)$, has only one element. Thus the submatrices under consideration is of size 1×1 and has no structure. Thus the only case of interest is $n > 2$. In this case consider the group

$$G = A(2^n) = C_2 \times C_{2^{n-2}}.$$

The group operation is still multiplication modulo N'. Form C_2 by using the order two element $(2^{n-1}-1)$ of G, i.e.,

$C_2 = \{ 1 , 2^{n-1}-1 \}$.

Let α be any element of G of order 2^{n-2} (when n=3, then α is an elemnt of order 2 not equal to $(2^{n-1}-1)$) and organize the elements of $A(2^{n-1})$ as

$$\alpha^i C_2 \bmod 2^{n-1},\ 0 \le i \le 2^{n-3}-1.$$

If the row index h_i and the column index g_j are arranged according to this order, one can see that the matrix transforms into a Kronecker product of an order 2 cyclic matrix and a Toeplitz matrix. The statement is fairly easy to prove. The order 2 cyclic submatrices are aresult of the subgroup C_2. Further because of the chosen order, the (i, j)th submatrix corresponds to $\alpha^i C_2 . \alpha^j C_2 = \alpha^{i+j} C_2$ elements of the group. Since this entry depends upon i+j rather than individual i or j values, the resultant structure is Toeplitz. Example 4.1 illustrates this restructuring of the matrix.

<u>Example 4.1</u> : Let N be equal to 16 and both c and d have the value 1. Thus $N' = 16 = 2^4$. Also, h_i and $g_j \in A(8) = \{ 1, 3, 5, 7 \}$ and $A(16) = \{ 1, 3, 5, 7, 9, 11, 13, 15 \}$. The matrix M looks like

$$\begin{array}{c|cccc|}
 & 1 & 3 & 5 & 7 \\
\hline
1 & 1.31 & 1.31 & 0.54 & -0.54 \\
3 & 1.31 & -1.31 & 0.54 & 0.54 \\
5 & 0.54 & 0.54 & -1.31 & 1.31 \\
7 & -0.54 & 0.54 & 1.31 & 1.31 \\
\hline
\end{array}$$

Let C_2 be $\{ 1, 7 \}$. Choose α as 5 (order of 5 is $2^2 = 4$).

Using this, if one orders the rows and columns as 1, 7, 5, 3 one gets M transformed to

$$\begin{array}{c|cccc|} & 1 & 7 & 5 & 3 \\ 1 & 1.31 & -0.54 & 0.54 & 1.31 \\ 7 & -0.54 & 1.31 & 1.31 & 0.54 \\ 5 & 0.54 & 1.31 & -1.31 & 0.54 \\ 3 & 1.31 & 0.54 & 0.54 & -1.31 \end{array}$$

If the cyclic 2x2 matrices are represented by

$$C_1 = \begin{bmatrix} 1.31 & -0.54 \\ -0.54 & 1.31 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0.54 & 1.31 \\ 1.31 & 0.54 \end{bmatrix}, \quad C_3 = \begin{bmatrix} -1.31 & 0.54 \\ 0.54 & -1.31 \end{bmatrix},$$

M can be represented as

$\begin{bmatrix} C_1 & C_2 \\ C_2 & -C_1 \end{bmatrix}$, which has a Toeplitz matrix structure.

It should be mentioned here that the shuffling of rows and columns of M does not contribute any additional computational complexity as demonstrated in Chapter 2.

When $N' = 2^n$, above structure enables one to compute the matrix for the specific values of c and d. Theorem 4.1 ensures that all the $\alpha^i C_2 \bmod 2^{n-1}$ values generated are distinct.

Theorem 4.1 : $\alpha^i C_2 \bmod 2^{n-1}$, $0 \leq i \leq 2^{n-3} - 1$ are all distinct.

Proof : Denote by H_i, set $\{ \alpha^i \bmod 2^{n-1}, \alpha^i(2^{n-1}-1) \bmod 2^{n-1} \}$. Let $H_i = (a, b)$ and $H_j = (c, d)$. If we can prove that a, b, c, d are all distint for $0 \leq i, j \leq 2^{n-3}-1$ and i

$\neq$ j, then we have essentially proved the stipulation. We consider the following four cases and use the method of contradiction to disprove each.

Case I : Suppose $a \equiv b \bmod 2^{n-1}$. Here $a = \alpha^i$ and $b = \alpha^i(2^{n-1}-1) = -\alpha^i \bmod 2^{n-1}$. The assumption $a \equiv b \bmod 2^{n-1}$ implies that

$$\alpha^i = -\alpha^i \bmod 2^{n-1}.$$

Thus $2^{n-1} \mid 2\alpha^i$ or $\alpha^i = 0 \bmod 2^{n-2}$.

This is a contradiction since $\alpha \in A(2^n)$ and therefore odd, which implies that no power of α can be divisible by an even integer(namely 2^{n-2}) for $n > 2$.

Case II : Suppose $a \equiv c \bmod 2^{n-1}$. $a = \alpha^i$ and $c \doteq \alpha^j$. The assumption $a \equiv c \bmod 2^{n-1}$ implies that

$\alpha^i = \alpha^j \bmod 2^{n-1}$. SInce $\alpha \in A(2^n)$, α has an inverse modulo 2^{n-1}.

Thus, $\alpha^{i-j} = 1 \bmod 2^{n-1}$ which gives $\alpha^{i-j} \bmod 2^n = 1$ or $2^{n-1}+1$.

In either case a contradiction arises since α is of order 2^{n-2}

i.e, $\alpha^{i-j} \bmod 2^n \neq 1$ since $i-j < 2^{n-2}$(since $i,j < 2^{n-3}$)

and if $\alpha^{i-j} \bmod 2^n = (1 + 2^{n-1})$

then squaring both sides of the equation and taking mod 2^n, one gets

$\alpha^{2(i-j)} \bmod 2^n = (1 + 2^{n+1}) \bmod 2^n = 1 \bmod 2^n$ which implies that $(i-j) \geq 2^{n-3}$

which is again impossible.

Case III : Suppose $a \equiv d \bmod 2^{n-1}$.

$a = \alpha^i$ and $d = \alpha^j(2^{n-1} - 1) = -\alpha^j \bmod 2^{n-1}$.

The assumption $a \equiv d \bmod 2^{n-1}$ implies that

$\alpha^i = -\alpha^j \bmod 2^{n-1}$,

i.e, $\alpha^{i-j} = -1 \bmod 2^{n-1}$.

Thus $\alpha^{i-j} \bmod 2^n = (2^n-1)$ or $(2^{n-1}-1)$. However (2^n-1) and $(2^{n-1}-1)$ are order two elements in group $A(2^n)$ and since α is not an order two element we again find a contradiction.

Case IV : Finally, Suppose $b \equiv d \bmod 2^{n-1}$. Since $b = \alpha^i(2^{n-1}-1)$ and $d = \alpha^j(2^{n-1}-1)$ $= -\alpha^j \bmod 2^{n-1}$ the assumption $b \equiv d \bmod 2^{n-1}$ implies that $\alpha^i(2^{n-1}-1) = \alpha^j(2^{n-1}-1)$ mod 2^{n-1}, or $\alpha^i = \alpha^j \bmod 2^{n-1}$. This has been proved impossible in case II earlier. Thus in all four cases a,b,c,d are all distint for $0 \leq i,j \leq 2^{n-3}-1, \neq j$. □

Theorem 4.2 proves that 3 has order 2^{n-2} in the group $A(2^n)$ and can therefore be the element α.

Theorem 4.2 : Order of 3 in Group $A(2^n)$ is 2^{n-2} $(n > 2)$.

Proof : One can use the method of mathematical Induction to prove this assertion. When n=3, one can verify by direct calculation that the order of 3 in $A(2^3)$ is 2^1. Now assume that order of 3 in $A(2^n)$ is 2^{n-2}. To find order of 3 in $A(2^{n+1})$, note that

$$3^{2^{n-2}} = 1 \bmod 2^n \text{ implies } 3^{2^{n-2}} = k.2^n + 1.$$

By squaring both sides of this equation and reducing by mod 2^{n+1}, one gets $3^{2^{n-1}} \bmod 2^{n+1} = 1$. Thus if L is the order of 3 in $A(2^{n+1})$, then $L \mid 2^{n-1}$. (4.1)

Now if $L \neq 2^{n-1}$, then $L \mid 2^{n-2}$ and consequently,

$$3^{2^{n-2}} \bmod 2^{n+1} = 1.$$

thus $2^{n+1} \,|(3^{2^{n-2}} - 1) = (3^{2^{n-3}} - 1)(3^{2^{n-3}} + 1)$.

Note that even though $3^{2^{n-3}}+1$ is even, 2^2 cannot divide it as $(3^{2^{n-3}} + 1) \bmod 4 = ((-1)^{2^{n-3}}+1) = 2$. Hence $2^n|(3^{2^{n-3}}-1)$ implying that the order of 3 in $A(2^n)$ divides 2^{n-3}, which is contrary to the assumption. Therefore $L = 2^{n-1}$. The proof is thus complete by Mathematical Induction. $\square$

4.3. Effective convolution length, 4Q or 8Q (for odd Q).

Let $N' = 2^nQ$, $n = 2, 3$ and Q odd. One can write the following isomorphic relationships for A(4Q) and A(8Q).[1]

$$A(4Q) \equiv C_2 \times A(2Q).$$

$$A(8Q) \equiv C_2 \times A(4Q).$$

The subgroup C_2 in the two cases can be generated by the elements $2Q + 1$ and $4Q + 1$ respectively. Let α represent the generator of C_2.

As in the previous case, here again $h_i, g_j \in A(N'/2)$ while the matrix elements are described by the group operation of the group $A(N')$. Example 4.2 illustrates this case.

Example 4.2 : Let $N = 36$, $c = d = 1$. Therefore $N' = 36(= 2^2.9)$.
Also $A(36) = \{ 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35 \}$ and $A(18) = \{ 1, 5, 7, 11, 13, 17 \}$. Let $\alpha(= 2\times 9 + 1 = 19)$ the generator of the group C_2. Let G_1 be the group generated by the element 5 with the operation multiplication mod 36.

[1]Note that A(16Q) is not isomorphic to $C_2 \times A(8Q)$. In general $A(2^nQ) \equiv C_2 \times A(2^{n-1}Q)$ iff n is 2 or 3.

The groups $C_2 = \{ 1, 19\}$ and $G_1 = \{ 1, 5, 25, 17, 13, 29 \}$ give

$$A(36) = \{ 1, 19 \} \times \{ 1, 5, 25, 17, 13, 29 \}$$

One can also see that G_1 is isomorphic to A(18). The submatrix formed by rows and columns from A(18) looks like :

	1	5	7	17	13	11
1	1.158	1.409	1.282	-0.811	0.123	0.598
5	1.409	-1.282	0.811	0.123	-0.598	-1.158
7	1.282	0.811	0.123	0.599	-1.158	1.409
17	-0.811	0.123	0.599	1.158	1.409	1.282
13	0.123	-0.599	-1.158	1.409	-1.282	0.811
11	0.599	-1.158	1.409	1.282	0.811	0.123

One can see that this matrix does not conform to a cyclic structure. If, on the other hand, one arranges the rows and columns according to the isomorphic group G_1, one gets the following cyclic matrix.

	1	5	25	17	13	29
1	1.158	1.409	-1.282	0.811	0.123	-0.598
5	1.409	-1.282	0.811	0.123	-0.598	1.158
25	-1.282	0.811	0.123	-0.599	1.158	1.409
17	0.811	0.123	-0.599	1.158	1.409	-1.282
13	0.123	-0.599	1.158	1.409	-1.282	0.811
29	-0.599	1.158	1.409	-1.282	0.811	0.123

As indicated in example 4.2, use of group G_1 instead of $A(N'/2)$ gives a cyclic convolution structure. One must now prove that G_1 is indeed isomorphic to $A(N'/2)$.

Lemma 4.1 : Let $A(4Q) \equiv \{1, \alpha\} \times G_1 \equiv C_2 \times A(2Q)$ where $\alpha = 2Q + 1$. If $x \in A(4Q)$ then exactly one of x or $(x + 2Q) \bmod 4Q$ is in G_1.

Proof : Suppose both x and $(x + 2Q) \bmod 4Q$ are in G_1. Let $y \in G_1$ denote the inverse of x in G_1, i.e, $xy \bmod 4Q = 1$. Since $y \in A(4Q)$, y is odd. Therefore,

$$(x + 2Q)\cdot y \bmod 4Q = 1 + 2Q.$$

However, $x + 2Q, y \in G_1$, hence their product $(1 + 2Q) \in G_1$. However, $1 + 2Q = \alpha \notin G_1$. This contradiction leads us to the result that both x and $(x + 2Q) \bmod 4Q$ cannot simultaneously belong to G_1. □

Lemma 4.2 : If $x \in A(4Q)$ then atleast one of x or $(x + 2Q) \bmod 4Q \in G_1$.

Proof : Suppose $x \notin G_1$. However since $x \in G_1 = \{ 1, 1+2Q \} \times G_1$, there must exist some $y \in G_1$ such that $x = (1 + 2Q)y \bmod 4Q = (y + 2Q) \bmod 4Q$. This last step uses the fact that y is odd. Thus $(x + 2Q) \bmod 4Q = y$ and therefore $\in G_1$. □

Following two lemmas are equivalent to Lemma 4.1 and 4.2 when $N' = 8Q$.

Lemma 4.3 : Let $A(8Q) \equiv \{1, \alpha\} \times G_1 \equiv C_2 \times A(4Q)$ where $\alpha = 4Q + 1$. If $x \in A(8Q)$ then both x and $(x + 4Q) \bmod 8Q$ is not in G_1.

Proof : Suppose both x, $(x + 4Q) \bmod 8Q \in G_1$. Let y denote the inverse of x in G_1. Clearly, y is odd.

$$(x + 4Q)y \bmod 8Q = 1 + 4Q.$$

Since both $x + 4Q$ and y are elements of G_1, their product $1 + 4Q$ is also in G_1. However, $1 + 4Q = \alpha \notin G_1$. This contradiction leads us to the result that both x and (x

$+ 4Q)$ mod $8Q$ cannot simultaneously belong to $\hat{G}_1$. □

Lemma 4.4 : If $x \in A(8Q)$ then atleast one of x or $(x + 4Q)$ mod $8Q \in G_1$.

Proof : Suppose $x \notin G_1$. Since $x \in G = \{ 1, 1+4Q \} \times G_1$, there must exist a y $\in G_1$(y is odd) such that $x = (1+4Q)y$ mod $8Q = (y + 4Q)$ mod $8Q$. Thus $(x + 4Q)$ mod $8Q = y \in G_1$. □

It is now shown that when $N' = 4Q$ or $8Q$, (Q odd), G_1 and $A(N'/2)$ are isomorphic and this isomorphism brings out the cyclic structure of the transform kernel after changing signs of certain row and column elements.

Theorem 4.3. The matrix $M(g_j, h_i)$ can be expressed as

$$M(\phi(h_i), \phi(g_j)) = \delta(h_i)\, \delta(g_j)\, f(h_i \odot g_j), \qquad h_i, g_j \in G_1,$$

where $A(N') = C_2 \times G_1$, $\odot$ denotes the group operation of the group G_1, $\delta : G_1 \rightarrow \{ 1 , -1 \}$ an d $\phi : G_1 \rightarrow G_2$ ϕ is a one-one onto mapping.

Proof : Since n=2 or 3, $A(N')$ has the desired structure. $A(N') = A(2^nQ) \equiv C_2 \times G_1$ and $A(N'/2) = A(2^{n-1}Q) = G_2$. An isomorphism between G_1 and G_2 can be established using the function $\phi : G_1 \rightarrow G_2$ defined as

$$\phi(x) = x \bmod 2^{n-1}Q, \; x \in G_1.$$

To see that $\phi(x)$ is an isomorphism note that $\phi(x) \in G_2$ because $\gcd(x, 2^nQ) = 1$, implying $\gcd(x \bmod 2^{n-1}Q, 2^nQ) = 1$. Also $(x \bmod 2^{n-1}Q) < 2^{n-1}Q$. Thus $\phi(x) \in A(2^{n-1}Q) = G_2$. For any $y \in G_2$, both y and $(y + 2^{n-1}Q)$ belong to $A(2^nQ)$ and as per Lemma 4.1 and 4.2 (4.3 and 4.4 for n=3) exactly one of them is in G_2. Thus any given y $\in G_2$ is image of some element of G_1. Finally, to complete the proof for isomorphism, one should show that $\phi(x)$ is one-one. Assume $\phi(x_1) = \phi(x_2)$ for $x_1, x_2 \in G_1$ and $x_1 > x_2$.

Then

$$x_1 \bmod 2^{n-1}Q = x_2 \bmod 2^{n-1}Q$$

or $\quad x_1 = x_2 + 2^{n-1}Q.$

This is in contradiction to the Lemmas 4.1 & 4.2 (4.3 and 4.4 for n=3) which showed that both x_2 and $x_2 + 2^{n-1}Q$ cannot belong to G_1. Hence $\phi(x) = x \bmod 2^{n-1}Q$ defines an isomorphism $\phi : G_1 \rightarrow G_2$.

Now define function $\delta : G_1 \rightarrow \{ 1 , -1 \}$ as

$$\delta(x) = 1 \text{ if } x < N'/2,$$

$$= -1 \text{ if } x > N'/2.$$

Let $h_i, g_j \in G_1$. Clearly both of these are odd since $G_1 \subset A(2^nQ)$. Consider the case of $h_i < N'/2$. One has in this case, $\phi(h_i) = h_i$, $\delta(h_i) = 1$. Therefore

$$M(\phi(h_i), g_j) = \text{cas}(\frac{h_i g_j 2\pi}{N'}) = \delta(h_i).\ M(h_i, g_j).$$

On the other hand, if $h_i > N'/2$, then $\phi(h_i) = (h_i - N'/2)$, $\delta(h_i) = -1$. Thus again,

$$M(\phi(h_i), g_j) = \text{cas}(\pi g_j + \frac{h_i g_j\ 2\pi}{N'}) = \delta(h_i).\ M(h_i, g_j).$$

Therefore for all values of $h_i \in G_1$ we have

$$M(\phi(h_i), g_j) = \delta(h_i).\ M(h_i, g_j), \qquad h_i, g_j \in G_1.$$

Similarly for index g_j we get the relationship (using the fact that $\phi(h_i)$ is odd)

$$M(\phi(h_i), \phi((g_j)) = \delta(h_i)\ \delta(g_j)\ M(h_i, g_j), \qquad h_i, g_j \in G_1. \qquad (4.1)$$

However when $h_i, g_j \in G_1$,

$$M(h_i, g_j) = cas(\frac{h_i g_j 2\pi}{N'}) = cas(\frac{(h_i g_j \bmod N)\ 2\pi}{N'}).$$

$$= cas(\frac{(h_i \odot g_j)\ 2\pi}{N'}). \qquad (4.2)$$

$$= f(h_i \odot g_j).$$

where $\odot$ denotes the operation of group G_1 and $f : G_1 \rightarrow \Re$ is defined as

$$f(x) = cas\ (\frac{x\ 2\pi}{N'}),\ x \in G_1$$

Combining (4.1) and (4.2) gives the desired result

$$M(\phi(h_i), \phi(g_j)) = \delta(h_i)\ \delta(g_j)\ f(h_i \odot g_j) \qquad h_i, g_j \in G_1\ \phi(h_i), \phi(g_j) \in G_2\ \Box$$

Theorem 4.3 shows that in the case under consideration the kernel matrix is indeed similar to a group table of group G_1. This has the following computational implications. When $N' = 4Q$ or $8Q$ for odd Q the matrix $M(ch_i, dg_j)$, $h_i, g_j \in A(N'/2)$ can be evaluated through a multidimentional convolution. The structure of convolution is given by the structure of group $G_1 \equiv A(N'/2)$. Since ϕ is a one-one mapping from $G_1 \rightarrow G_2$, $M(\phi(h_i), \phi(g_j))$ is only reordering of the rows and columns of M. Similarly, $\delta(h_i)$ and $\delta(g_j)$ functions mearly correspond to changing signs of specific rows or columns and colud be implemented by negating corresponding signal and transform components.

4.4. Effective convolution length, $2^n Q$, $n > 3$, Q odd.

This section considers those N' that have an odd factor and are divisible by 16. It will be shown that the kernal matrix for $N' = 2^n Q$ (n>3) has a $2^{n-3} \times 2^{n-3}$ Toeplitz matrix structure and each elememt in this structure is a multidimentional cyclic matrix with structure dictated by group A(4Q). Following example illustrates this case

Example 4.2 : Let $N = 48$ ($2^4.3$) and $c = d = 1$ giving $N' = 48$.

Clearly, $A(48) = \{ 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47 \}$ and

$A(24) = \{ 1, 5, 7, 11, 13, 17, 19, 23 \}$.

The Matrix M, one is interested in this case, is defined as $M(i, j) = \text{cas}(ij2\pi/48)$; $i,j \in A(24)$. One can write $A(48)$ as $A(48) \equiv G_1 \times C_4 = \{ 1, 7, 41, 47 \} \times \{ 1, 11, 25, 35 \}$. Note that G_1 is isomorphic to $C_2 \times C_2$ and 11 is the generator for C_4. If one arranges the convolution using the following row and column order :

$$S = \{ \alpha^i.j \mid 0 \leq i \leq 1 , j \in G_1 \}.$$

It can be shown that there is a one-to-one correspondence between the elements of S and the elements of A(24) (In fact for every $i \in S$, $i \bmod 24 \in A(24)$ and is distict). The matrix with the stated order becomes :

$$\begin{array}{r|rrrrrrrr}
 & 1 & 7 & 41 & 47 & 11 & 29 & 19 & 37 \\
\hline
1 & 1.122 & 1.402 & -0.185 & 0.861 & 1.122 & -1.402 & -0.185 & -0.861 \\
7 & 1.402 & 1.122 & 0.861 & -0.185 & -1.402 & 1.122 & -0.861 & -0.185 \\
41 & -0.185 & 0.861 & 1.122 & 1.402 & -0.185 & -0.861 & 1.122 & -1.402 \\
47 & 0.861 & -0.185 & 1.402 & 1.122 & -0.861 & -0.185 & -1.402 & 1.122 \\
11 & 1.122 & -1.402 & -0.185 & -0.861 & -1.122 & -1.402 & 0.185 & -0.861 \\
29 & -1.402 & 1.122 & -0.861 & -0.185 & -1.402 & -1.122 & -0.861 & 0.185 \\
19 & -0.185 & -0.861 & 1.122 & -1.402 & 0.185 & -0.861 & -1.122 & -1.402 \\
37 & -0.861 & -0.185 & -1.402 & 1.122 & -0.861 & 0.185 & -1.402 & -1.122
\end{array}$$

We see that the above matrix can be represented as

$$\begin{bmatrix} C_a & C_b & C_c & C_d \\ C_b & C_a & C_d & C_c \\ C_c & C_d & -C_a & -C_b \\ C_d & C_d & -C_b & -C_a \end{bmatrix}$$

with

$$C_a = \begin{bmatrix} 1.122 & 1.402 \\ 1.402 & 1.122 \end{bmatrix}, C_b = \begin{bmatrix} -1.185 & 0.861 \\ 0.861 & -1.185 \end{bmatrix},$$

$$C_c = \begin{bmatrix} 1.122 & -1.402 \\ -1.402 & 1.122 \end{bmatrix} \text{ and } C_d = \begin{bmatrix} -1.185 & -0.861 \\ -0.861 & -1.185 \end{bmatrix}.$$

We see that the matrices C_a, C_b, C_c and C_d are cyclic matrices and the overall matrix has a Toeplitz structure. We now prove that the above is true for all cases when $n > 3$.

Theorem 4.4 : The matrix $M(h_i, g_j) = \text{cas}(\frac{h_i g_j 2\pi}{N'})$, $h_i, g_j \in A(\frac{N'}{2})$ and $N' = 2^n Q$, $n > 3$, Q odd can be expressed as

$$M(\phi(h_i), \phi(g_j)) = \delta(h_i)\, \delta(g_j)\, f(h_i \odot g_j), \quad h_i, g_j \in S,$$

where $\odot$ denotes the operation of multiplication modulo N', $\delta : S \rightarrow \{ 1 , -1 \}$ and $\phi : S \rightarrow A(N'/2)$ ϕ is a one-one onto mapping.

Proof : Note that the structure of $A(N')$ is

$$A(N'/2) = A(2^n Q) \equiv C_2 \times C_{2^{n-2}} \times A(\ Q\ ) \equiv A(4Q) \times C_{2^{n-2}}.$$

Define mapping $\phi : S \rightarrow A(N'/2)$ as $\phi(x) = x \bmod N'/2$. Now $\gcd(x, N') = 1$ implies that $\gcd(x, N'/2) = 1$, i.e, $\gcd(\phi(x), N'/2) = 1$. Also $\phi(x) < N'/2$, therefore $\phi(x) \in A(N'/2)$.To prove that ϕ is one-to-one by the method of contradiction, let $x = \alpha^{i_0}{}_{i_1}$, $y =$

$\alpha^{j_0}j_1 \in S$, $x \geq y$ and $\phi(x) = \phi(y)$. Then $x = y \bmod N'/2$, implying that $x = y$ or $y + N'/2$.

But $\phi(\alpha^{i_0}i_1) = \phi(\alpha^{j_0}j_1)$ gives

$$\alpha^{i_0}i_1 = \alpha^{j_0}j_1 \bmod N', \text{ implying } \alpha^{i_0-j_0} = (j_1 i_1^{-1}) \bmod N' \qquad (4.3)$$

or $$\alpha^{i_0}i_1 = (\alpha^{j_0} j_1 + N'/2) \bmod N', \text{ implying } \alpha^{i_0-j_0} = (j_1 i_1^{-1} + \tfrac{N'}{2}) \bmod N' \qquad (4.4)$$

Clearly (4.3) is false since $j_1 i_1^{-1}$ is an element of the group A(4Q) and therefore cannot be an element of $C_{2^{n-2}}$. To prove that (4.4) is impossible, let $g = j_1 i_1^{-1} \in G_1$. But $(N'/2 + 1) \in G_1$ implies $g(N'/2 + 1) \in G_1$, i.e,

$$(g\tfrac{N'}{2} + g) \bmod N' \in G_1.$$

Using the fact that g is odd ($\in$ A(4Q)), one gets

$$(\tfrac{N'}{2} + g) \bmod N' \in G_1.$$

But (4.4) shows that this element also belongs to $C_{2^{n-3}}$, which is a contradiction. Therefore one can conclude from $\phi(x) = \phi(y)$ that $x = y$ proving that the function ϕ is one to one.

To complete the proof one may define function $\delta : S \rightarrow \{ 1 , -1 \}$ as

$$\delta(x) = 1 \text{ if } x < N'/2,$$
$$= -1 \text{ if } x > N'/2.$$

Let $h_i, g_j \in S$. Clearly both of these are odd since $S \subset A(2^n Q)$. Consider the case of $h_i < N'/2$. One has in this case, $\phi(h_i) = h_i$ and $\delta(h_i) = 1$. Therefore

$$M(\phi(h_i), g_j) = \text{cas}(\frac{h_i g_j 2\pi}{N'}) = \delta(h_i).\ M(h_i, g_j).$$

On the other hand, if $h_i > N'/2$, then $\phi(h_i) = h_i - N'/2$ and $\delta(h_i) = -1$ and again,

$$M(\phi(h_i), g_j) = \mathrm{cas}(\pi g_j + \frac{h_i g_j 2\pi}{N'}) = \delta(h_i).\ M(h_i, g_j).$$

Therefore for all values of $h_i \in S$ one concludes that

$$M(\phi(h_i), g_j) = \delta(h_i).\ M(h_i, g_j), \qquad h_i, g_j \in S.$$

Similarly for index g_j one can get the relationship (using the fact that $\phi(h_i)$ is odd)

$$M(\phi(h_i), \phi(g_j)) = \delta(h_i)\ \delta(g_j)\ M(h_i, g_j), \qquad h_i, g_j \in S. \tag{4.5}$$

However when $h_i, g_j \in S$,

$$\begin{aligned} M(h_i, g_j) &= \mathrm{cas}(\frac{h_i g_j 2\pi}{N'}) = \mathrm{cas}(\frac{(h_i g_j \mathrm{mod}\ N')2\pi}{N'}), \\ &= \mathrm{cas}(\frac{(h_i \odot g_j)2\pi}{N'}), \\ &= f(h_i \odot g_j), \end{aligned} \tag{4.6}$$

where $\odot$ denotes the operation of set S and $f : S \rightarrow \Re$ is defined as $f(x) = \mathrm{cas}(\frac{x2\pi}{N'})$, $x \in S$. Combining (4.5) and (4.6) gives the desired result

$$M(\phi(h_i), \phi(g_j)) = \delta(h_i)\ \delta(g_j)\ f(h_i \odot g_j),\ h_i, g_j \in S\ \phi(h_i), \phi(g_j) \in A(\tfrac{N'}{2}). \quad \Box$$

Theorem 4.4 has the following computational implications. This theorem indicates that in the present case ($N' = 2^n Q$, Q odd, n>3), the kernel matrix, on shuffling of rows and columns according to the ϕ function and negating them according to the δ function, is transformed into a highly structured martix. The resultant matrix can be described as made up of elements which are multidimentional cyclic martices mimicing the structure of the group A(4Q). Further, these submatrices form a toeplitz structure as is evidenced by the fact that the (i, j)th submatrix is dependent only upon i+j, and not on individual values of i and j.

Multiplication of such a structured kernel with a vector implies a multidimentional convolution (based upon A(4Q)) with an additional dimension reserved for Toeplitz product of length 2^{n-4} (thus if $n<4$, there is no Toeplitz product to be considered, as in the section 4.3). Finally, it must be mentioned that the use of ϕ function only implies a shuffling of signal and transform components. Thus neither ϕ nor δ add to the computational complexity of the transform.

The section concludes by proving that one can use $(4Q - 1)$ as the generator α of $C_{2^{n-2}}$.

Theorem 4.5 : Order of $(4Q - 1)$ in Group $A(2^nQ)$ is 2^{n-2} $(n > 2)$.

Proof : For $n = 3$, it is easy to verify that the order of $(4Q - 1)$ in $A(8Q)$ is two as expected in the theorem. To prove the theorem by mathematical induction, assume that order of $(4Q - 1)$ in $A(2^nQ)$ is 2^{n-2}. To find order of $(4Q - 1)$ in $A(2^{n+1}Q)$, note that

$$(4Q - 1)^{2^{n-2}} = 1 \bmod 2^nQ \text{ or } (4Q - 1)^{2^{n-2}} = k.\, 2^nQ + 1.$$

Thus if k is even then

$$(4Q - 1)^{2^{n-2}} \bmod 2^{n+1}Q = 1, \text{ or}$$

$$2^{n+1}Q \mid ((4Q - 1)^{2^{n-2}} - 1),$$

i.e, $2^{n+1}Q \mid ((4Q - 1)^{2^{n-3}} - 1)((4Q - 1)^{2^{n-3}} + 1)$.

Since 4Q does not divide $((4Q - 1)^{2^{n-3}} + 1)$ $\{ ((4Q-1)^{2^{n-3}} + 1) \bmod 4Q = ((-1)^{2^{n-3}} + 1) = 2 \}$, it implies that

$$2^nQ \mid ((4Q - 1)^{2^{n-3}} - 1).$$

This implies that order of $(4Q - 1)$ in $A(2^nQ)$ divides 2^{n-3}, which is contrary to the assumption. Therefore

$$(4Q - 1)^{2^{n-2}} \bmod 2^{n+1}Q \neq 1.$$

The other choice is when k is odd then

$$(4Q - 1)^{2^{n-2}} \bmod 2^{n+1}Q = 1 + 2^nQ.$$

If L is the order of $(4Q - 1)$ in $A(2^{n+1}Q)$, it implies that $L \neq 2^{n-2}$. Also L is not less than 2^{n-2}, otherwise $2^nQ \mid ((4Q - 1)^L - 1)$ i.e, order of $(4Q - 1)$ in $A(2^nQ)$ is less than 2^{n-2}. Also

$$A(2^{n+1}Q) = C_2 \times C_{2^{n-1}} \times A(Q).$$

This implies that L must divide 2^{n-1}. Since it has been shown to be greater than 2^{n-2}, it must be 2^{n-1}. $\square$

Chapter 5
POST-PROCESSING OF CONVOLUTION

5.1. Introduction.

As has been explained in Chapter 1, the third and last stage of the algorithm proposed in this thesis is combining the results of the appropriate convolutions to produce the Hartley Transform output components. From (2.5) and (2.9), one can see that the output components with indices having the same gcd d with N are evaluated together.

In particular, $X(dg_j)$ is obtained by summing all the $X_{d,\,c}(j)$ where c's divide N/d (or N/2d if N/d is even) for every $g_j \in A(N/d)$. Further, all $X_{d,\,c}(j)$'s are not independent. One has the following results that may be used to find dependent $X_{d,\,c}(j)$'s.

$$X_{d,\,c}(j) = X_{d,\,c}(j') \tag{5.1}$$

if 4 does not divide N/cd and $m(N/cd) = g_j - g_{j'}$ for some m

and $$X_{d,\,c}(j) = (-1)^m X_{d,\,c}(j'), \tag{5.2}$$

if $4 \mid (N/cd)$ and $m(N/2cd) = g_j - g_{j'}$, for some m.

In order to improve the readability of this chapter, we use the following notation.

$g_j = u$ and $\gamma(u) = j$.

Thus γ is the inverse function of g_j from the group A(N/d) into the set of integers { 0, 1, ... , |A(N/d)| − 1 }. With this new notation (5.1) and (5.2) take the following simple form :

$$X_{d,\,c}(\gamma(u)) = X_{d,c}(\gamma(u \bmod (N/cd)), \text{ if 4 does not divide } N/cd, \qquad (5.3)$$

and $$X_{d,\,c}(\gamma(u)) = (-1)^{\left\lfloor \frac{u}{(N/2cd)} \right\rfloor} X_{d,\,c}(\gamma(u \bmod (N/2cd))), \text{ if } 4 \mid N/cd \qquad (5.4)$$

It will be shown in this chapter that combining the $X_{d,\,c}$'s to produce the transform componets is a systematic operation and can be carried out through regular computational graphs.

5.2. Transform length, a power of 2.

Let the transform length, $N = 2^n$ and $d = 2^x$, $x<n$. Clearly, $A(N/d) = \{ 1, 3, 5, \ldots, 2^{n-x} - 1 \}$. The output can be written in terms of $X_{d,\,c}$'s as follows :

$$X(2^x u) = \sum_{c \mid 2^{n-x-1}} X_{2^x,\,c}(\gamma(u)) \quad \text{where } 1 \leq u \leq 2^{n-x}, \text{ u odd.}$$
$$= \sum_{l=0}^{n-x-2} X_{2^x,\,2^l}(\gamma(u)) + X_{2^x,\,2^{n-x-1}}(\gamma(u)).$$

Now using (5.4) within the first summation and (5.3) in the last term gives

$$X(2^x u) = \sum_{l=0}^{n-x-2} (-1)^{\lfloor u.2^{-n+x+l+1} \rfloor} X_{2^x,\,2^l}(\gamma(u \bmod 2^{n-x-l-1}))$$
$$+ X_{2^x,\,2^{n-x-1}}(\gamma(u \bmod 2)). \qquad (5.5)$$

Even through at first sight (5.5) looks complex, it can be shown to be recursive in nature. Define blocks B_{2^s}, $s = 0, 1, \ldots, n\text{-}x$ with 2^{s-1} outputs, recursively as

$$B_1(0) = X_{2^x,\,2^{n-x-1}}(\gamma(1)),$$

and $$B_{2^{s+1}}(i) = B_{2^s}(i) + X_{2^x,\,2^{n-x-s-2}}(\gamma(2i+1)),$$

$$B_{2^{s+1}}(i + 2^{s-1}) = B_{2^s}(i) - X_{2^n,\,2^{n-x-s-2}}(\gamma(2i+1)),$$

$i = 0, 1, ..., 2^{s-1}-1$ and $s = 0, 1, ..., n\text{-}x\text{-}1$,

then one can easily see that (5.5) can be expressed as

$$X(2^x(2i+1)) = B_{2^{n-x}}(i), \; i = 0, 1, ..., 2^{n-x-1}-1. \qquad (5.6)$$

Fig. 5.1. illustrates the calculation of the output points cerresponding to N = 16 and d = 1. The equation (5.5) can be seen to produce a regular graph. Notice the addition of $X_{1,16}(0)$ to $X_{1,8}(0)$. This adds the signal component with index zero which was not considered as parts of sets of nonzero indices. Fig. (5.2) shows the same calculation illustrating the use of (5.6). The modularity of this implementation is an attractive feature for use of parrallel architectures.

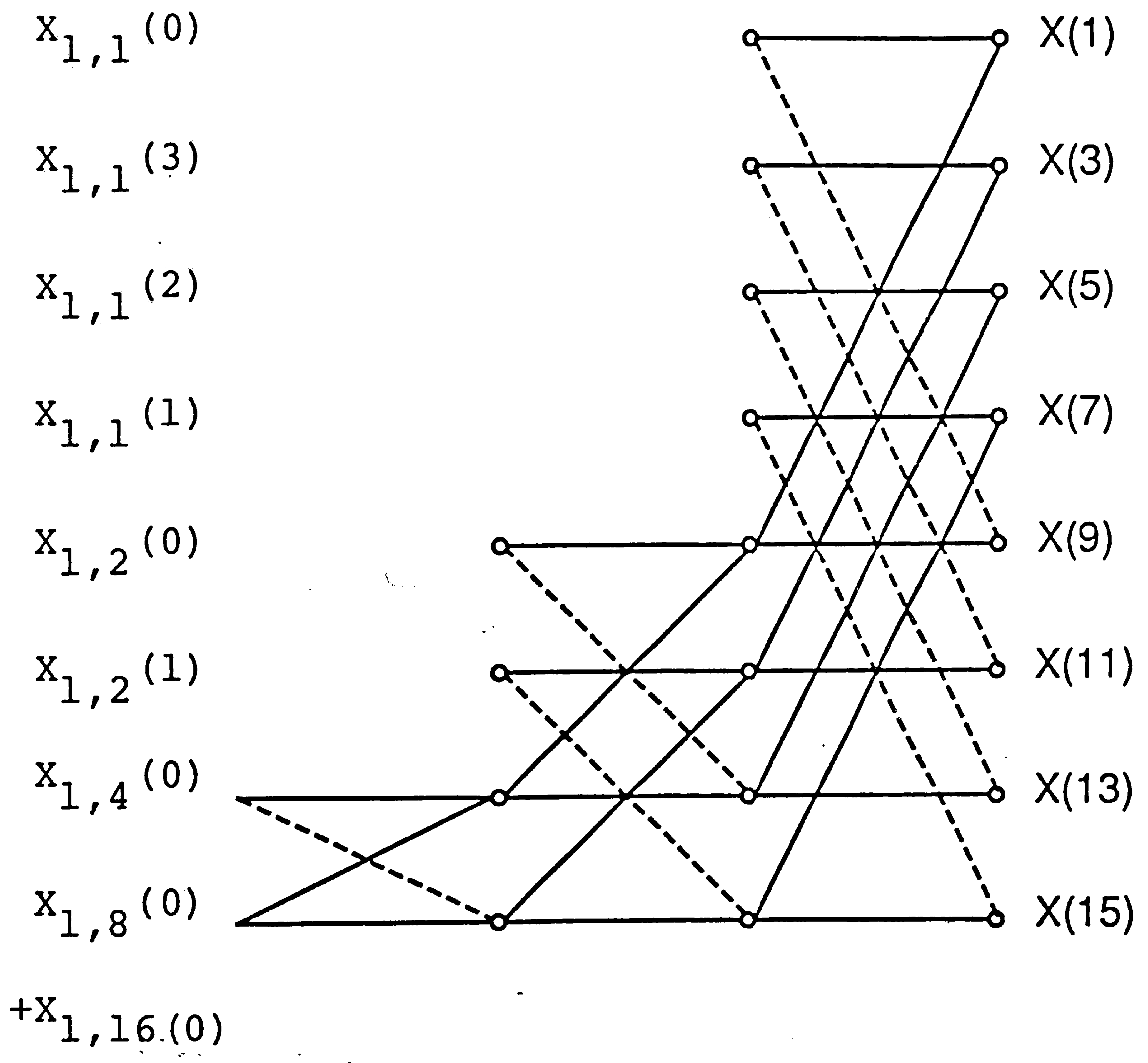


Fig 5.1. Calculation of Hartley transform output points for N=16 and d=1

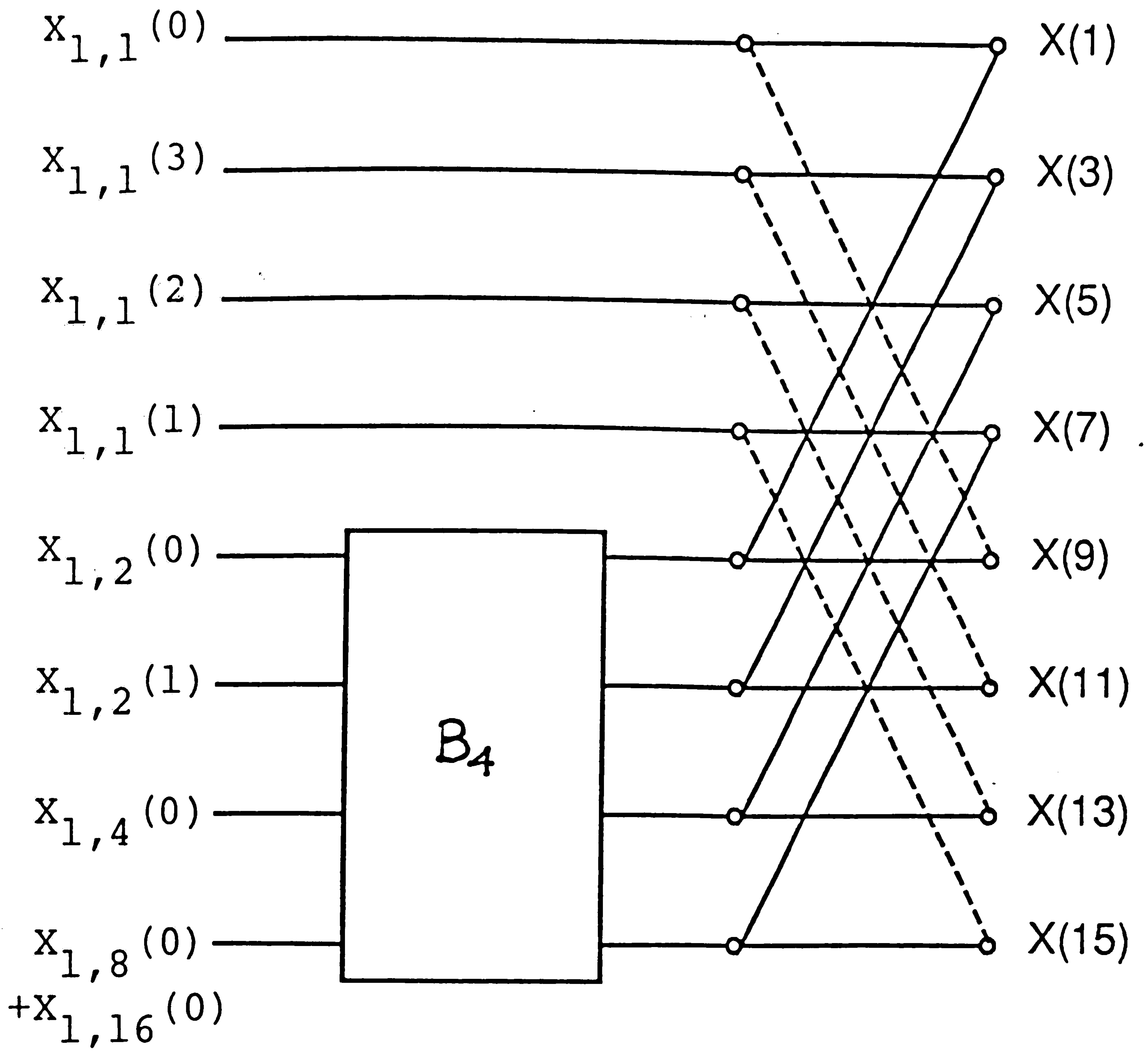


Fig 5.2. Construction of box B_8 from B_4 and its use in calculation of output points N=16, d=1

One can see that the complexity of the box B_{2^s} is 2^s additions. Thus the total complexity of the post processing stage when $N = 2^n$ is given by

$$\sum_{x=1}^{n-1}\sum_{s=1}^{n-x} 2^s = 2(2^n - n - 1) \tag{5.7}$$

The inner summation in (5.7) provides the number of additions required to compute all output points with $d = x$ and the outer summation mearly sums this number over all groups with different d's.

5.3. Length, power of an odd prime.

When $N = p^n$, p being an odd prime, for every possible c and d, 4 does not divide (N/cd). Thus (5.3) applies uniformly in all these cases. Suppose $d = p^x$, $0 \le x < n$. Then one has, for $u \in A(p^{n-x}) = \{ 1 \le u < p^{n-x} \mid p \text{ does not divide } u \}$,

$$X(p^x u) = \sum_{c \mid p^{n-x}} X_{p^x,\, c}(\gamma(u)). \tag{5.8}$$

Clearly, in (5.8), c = p^l, $0 \leq l \leq$ n-x. Using this fact along with (5.3) gives

$$X(p^x u) = \sum_{l=0}^{n-x} X_{p^x,\, p^l}(\gamma(u \bmod p^{n-x-l})). \tag{5.9}$$

Equation (5.9) can be implemented in a regular or in a block fashion as for done for the case of N = 2^n. However, it might be more instructional to look first at case when N = p, i.e., n = 1. In this case, d has an unique value of $p^0 = 1$ and therefore (5.9) gives

$$X(u) = \sum_{l=0}^{1} X_{1,\, p^l}(\gamma(u \bmod p^{1-l})),$$

$$= X_{1,1}(\gamma(u)) + X_{1,p}(\gamma(0)). \tag{5.10}$$

The absence of the modulo function in the first term is explained by the fact that in the present case, u < p. The relation (5.10) may be implemented as in Fig. 5.3 with a complexity of (p-1) additions. This implementation is called the block B_p.

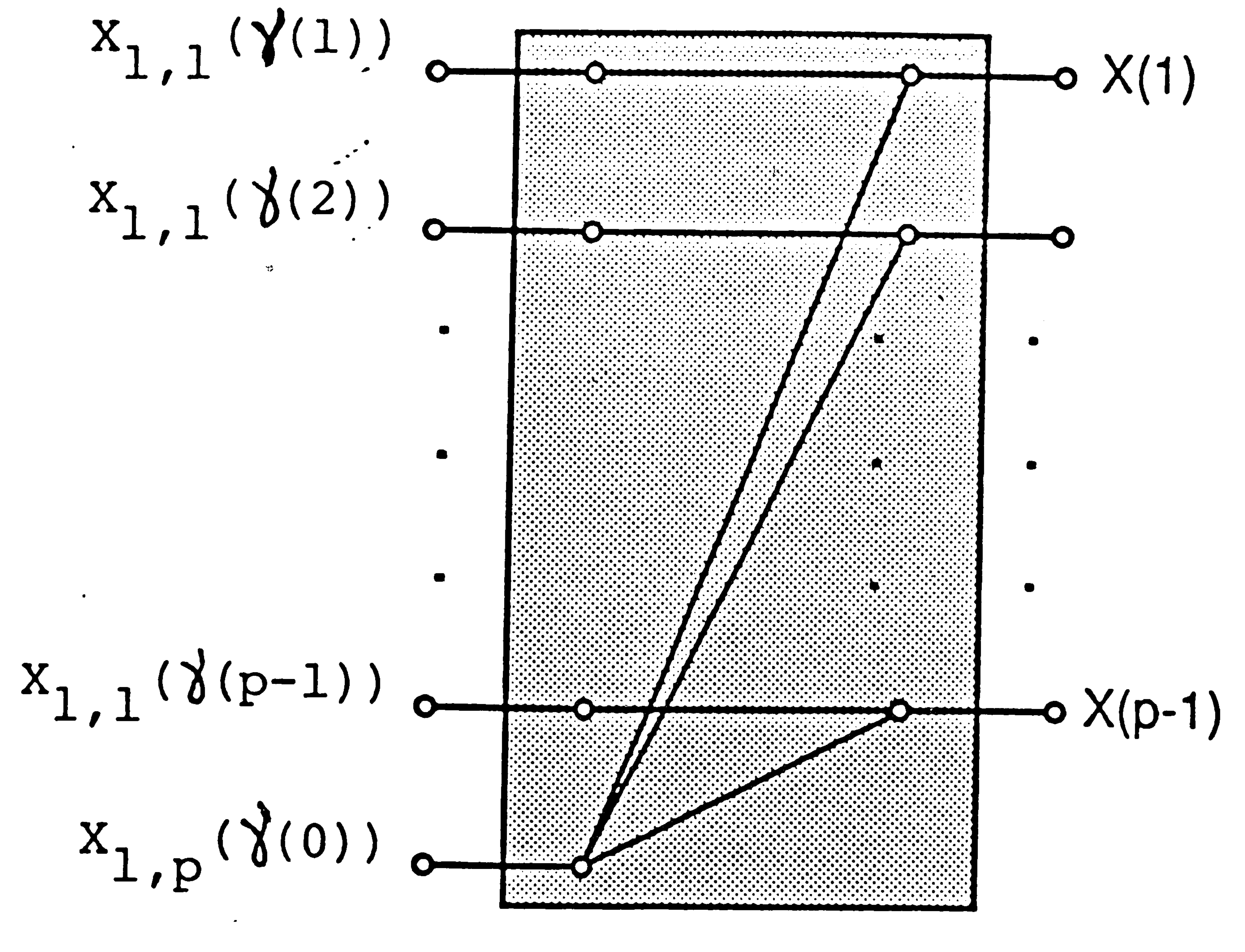
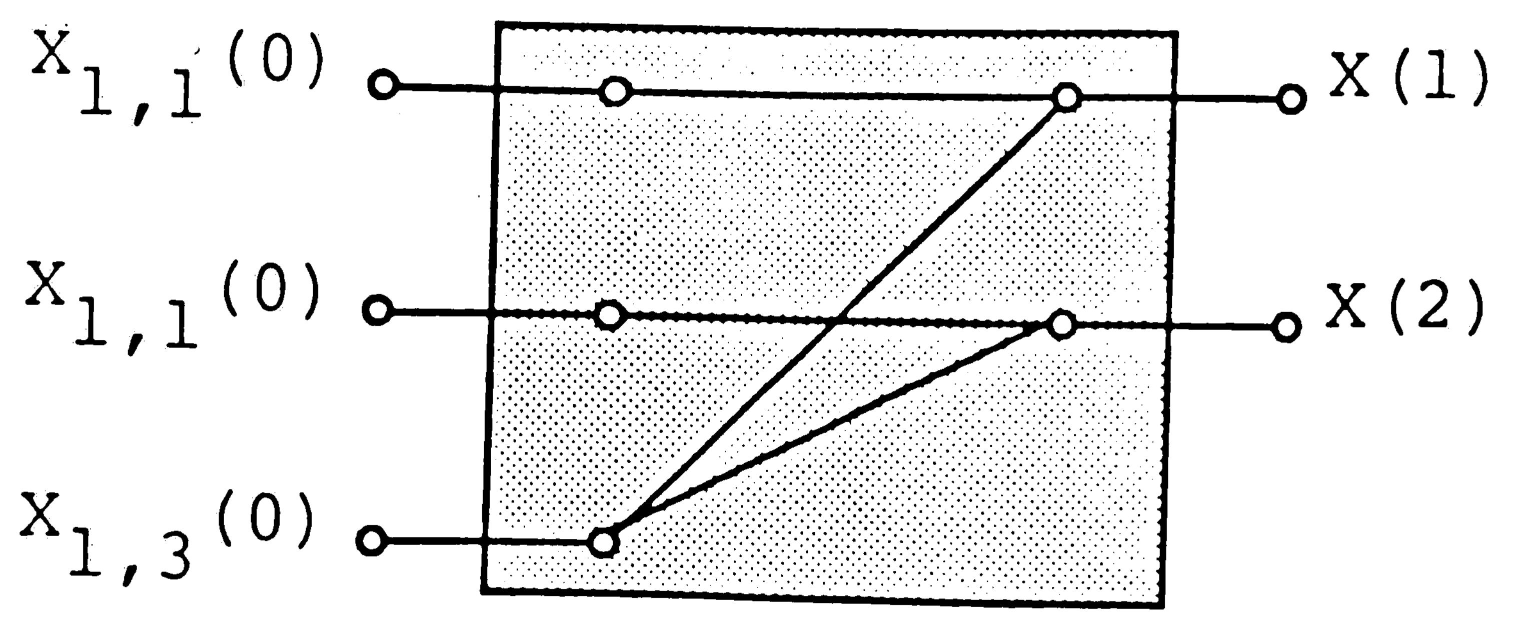
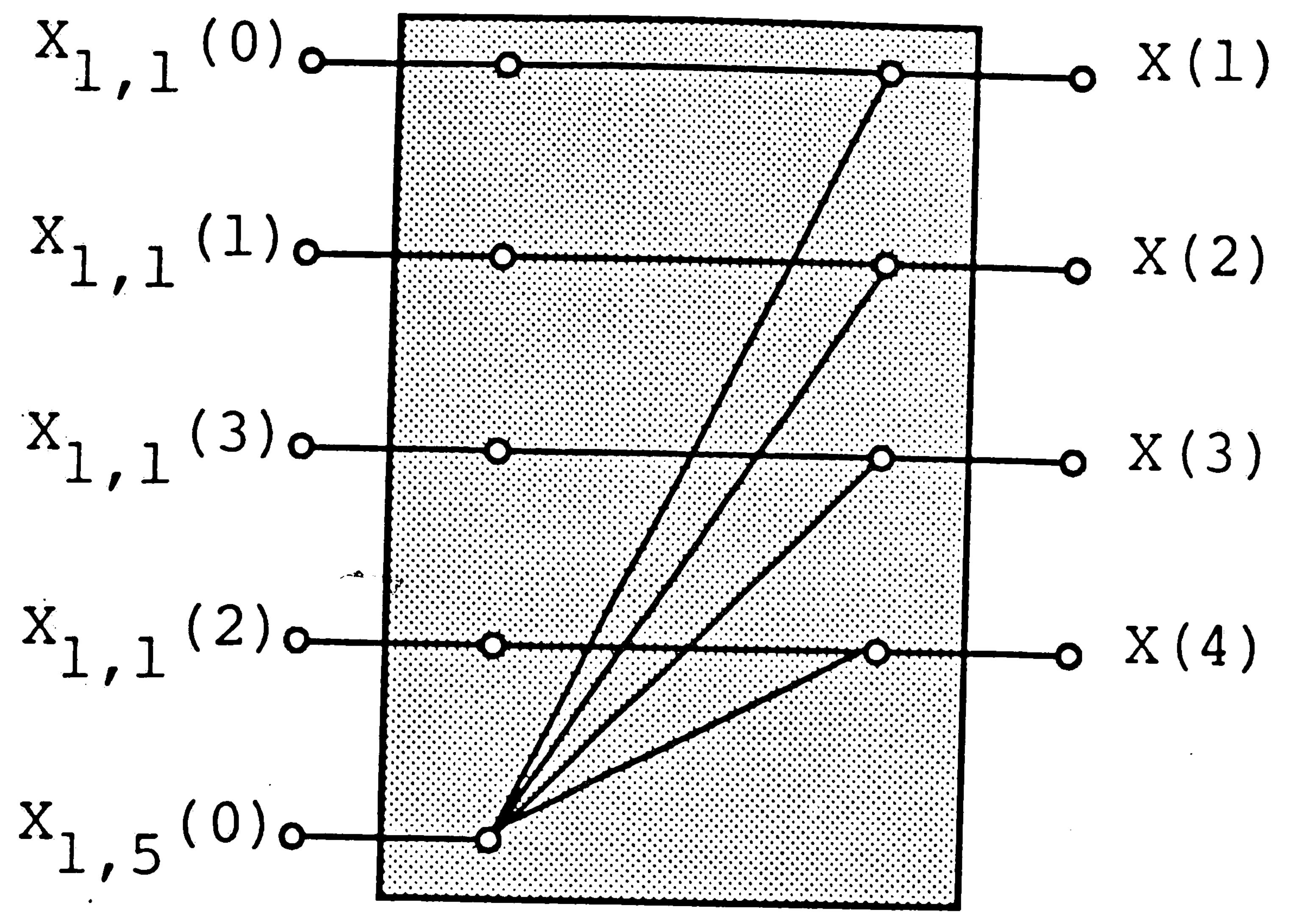


Fig 5.3(a) The general box B_p



B3

Fig. 5.3(b) Calculation of X(1), X(2) for N=3 illustrating Box B_3



B5

Fig. 5.3(c) Calculation of output for group A(5) illustrating Box B_5

When one is dealing with a length p^n, $n > 1$, the block structure of (5.9) can be brought out as follows :

Define blocks B_{p^s}, $s = 0, 1, \ldots, n\text{-}x$ to have $(p\text{-}1)p^{s\text{-}1}$ outputs as :

$$B_1(0) = X_{p^x,\, p^{n-x}}(\gamma(0)),$$

and $B_{p^{s+1}}(i + (p\text{-}1)p^{s\text{-}1}j) = B_{p^s}(i) + X_{p^x,\, p^{n-x-s-1}}(\gamma(i + (p\text{-}1)p^{s\text{-}1}j)),$ (5.11)

$$i = 0, 1, \ldots, (p\text{-}1)p^{s-1}-1 \text{ and } j = 0, 1, \ldots, (p\text{-}1).$$

One can easily show that in this new notation, the relation (5.9) transforms to

$$X(p^x u) = B_{n-x}(u), \;\; 0 \leq u < p^{n-x}. \qquad (5.12)$$

Thus the post processing stage is indeed regular and recursive as (5.11) shows. Fig. (5.4) illustrates a typical box heirarchy.

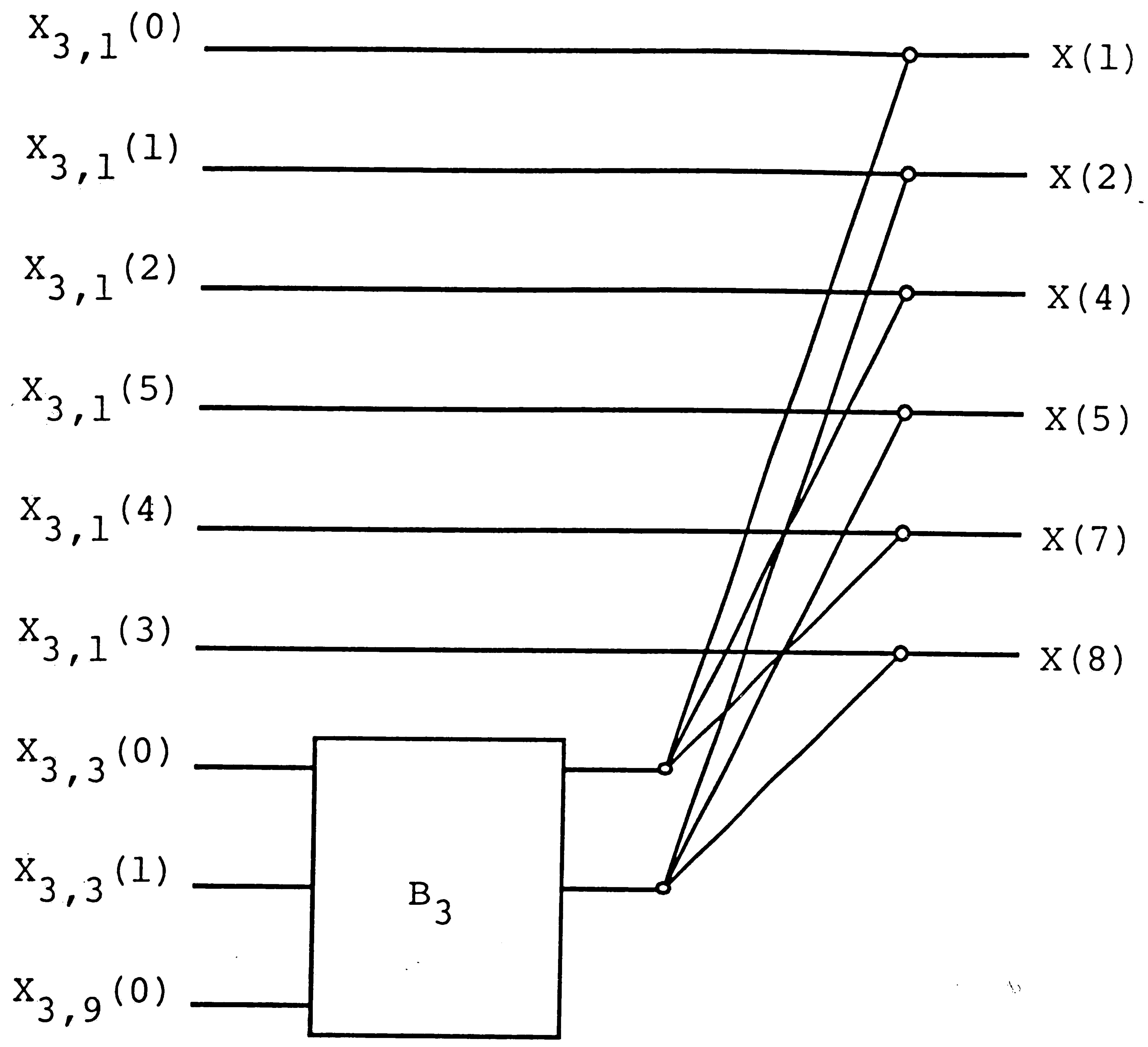


Fig. 5.4. Calculation of the transform points for N = 27 and d = 3 illustrating Box B_9 recursively obtained from B_3

One can also see that in a block B_{p^s}, there are $(p\text{-}1)p^{s-1}$ additions. Thus the total number of post processing additions in the computation of all the transform points when

$N = p^n$ is given by

$$\sum_{x=1}^{n-1} \sum_{s=1}^{n-x} (p^s - p^{s-1}) = \frac{(p^n-1)}{(p-1)} - n.$$

5.4. Length, product of relatively prime factors.

Section 5.2 and 5.3 have considered the post processing when the length is power of a prime. The case of $N = q_1q_2$ where $\gcd(q_1, q_2) = 1$ is considdered in this section. Together these results enable one to develop the post processing stage of the algorithm for any arbitrary length. Here one may use a multidimentional approach to do summations (2.5) and (2.9). If N is factored as q_1q_2 and one uses B_{q_1} blocks first and multiplexes their output through the B_{q_2} blocks, the number of additions is given by

$$q_2(\#\text{ of adds in } B_{q_1} \text{ block}) + |A(q_1)|(\#\text{ of adds in } B_{q_2} \text{ block}).$$

If instead one uses the B_{q_2} blocks first, the number of additions are

$$q_1(\#\text{ of adds in } B_{q_2} \text{ block}) + |A(q_2)|(\#\text{ of adds in } B_{q_1} \text{ block}).$$

One can easily see from this that if

$$\frac{q_1 - |A(q_1)|}{\#\text{of adds in } B_{q_1}} > \frac{q_2 - |A(q_2)|}{\#\text{of adds in } B_{q_2}},$$

then use of B_{q_1} blocks first and B_{q_2} blocks later leads to a smaller complexity.

The relation above has important consequences. When q_1 and q_2 are primes, as shown in section 5.3, number of additions in B_{q_1} and B_{q_2} are (q_1-1) and (q_2-1) respectively. Also $|A(q_1)| = q_1 - 1$ and $|A(q_2)| = q_2 - 1$. Thus one should use B_{q_1} before B_{q_2} iff

$$\frac{q_1 - (q_1 - 1)}{(q_1 - 1)} > \frac{q_2 - (q_2 - 1)}{(q_2 - 1)}, \text{ i.e, iff } q_1 < q_2.$$

The (minimal) complexity in this case is $q_2(q_1-1) + (q_1-1)(q_2-1) = (q_1-1)(2q_2-1)$. Thus, for example, when $N = 15$ and $d = 1$, the 8 transform points in the set S_d can be calculated using 18 post processing additions if one uses B_3 blocks first. If instead, B_5 was used first, one would have needed 20 additions.

Chapter 6
CONCLUSION

6.1. Summary of results.

Signal processing algorithms need efficient computational architectures and algorithms to process increasingly large volume of data encountered in modern day signal processing techniques. The work presented in this thesis deals with efficient computation of discrete Hartley transform which is becoming a new popular signal processing tool. The algorithms presented in this thesis are radically different from other algorithms available in literature in that it, for the first time, it meshes the group theoretic techniques with the properties of the Hartley transform itself. It allows one to identify the multidimensional cyclic structures hidden within the Hartley transform kernel and thus compose the transform result from many cyclic convolutions. Since efficient techniques are now available for cyclic convolution, the resultant procedure turns out to be computationally superior to the other known techniques. Table 6.1 presents the computational complexities of the algorithm proposed in this thesis for certain values of transform lengths.

Table 6.1. Complexity of the proposed algorithm

N	preprocessing	convolution		postprocessing	Total	
	adds	mults	adds	adds	mults	adds
8	14	2	0	8	2	22
12	22	8	16	18	8	56
15	22	22	78	26	22	126
16	30	10	14	22	10	66
30	78	48	152	53	48	283
32	62	36	74	52	36	188

The complexities of the proposed algorithms are compared with those of Bracewell [7] (algorithm B), Mackelberg and Lipka [9] (algorithm ML) and Sorenson et. al. [10] (algorithm SJBH) in Table 6.2. One can see from this table that the multiplicative complexity of the proposed algorithm is less than that of previous algorithms and the additive complexity is comparable. It must be mentioned here that there is further scope for optimizing the proposed algorithms. In particular, an examination of the specific values of elements within the cyclic matrices will enable one to reduce the operation counts (and specifically the number of additions). No such attempt to do this was made, however, because the main thrust of this thesis was to derive techniques to find multidimentional cyclic matrices within the transform kernel using group techniques.

Table 6.2. Comparison of the complexity of the proposed algorithm with other algorithms

N	algorithm B		algorithm ML		algorithm SJBH		new algorithm	
	mults	adds	mults	adds	mults	adds	mults	adds
8	-	-	-	-	2	22	2	22
16	34	98	20	74	12	64	10	66
32	98	258	68	194	42	166	36	188

Another major advantage of the proposed algorithm seems to be its applicability to almost any length. The algorithms reported in litrarure are limited to transforms of 2^n point sequences. Finally, the new algorithms are highly regular and show a high degree of parallelism. This implies that it may be implemented as an application specific integrated circuit (ASIC) as well as used on parallel architecture.

6.2. Future extensions.

The work presented here opens new avenues for further research in two directions. Firstly, it was often observed that the sequences made up of elements from Hartley kernel and taking part in cyclic convolutions exhibit certain interesting relationships. These relationships, if systematically predicted and exploited, can result in an algorithm with lower computational complexity. For example, while calculating the d=1 output points of the discrete Hartley transform of 9 points, one encounters the two dimensional cyclic structure C_3 X C_2 (for c=1). This results in a cyclic convolution of two dimensional patterns of 3 X 2 points. Interestingly, both the length 3 rows in one of the patterns add to zero. If this fact is used in the evaluation of that two dimensional convolution, one will be able to cut down on a large number of multiplications and additions.

The second research direction is to study parallel architectures to evaluate Hartley transform. The algorithm presented has regular recursive structure as has been proved in this work. This observation about the algorithm is true for all three of its stages. The algorithm may also possess greater regularity at a global level. The execution time of any algorithm does finally depend upon its implementation just as much as the operations count, and a study of the parallel implementation of this algorithm will therefore be a very worth while project.

References

[1] G. Goertzel, "An Algorithm for the evaluation of finite Trigonometric Series," *Amer. Math. Monthly*, vol. 65, pp. 34-35, Jan. 1958.

[2] J. W. Cooley and J. W. Tukey, "An Algorithm for the machine calculation of Complex Fourier Series," *Math. Computation*, vol. 19, pp. 297-301, 1965.

[3] L. I. Bluestien, "A linear Filtering Approach to the Computation of discrete Fourier Transform," *IEEE Audio Electroacoust.*, vol. AU-18, pp. 451-455, Dec. 1970.

[4] S. Winograd, "On computing the Discrete Fourier Transform," *Math. of computation*, vol. 32, no. 141, pp. 175-199, Jan. 1978.

[5] R. V. Hartley, "A more symmetrical Fourier analysis applied to Transmission problems," *Proc. Inst. Radio Engrs., pp.* 144-150, March 1942.

[6] R. N. Bracewell, "Discrete Hartley Transform," *J. Opt. Soc. Amer. Lett.*, vol. 73, pp. 1832-1835, 1983.

[7] R. N. Bracewell, "The fast Hartley Transform," *Proc. IEEE*, vol. 72, pp. 1010-1018, Aug. 1984.

[8] Chan Yun-Hsu and Ja-Ling Wu, "Fast computation of Discrete Hartley Transform via Walsh-Hadamard Transform," *Electronic Letters*, vol. 23, No. 9, 1987.

[9] H. J. Meckelberg and D. Lipka, "Fast Hartley Transform Algorithm," *Electronic Letters*, vol. 21, No. 8, 1985.

[10] H. V. Sorenson, D. L. Jones, C. S. Borres, M. T. Heideman, "On computing the discrete Hartley Transform," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-33, pp. 1231-1238, 1985.

[11] N. Suchiro and M. Hatori, "Fast algorithms for DFT and other sinusoidal transforms," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-34, pp. 642-645, 1986.

[12] R. C. Agarwal and J. W. Cooley, "New Algorithm for Digital Convolution," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-28, No. 5, Oct '77.

Vita

Mukesh Sharma was born to Mr. and Mrs. S. N. Sharma in Dec '65. After graduating from Kendriya Vidyalaya, Kanpur, he went on to get a Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, in May 87. During this period he received the merit scholarship from the Indian Institute of Technology, Kanpur. Subsequently, he was enrolled for a Master of Science degree in Computer Science and Electrical Engineering at Lehigh University, Bethlehem.

After his graduation in Oct '89, he is going to continue at Lehigh University for a degree of Doctor of Philosopy in the department of Computer Science and Electrical Engineering.

His professional interests are in computer system architecture, parallel algorithms and processing and distributed computer system architecture.